Data Science with Foundation Models

An Evidence-Based, Comprehensive Project Methodology

Dissertation
zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

von

Sven Alexander Gießelbach
aus
Troisdorf

Bonn, 2023

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter:	Prof. Dr. Stefan Wrobel
2. Gutachter:	Prof. Dr. Christian Bauckhage
Tag der Promotion:	17.01.2024
Erscheinungsjahr:	2024

Acknowledgements

I extend my profound gratitude to Prof. Dr. Stefan Wrobel, whose guidance and invaluable feedback on this enthralling topic have been indispensable and who has taken time for me even when it meant missing out on lunch. Deep appreciation goes to Prof. Dr. Christian Bauckhage for his invaluable advice from my studies to my PhD, and to Prof. Dr. Lucie Flek and Prof. Dr. Ulrike Attenberger for joining my thesis committee.

Stefan Rüping deserves a special mention for his support in discussions, proofreading, guidance throughout and for having my back by keeping work away from me. Dirk Hecker, whose original idea propelled this thesis, has been a consistent pillar of support. Georg, Claus, Hendrik, Tim and others who have gladly taken on tasks to enable me to work on my thesis. My colleagues at KD and Fraunhofer IAIS have created a stimulating environment, with a special nod to team NLU for their unwavering support even when it meant more work for them. Thank you Jörg and Gerd for guiding me with your experience.

Gratitude is due to the incredible PhD peers who enriched my journey, notably Daniel Trabold, Michael Kamp, Linara Adilova, Dorina Weichert, Sebastian Konietzny, Daniel Paurat, Pascal Welke, Nathalie Paul, and Jil Sander. Many thanks to those who reviewed my thesis, especially Dario Antweiler, Joachim Sicking, Katharina Beckh, Katrin Klug, Maram Akila, Niclas Doll, Lukas Pin and Dennis Wegener. Niclas Doll and Alexander Zorn thank you for staying with me even late at night. Special thanks to Katja Wolters for keeping me on track.

Birgit Kirsch's dedication in sifting through countless publications and helping refine my figures late at night was invaluable. Barbara Lix imparted essential business insights and confidence. I could not wish for a better mentor. Angelika Voß, your clarity and structuring prowess have been irreplaceable. I cannot thank you enough.

To my friends, including Patrick, Franzi, Sevi, Johannes, Gulli, Christian, Richard, Christos, Alina, Nici, Pira, the NYU crew, and many others - your understanding and support mean everything. Immense thanks to my extensive family, with a special mention of Goko, Ace, Jan, Iva, Ana, Ivana, Maja, Goce, Valentina, Zaklina, Naste, Bobi, Lidija, Stevco, Jone, Sneza, Carla, Markus, Ulrike, Lukas, Andreas, and more.

To mum and dad, your sacrifices and relentless support are beyond words. Finally, Judith, your love and understanding have been the backbone of this journey. Without the three of you, this thesis would remain incomplete.

Abstract

In the past decades, the interest in data science has surged across sectors and industries. Data science has been impacted by a series of paradigm shifts, such as the emergence of big data, cloud computing or the demand for developing and operating machine learning-based applications. Project methodologies have continually strived to catch up, but with limited success, as 80% of data science projects never reach deployment. While initially data science projects ended once a useful model was obtained from carefully engineered data, nowadays models are integrated into software applications which afterwards must be maintained, improved, and adapted. This requires careful management of all kinds of digital project artifacts.

The latest paradigm shift in machine learning, which has not yet been reflected in any project methodology so far, is due to foundation models, which have established themselves as a de facto standard across various media including text, images, video, and audio. Among them large language models play a central role, as they allow reuse for other tasks, without additional training, by prompting them with natural language. Besides, a foundation model can be fine-tuned to new tasks and domains with limited data, while building a foundation model from scratch requires enormous amounts of data and careful provisions for alignment. Consequently, the nature of data science projects has changed.

This thesis proposes the first methodology specifically crafted for modern data science projects, placing foundation models at its core. It seamlessly integrates application development phases into the project lifecycle, adopts an encompassing view of project management, and places a particular emphasis on artifact management for cross-project business purposes. A guiding principle of our methodology is the persistent engagement of domain experts and users throughout the project lifecycle. Recognizing the limited availability of these stakeholders, we propose specific tools to enhance their efficiency, particularly in labor-intensive project activities such as data annotation, modeling, and evaluation.

The methodology fully covers a hierarchical catalog of requirements, which comprises 163 groups of project characteristics. The catalog is comprehensive and evidence-based, derived from two exhaustive literature studies. The first study provides a broad and general overview of data science projects, while the second examines the state-of-the-art literature on foundation models. The catalog is validated theoretically by matching it to 8 meta studies on data science methodologies, and practically by case studies from 26 natural language understanding projects.

The catalog provides a framework for assessing data science methodologies. 27 such methodologies are compared, and a gap analysis confirms weaknesses in each methodology. The most prominent gaps are the insufficient support for application development and cross-project artifact management, making them inadequate for modern foundation model projects. All the identified gaps are covered by our proposed methodology, proving it to be both comprehensive, dedicated to projects with foundation models and fostering cross-project reuse.

The ecosystem and frameworks around foundation models are evolving rapidly and we expect them soon to offer tools that can facilitate project management processes in our methodology.

Contents

CHAPTER 1

Introduction

1.1 Background and Motivation

In the past decades, data science has seen a surge of demand, with market volume estimates projected to grow from $32 billion in 2021 to an astonishing $330 billion by 2030 [1]. There are many drivers to this growth, such as the exponential increase in available data, the need for software which automates processes and the improvement of key technologies in the field, first and foremost machine learning, which enable new use cases.

Data science has always been a primarily data-driven science, i.e., data science projects are centered around processing, analyzing and managing data. However, it has been impacted by various developments and shifting customer expectations such as big data, agile development methods, cloud computing, the development as well as continuous maintenance and operation of machine learning-based applications.

Data science project methodologies, or data science methodologies for short, provide guidance for project managers on how to execute projects. They structure projects into phases and processes, define relevant project roles, explain which artifacts to use and produce and suggest tools and techniques. While project management methodologies in software development have matured over time, project management in data science is still an evolving field. Due to the significant differences between data science and software development, software development methodologies cannot simply be reused in data science without adaptation. Generally, methodologies should be adapted to the changing requirements, challenges and success factors of data science projects.

The most popular data science project methodology to date is CRISP-DM [2], which is more than 20 years old. CRISP-DM provides a generic view on data science projects. It structures them into six phases, ranging from business understanding to deployment. It suggests taking an iterative approach to projects, proposes tasks for each phase and gives detailed information about project artifacts. One of the strengths of CRISP-DM is its flexibility. The authors of CRISP-DM explicitly encourage data science teams to tailor CRISP-DM to the specific requirements and challenges of their project [2]. Simultaneously, this is one of its biggest weaknesses. CRISP-DM more or less fits to every data science project but is not tailored to any project specifically. Tailoring methodologies is a difficult process [3] and rather than making the effort of tailoring CRISP-DM, studies show that most teams fall back to so-called ad-hoc methodologies [4]. This means they either use a project-specific methodology or an undefined methodology. A solution to this problem are methodologies which are better tailored

to the current requirements and the challenges teams face in their field.

Along with the evolution of data science, many data science project methodologies have been suggested to address new requirements. With the introduction of big data, specific methodologies were designed to tackle its challenges [5]. Another wave of methodologies tackled the challenge of agile development in data science [6]. The introduction of cloud platforms for developing and deploying has led to specific methodologies, tailored to the specific tools provided in these platforms [7]. Most recently, the operation of machine learning models has gained more interest, as more and more companies aim to include machine learning models in their productive environments and applications [8].

However, it appears the methodologies fail to address the practical requirements adequately. According to a survey from IDC, most companies report failures of data science projects, 25% of companies even report that 50% of their data science projects failed [9], i.e., were aborted or did not achieve their desired goals. Further, a survey from Alegion Survey shows that around 80% of projects never reach the stage of deployment [10]. Saltz [11] argues that the reason for data science project failures is the sole focus of the scientific community on improving models and algorithms and the neglect of project management. Modern data science projects include application development. Comprehensive methodologies should cover processes, roles, artifacts and tools.

Recently, another fundamental development has changed the way in which data science projects are executed. The development of foundation models, in particular large language models, has led to many breakthroughs in machine learning. These models can cope with inputs in the form of various media, can be adapted via natural language prompting or fine-tuning with small amounts of data and can hence be reused across projects and tasks. Not only do these models beat scientific benchmarks in every natural language understanding task [12], but they have also gained attention by media and companies. The GPT-4 model has successfully passed human exams in the upper percentiles [13], AlphaCode has scored better than the average programmer in a coding competition [14] and the novel chatbot ChatGPT, has become one of the most rapidly adopted applications ever [15].

With foundation models, a new paradigm of reusing and adapting general models to specific tasks and datasets has emerged, significantly reducing modeling efforts. Natural language understanding (NLU) is arguably the subfield of data science which adopted foundation models first. The emergence of large language models has revolutionized the way NLU projects are executed. Large language models, as well as other foundation models, are trained in a self-supervised manner, i.e., without requiring costly annotated data, to form a general understanding of language. This enables transfer learning, i.e., the general models can be adapted to specific tasks, which gives these models value outside of a single project. They are artifacts which can be used across projects and can be further refined to understand data in specific domains of expertise. Hence, modeling can be as simple as downloading a model and using it off the shelf or adapting it. Adaptation can be done via fine-tuning on smaller datasets or via prompting, i.e., giving the model natural language instructions to perform a task without any further training. This is in stark contrast to classical machine learning approaches, which require a lot of effort for feature engineering and modeling.

We argue that this technological shift has led to new requirements, success factors and challenges in projects which make use of foundation models, in short foundation model projects, which need to be addressed specifically by a project management methodology. In particular, this calls for a holistic approach to project management, i.e., embedding a project and its artifacts in the context of other projects of a team. Teams that use foundation models should pay special attention to the versioning and reusability of models and data. Finally, the need for mature software solutions also

calls for reusable software components, which are reused and developed across projects. Even recent methodologies which are tailored towards NLU projects ignore foundation models and their implications on comprehensive project management [16, 17].

1.2 Our Contributions

A complete overview of the thesis and our contributions is displayed in Figure 1.1 and Figure 1.2. The core contribution of this thesis is a data science project methodology, tailored towards projects using foundation models such as large language models and the context in which modern enterprises operate.

The development and the empirical validation of a project methodology is a hard endeavor. A substantial evaluation would imply the independent execution of a project with multiple methodologies. To the best of our knowledge, only one such study exists in the field of data science [18] and only in a simplified laboratory setting, using students. Performing such an evaluation in real-world settings is unfeasible. Instead, we follow an evidence-based approach which uses two systematic literature studies, one for data science project management literature and one for state-of-the-art foundation model literature, and 26 NLU case studies as empirical basis.

In Chapter 2 we introduce the necessary preliminaries to understand this thesis. We explain how the field of data science has evolved over the years. We introduce foundation models and their building blocks. In particular, the transformer architecture and the concept of pre-training and adapting models. We proceed to introduce the most important concepts in project management and project management methodologies. We propose four key components of data science project methodologies: (1) Their lifecycle, phases and processes, (2) definitions of roles and responsibilities, (3) artifacts and (4) the tools and techniques.

In Chapter 3 we derive a catalog of project characteristics, which represent the requirements that a methodology needs to meet that deals with foundation models. This catalog is based on two systematic literature studies and the analysis of 26 case studies from an NLU team. It starts by illustrating how paradigm shifts in data science have given rise to novel project methodologies. Five major paradigm shifts are identified: (1) the shift from software engineering to data science, (2) the emergence of big data, (3) the use of agile development methods, (4) the development of integrated frameworks, providing a methodology tailored around a specific software and (cloud) infrastructure and (5) the trend towards bringing machine learning applications into operation. In our first literature study, we analyze project management literature in data science. The study provides us with a broad and general overview of data science projects and their characteristics. We organize them into a hierarchical structure. Since the project management literature lacks information on foundation models, we perform a second literature study. In this literature study we analyze state-of-the-art literature on foundation models. We incorporate our insights into the catalog of characteristics. The final catalog contains 163 aggregated characteristics. Following a comprehensive empirical examination of 26 projects executed by an NLU team, we extract significant characteristics that have noticeably impacted the progression and outcomes of these projects. By aligning our results with the corresponding characteristics within our catalog, we create a more condensed version, particularly relevant to the scope of the examined projects. This project-relevant version incorporates 45 characteristics, providing a condensed but potent resource for understanding and predicting project outcomes in the context of NLU projects.

In Chapter 4 we investigate the hypothesis that none of the existing methodologies is suitable for modern foundation model projects. 27 methodologies are introduced and grouped based on the

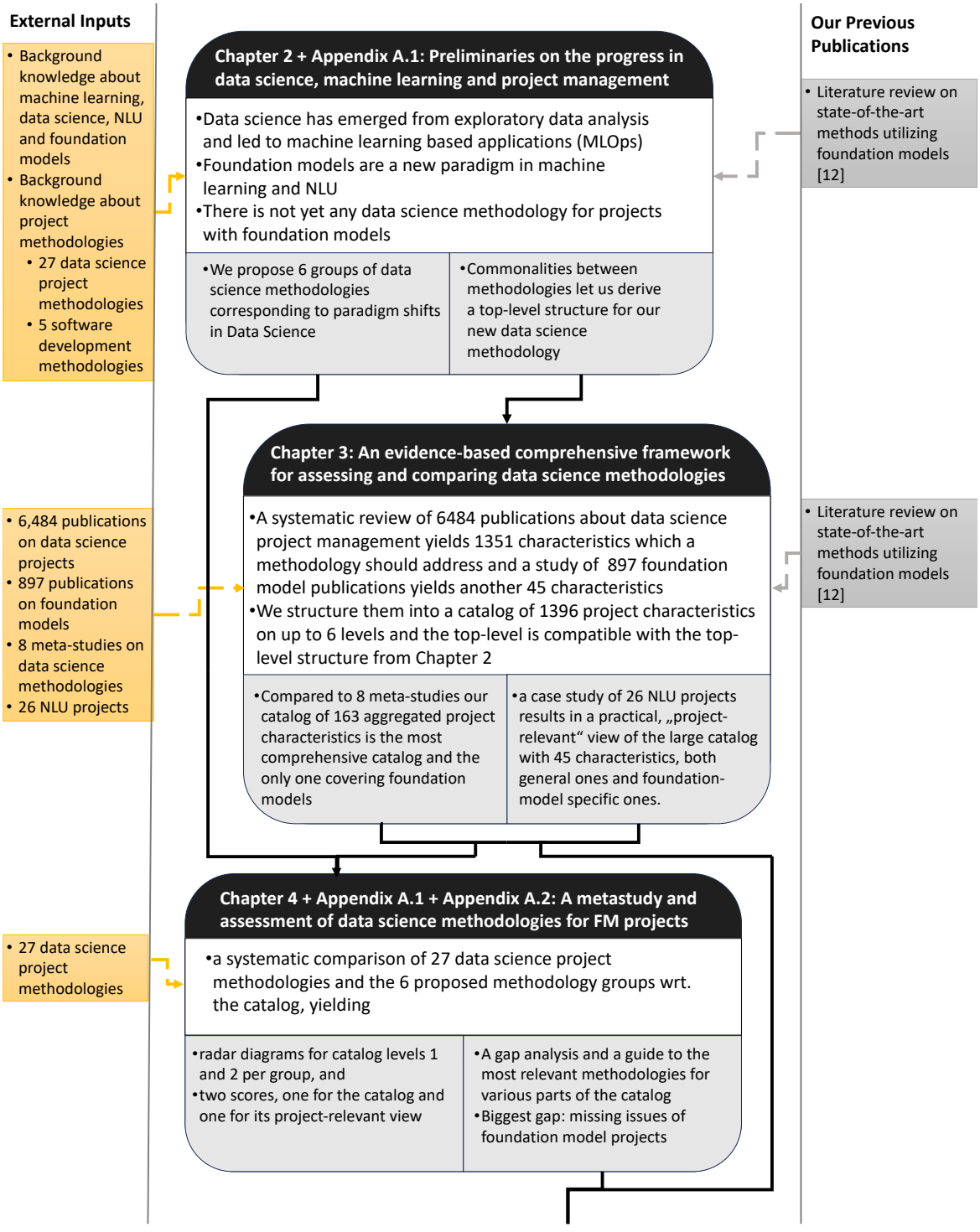

Figure 1.1: Overview of Chapters 1-4, our contributions, inputs and previous publications.

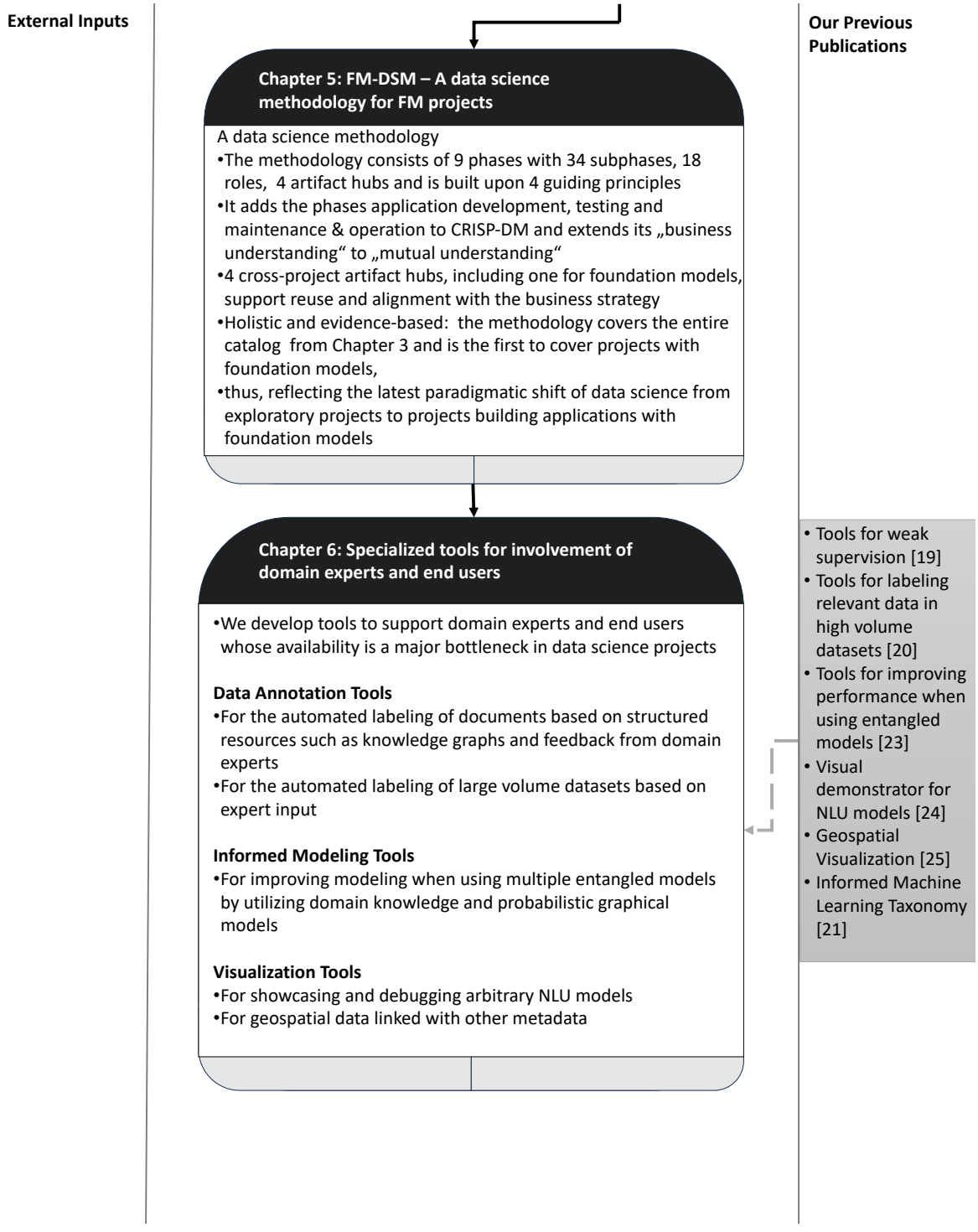

Figure 1.2: Overview of Chapters 5 and 6.

paradigm shifts in data science they address. Each methodology undergoes an assessment using the characteristics outlined in our catalog. Given that our project-relevant catalog offers a practical, albeit biased perspective geared towards the NLU projects of a single team, evaluations are also conducted based on our complete catalog. Methodologies are rated by evaluating if they mention or address specific characteristics from our catalog. Additionally, any gaps or partial coverage in our catalog by existing methodologies are investigated. Finally, we also investigate whether the paradigm shifts in data science are visible based on the different characteristics that methodologies address.

In Chapter 5 we present the core result of this thesis. We derive our own methodology for foundation model projects, tailored towards the aforementioned characteristics. Our methodology is built around four guiding principles: encouraging holistic thinking, involving end-users and domain experts in every phase, embracing visualization in many subphases, and frequently delivering and verifying results and insights. The methodology consists of 9 phases and 34 subphases. It embeds each project in artifact management processes and motivates the use of data, knowledge and software hubs. For each phase, we list the key roles involved, the artifacts used and produced, as well as the tools and techniques to aid in automating the phase. The methodology combines best practices of former methodologies and extends them to fully cover our catalog of characteristics. By incorporating application development and maintenance, exhaustive artifact management and team processes, it is the only methodology to reflect the practical implications of foundation models and better reflects the necessity of embedding projects into overarching processes and business strategies.

The involvement of domain experts and end users in data science projects is a critical success factor. Indeed, domain experts and end users play an important role in every phase of the methodology. However, a common challenge is that their time is limited. In Chapter 6 we provide contributions in the form of specialized tools to optimize their involvement in projects, both by making it easier to integrate their knowledge in the solution and by making it easier for them to understand the project results. We present three approaches to incorporate expert knowledge in the annotation and modeling (sub-)phases. Two of these approaches tackle the efficient annotation of data while one addresses the issue of error propagation when using multiple models in an application. Further, we present two tools for the visualization of textual data and modeling results. The first tool aims at the geospatial visualization of topics in documents, the second tool, called the NLU showroom, serves as a demo platform for arbitrary NLU models, in particular foundation models.

1.3 Previously Published Work

Parts of the contributions in this thesis have been published in conference and workshop proceedings. In [19], we introduce a tool which utilizes weak supervision, explainability methods and a domain expert to create a weakly annotated dataset. My contributions in this publication were the supervision of the master's thesis this originated in, the idea to tackle relation extraction as a distant supervision problem, utilizing a knowledge graph, and the inclusion of a domain expert for reviewing the weakly annotated data and improving it iteratively. In [20] we propose a process for labeling relevant data in a high-volume dataset. We start by obtaining filter criteria such as relevant terms by domain experts, filter the dataset and then train topic models on the dataset. The domain experts can then explore the topic model results to further narrow the dataset down to its relevant parts. My contributions include the idea of the process, the implementation and the evaluation. In [21] we propose a so called "Informed Machine Learning Taxonomy", i.e., a taxonomy of enriching machine learning

with knowledge. My contribution is the research for the section about including knowledge from knowledge graphs and the writing of the section. In [22] we propose a framework for identifying fake news by putting them into context of articles from trusted sources. My contribution was the supervision of the project and the design of the solution architecture. In [23] we propose a method to optimize the performance of entangled machine learning models, i.e., machine learning models which are dependent on the results of other machine learning models. My contributions were the training of the base machine learning models and the idea to allow both models to correct each other's outputs. In [24] we introduce the NLU Showroom, a visual demonstrator for NLU models. My contributions were the idea, the concept design and the supervision of the project. In [25] we introduce a geospatial visualization tool, to showcase textual data with geospatial and other metadata. My contributions were the idea, the concept and architecture design and the training of the sentiment detection model used in the demonstrator. In [12] we present the results of a literature review on state-of-the-art methods utilizing foundation models. My contributions were literature search, writing parts of Chapter 2 and 3, most of Chapter 5 and parts of Chapter 7, as well as proofreading the whole document.

CHAPTER 2

Preliminaries on the Progress in Data Science, Machine Learning and Project Management

This chapter serves as an overview of the most relevant concepts and developments in the fields of data science, machine learning (ML), foundation models (FM) and data science project methodologies. Data science methodologies have always been adapted to changes in paradigms and technology. Hence, it is crucial to explore the evolution of data science in order to understand the history of data science methodologies. To grasp the impact of foundation models, we focus on a subfield of data science named natural language understanding. Most of the research on foundation models originated in and focused on NLU. We identify four developments which are crucial for the creation of a new project methodology tailored towards foundation models: (1) the evolution of data science, (2) the concept of self-supervised learning, (3) the introduction of the transformer architecture, (4) the transfer learning capabilities of pre-trained models.

First, in Section 2.1 we describe the evolution of the data analysis field in general. From earlier days, when the focus was on explorative analysis of data in databases, with the goal of discovering valuable patterns and insights, to modern times when the focus shifted towards projects which yield applications such as predictive services. Looking at data science methodologies published over the course of the last three decades, we can see that the focus of data analysis projects and their requirements shifted multiple times. In the 1990s the predominant term for data analysis projects was knowledge discovery. This referred to the explorative discovery of knowledge in databases. Data mining was merely seen as a step in the knowledge discovery process. In the early 2000s the field matured, and the focus shifted to larger scale projects, with project teams and different stakeholders. The term data mining was adopted in a more general sense to describe all activities in data analysis projects. With the availability of more and more data, the term big data was introduced, which does not only refer to the volume of data that needs to be analyzed, but also to factors such as the rate in which the data is processed and the tools built to cope with the requirements of big data. More recently, the term data science has been used with a focus on ”science” and hence referring to the machine learning technology used in these projects. We can also observe a shift from more explorative projects towards projects which produce predictive solutions with the goal to deploy them. To this end, machine learning operations (MLOps) was introduced to complement data science with regards to deployment, monitoring and operation of machine learning based solutions.

The second development, described in Subsection 2.2.1, is the introduction of self-supervised

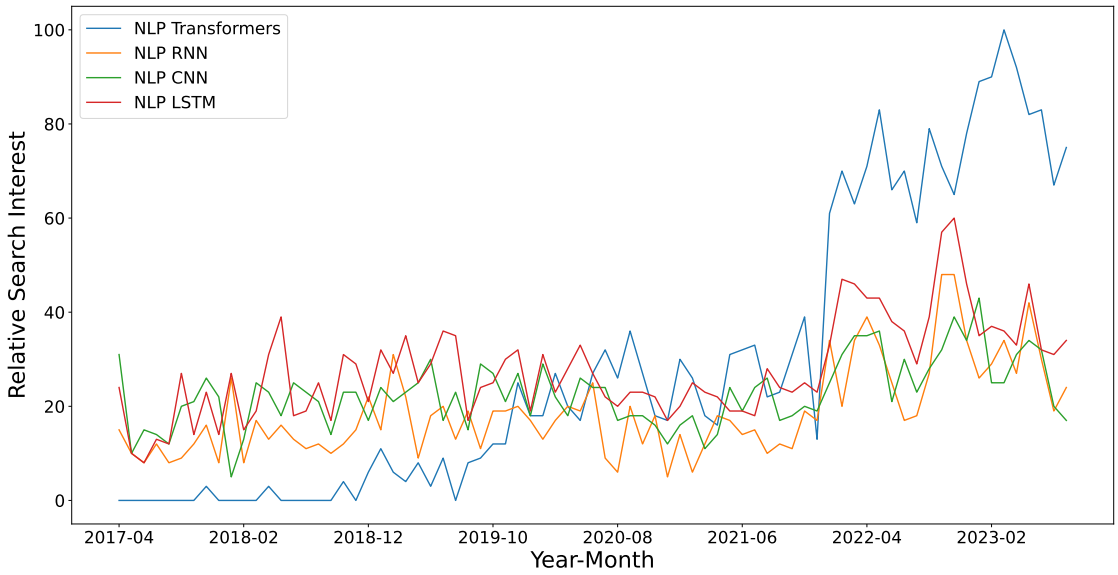

Figure 2.1: Google Trends results for the search queries NLP concatenated with Transformers, RNN, CNN or LSTM from April 2017 until August 2023. The y-axis represents the search interest, normalized by the highest value. It is evident, that in the last year, the interest for Transformers surpassed that of all other architectures.

learning as a form of machine learning, which allows machine learning models to train on large amounts of data, without the need of human annotation. A typical way of achieving this is by automatically masking parts of the training data and letting the model reconstruct the masked parts. Not needing human annotators gives rise to the possibility of training models on large amounts of training data. Self-supervised learning is the backbone of foundation models. These models are often initially trained in a self-supervised way, to capture the meaning of data, and then adapted to a specific task, either via a process called fine-tuning or via prompting.

Third, the introduction of the transformer neural network architecture has marked a significant shift in machine learning. Until the publication of the transformer architecture, recurrent neural networks were the natural choice for most NLU tasks and convolutional neural networks (CNNs) for computer vision. As the transformer architecture originated in NLU, we will focus on the development with respect to NLU. The transformer architecture superseded recurrent neural networks (RNN) by eliminating many of their computational problems. It allows for better parallelization, deeper networks and better capturing of long-term dependencies of inputs. We briefly describe the evolution from RNNs to transformers in Subsection 2.2.2. The importance and relevance of the transformer becomes evident when looking at the citation count of the publication that introduced it. As of writing, the transformer paper [26] has surpassed the original recurrent neural network paper [27] and is about to pass the citation count of the most popular recurrent network architecture, the long short-term memory (LSTM) network [28]. Since its publication the google searches for ”NLP transformers” have also surpassed the searches for ”NLP RNN”, ”NLP CNN”, and ”LSTM” as can be seen in Figure 2.1.

Fourth, to understand the transfer learning capabilities of foundation models and their impact on the machine learning process, we describe the evolution of text representations in NLU in Section 2.3. We start with classical machine learning with bag-of-words representations and end with large language models (LLMs). NLU as a subfield of data science focuses on data with a textual modality.

The modality of data has always been a major differentiation factor in machine learning, distinguishing methods for image, audio, text, video and structured data. Historically, each modality benefited from different data representation methods. Today, the transformer architecture is the predominant neural network architecture across most types of media. Transformers, together with self-supervised learning, gave rise to foundation models. With their transfer learning capabilities, foundation models present a stark contrast to early text representations. The latter required heavy preprocessing and task- and dataset-specific machine learning models to achieve good performance. The most prominent type of foundation models are large language models. Today, large language models are the state-of-the-art solution for every NLU task. As of writing, the popular hugging-face repository contains over 240,000 pre-trained language models, available for download and reuse.

Additionally, we provide an overview of project management concepts and focus on project management methodologies in software development and data science as data science is closely related to software development in Section 2.5. Many data science project methodologies borrow concepts from software development, so we start with an overview of software development methodologies. We then present CRISP-DM, which is to date the most important data science methodology.

2.1 Data Analysis

Over the course of the last decades, analyzing data has been coined under many different terms: data analysis, data analytics, knowledge discovery, data mining and most recently data science. In today's literature these terms are often used synonymously, however, when looking at literature about project methodologies we can observe that the terms were often introduced with changing project requirements, challenges and success factors. Hence, we provide a short overview of their differences and evolution. The order we choose is based on the appearance of the terms in data analysis project methodologies.

2.1.1 Knowledge Discovery

The term knowledge discovery was widely used in the 1990s to reference projects in which data was analyzed in databases [29, 30]. The methods used in these projects are at the intersection of research fields such as machine learning, pattern recognition, databases, statistics, AI, knowledge acquisition for expert systems and data visualization. Fayyad et al. describe knowledge discovery projects as data-centric and with data mining as a core activity [30]. Although other terms have become more popular for data analysis projects, knowledge discovery is still used today [31]. Methodologies for knowledge discovery describe rather exploratory tasks, i.e. the purpose of the projects is to discover insights in data and to present the information to a business stakeholder, so that they can make decisions based on the insights.

2.1.2 Data Mining

Witten et al. define data mining as "[t]he process of discovering patterns in data" [32]. They further add "[t]he process must be automatic or (more usually) semiautomatic. The patterns discovered must be meaningful in that they lead to some advantage, usually an economic one. The data is invariably present in substantial quantities" [32]. The authors argue that analyzing data with computers has not only emerged with data mining, but that the increase of available data has made it necessary to

create (semi-)automated solutions, because it is impossible for humans to process the data manually. This motivates the need for data mining tools. Many of the tools and techniques used for data mining stem from the field of machine learning. While the Knowledge Discovery in Databases methodology (KDD) referred to data mining as a subphase of knowledge discovery projects [30], CRISP-DM [2], which was released in 1999, referred to the whole process as data mining.

2.1.3 Big Data

In the early 2000s, a trend of exponentially increasing amounts of data was observed, which still holds today. Current estimates are that in 2022 around 97 zettabytes (97 billion terabytes) of data was processed worldwide [33]. With the increase of available data, the term ”Big Data” was introduced. However, the name does not only reflect the data quantity but also challenges and technical solutions which came along with it. Kaufman [34] provides an overview of the different meanings that big data can take, which we will now briefly summarize. First, in 2001, Laney released an article describing 3V's, namely volume, velocity and variety of data which need to be controlled [35]. Gartner specified those 3Vs as big data challenges [34]. Volume reflects the vast amounts of data, velocity the speed in which data needs to be processed and variety the heterogeneity of the incoming data. Later, Schroeck et al. added veracity as fourth V, which describes the uncertainty about the quality of data [36]. Further definitions with five [37], six [38] or even seven Vs exist [39]. A standardization of the definition was provided by the NIST Big Data Public Working Group: “Big Data consists of extensive datasets primarily in the characteristics of volume, variety, velocity, and/or variability that require a scalable architecture for efficient storage, manipulation, and analysis” [40]. Aside of these definitions, which are of technical nature, there are definitions which emphasize the paradigm shift in the field towards decision making by algorithms [41].

The challenges of big data required novel tools and techniques, tailored towards distributed computing, as single computers were often not sufficiently powerful to handle the storage capacity and processing power. Apache Spark [42], Apache Kafka [43] or Apache Hadoop [44] are only few examples of such tools.

2.1.4 Data Science

Data science is an interdisciplinary field, combining elements of computer science, mathematics, and business. Often the goal of data science is to extract knowledge from data to support business decisions. Dhar [45] argues that there are multiple factors which distinguish data science from prior forms of data analysis. One of the distinguishing factors is the emergence of large amounts of unstructured data such as text, images and video. He further argues that there is an emphasis on the ”science” part in the name and especially underlines the importance of predictive power of machine learning models, which takes on a large role in data science. Provost and Fawcett [46] describe data science as a “set of fundamental principles that support and guide the principled extraction of information and knowledge from data via technologies that incorporate these principles”. They add that data science connects big data with data-driven decision making.

Recently, it has become more important to deploy the solutions of data science projects in productive system. Projects often have the goal to create predictive solutions, i.e. train machine learning models to make predictions on data that is unseen during the project, e.g. to automate processes. The deployment of applications which embed machine learning models and their operation and maintenance only

recently became more relevant and is often coined under the term MLOps [47]. More recent project methodologies combine data science and MLOps approaches [48].

2.2 Machine Learning

Machine learning is one of the core technologies used in data science projects. It is used to detect patterns in data or to learn functions which help automating certain processes, given data.

Tom Mitchell defines machine learning algorithms as computer programs which learn to perform tasks, while their performance improves with more experience [49]. Shalev-Shwartz and Ben David add to that definition [50] that experience refers to the input of machine learning algorithms, which is training data. The output of machine learning algorithms are programs which perform tasks. Both sources describe machine learning as a field in the intersection of statistics, artificial intelligence and many other disciplines.

The input data is often referred to as input features x, the machine learning algorithm produces a function $f(x)$, often also called a machine learning model. The output of the model is y'. y is often reserved for a so-called ground-truth observation, i.e. the ideal result a machine learning algorithm should produce given a certain input. Assigning ground-truth observation to data is a process which is called annotation or labeling. This process is often manually performed by human annotators. Annotations vary depending on the machine learning task.

While in traditional programming, a programmer defines a program which transforms an input to a desired output, in machine learning the ”program” is a function that is learned based on input and output pairs, i.e. the program is the output of the learning process. Here, the data scientist chooses the machine learning algorithm and its parameters to produce the ”best” program, i.e. the program which produces the output closest to the ground-truth.

2.2.1 Types of Machine Learning

The literature distinguishes multiple forms of machine learning, which mainly differ in their learning objectives and their training data. We will focus on the three types of machine learning, most relevant to this thesis, which are supervised, unsupervised as well as self-supervised learning. Especially the latter plays an important role for foundation models and NLU. Key differences between the three types are whether the data contains labels or not and whether these labels are automatically created or obtained via manual annotation. This is visualized with examples in Figure 2.3.

Supervised Learning

In supervised learning, the training data does not only contain the data itself but also the desired solution a machine learning algorithm should produce when performing a certain task with the data, i.e., the input consists of a data and label tuple (x, y). For example, when the task is classifying restaurant reviews into positive or negative ratings, the training data would not only encompass the reviews but also their respective ratings. These solutions are often called labels and assigning labels to a dataset is referred to as annotation.

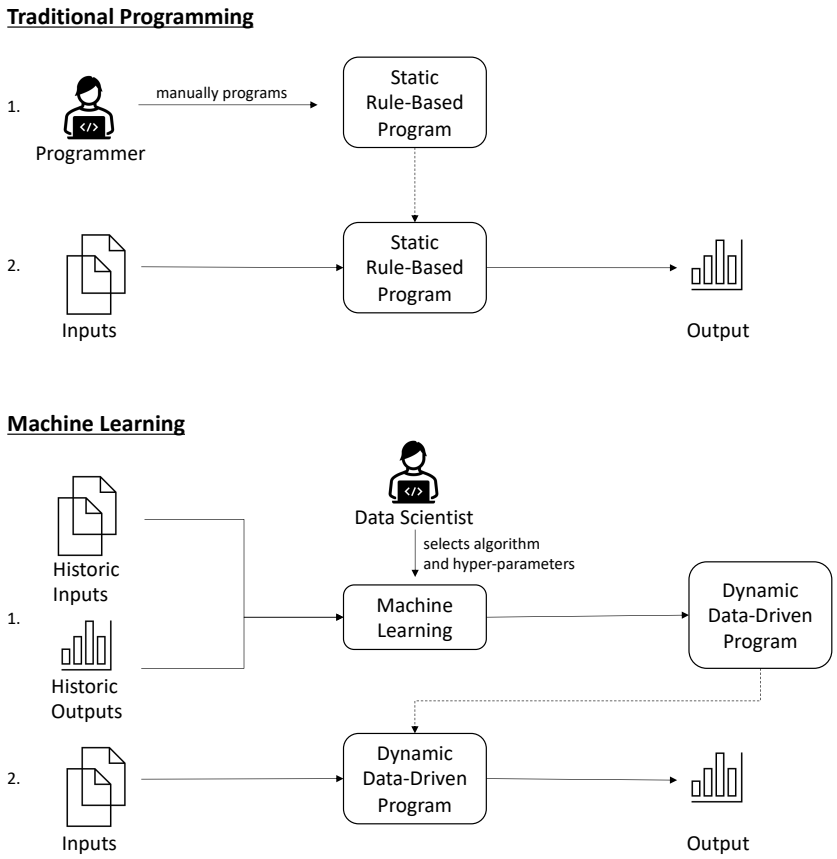

Figure 2.2: Traditional Programming versus Machine Learning

Unsupervised Learning

Unsupervised learning is characterized by the absence of labels in the training data. A typical task in unsupervised learning is the grouping or clustering of similar inputs. Let us assume we have a collection of millions of texts and we want to discover commonalities among them. Clustering methods would now aim to group these texts, e.g. based on the similarity in meaning that they share.

Self-supervised Learning

One form of machine learning, which is particularly important for foundation models, is self-supervised learning. It is a blend of unsupervised and supervised learning. This form of learning generally does not require any manual annotations, but rather automatically corrupts training data, e.g. by masking words or image patches, and lets the machine learning algorithms learn to reconstruct the original state of the training data. In this way arbitrarily many training samples can be created without the

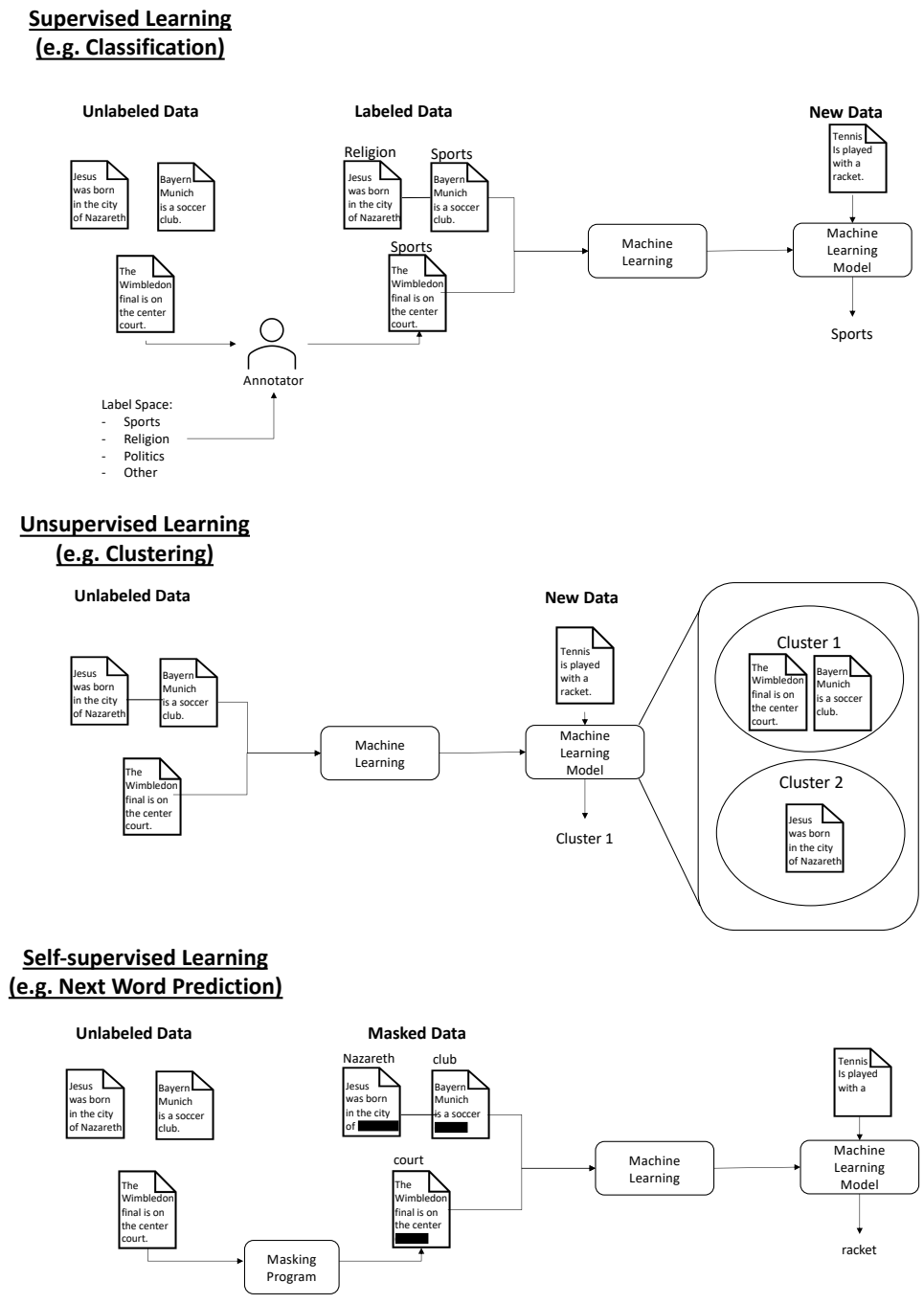

Figure 2.3: In supervised learning, human annotators assign labels to data, a machine learning algorithm then learns to assign labels to unseen data. In unsupervised learning data is provided without labels and the machine learning algorithms use similarities to discover patterns and cluster data. In self-supervised learning, labels are created automatically by masking parts of the data, machine learning models then learn to reconstruct the masked parts.

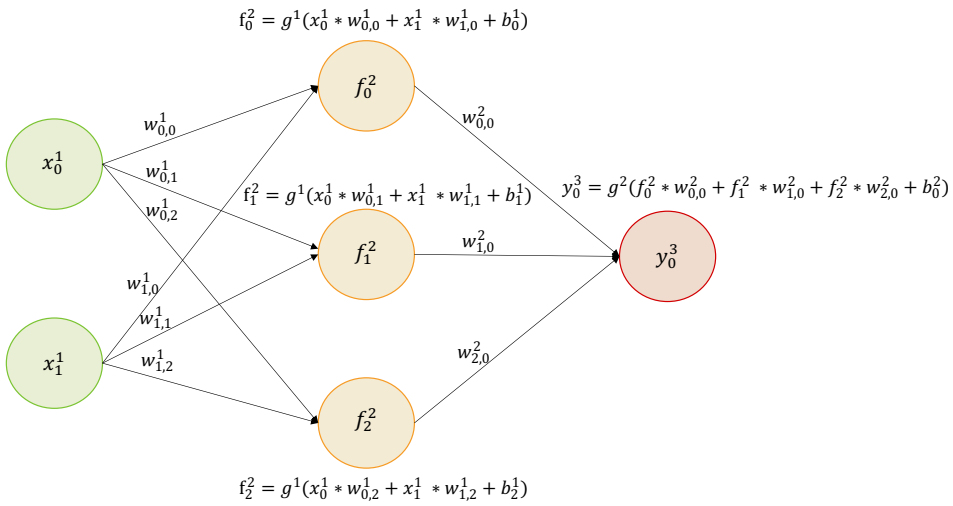

Figure 2.4: A simple feed forward network with two input units (x_0^1, x_1^1), three hidden units (f_0^2, f_1^2, f_2^2) and one output unit y_0^3. Between layers all units are connected with weights w. Hidden and output units are calculated as weighted sums over their inputs, with unit specific biases b and layer-specific activation functions g.

intervention of a human. Self-supervised learning tasks often only serve as auxiliary learning tasks for machine learning models. They provide the models with a general understanding of the underlying data, such as knowledge about syntax and semantics. Models which are trained via self-supervised learning, are often adapted to specific tasks, e.g. by training them on task-specific data. Generative models, such as models for text completion, do not require task-specific fine-tuning, as they can be adapted via prompting, i.e. giving the model natural language instructions to perform a task.

2.2.2 Neural Networks

Today, neural networks are the most commonly used type of machine learning models when training on unstructured data such as text [51]. They are loosely based on concepts from neuroscience, such as neurons. One of the most general neural network types is the feedforward network [52].

Feedforward networks got their name for two reasons. Feedforward because information flows from the input to the output, without back-connections. Networks, because they represent networks of functions which are chained together. The first function is applied to the input, the second function to the output of the first function and so on. Each such function is also referred to as layer and if the amount of layers increases, these networks are often called deep neural networks.

A special form of the feedforward network is the fully-connected network. Each layer in a neural network contains units. The network consists of units x_i^0, f_i^l and y_i^d where i is the number of the unit in the layer, l the layer of the network and d its depth. x specifies input units, f specifies units within the network, often named hidden units and y specifies output units. Figure 2.4 illustrates a simple example with one input, one hidden and one output layer. In the case of fully connected layers, all units in one layer are connected to all units in the successive layer via weighted edges w. The weights $w_{i,j}^l$ are learnable parameters, which the neural network adjusts during training via the backpropagation algorithm [53], based on a loss function. Figure 2.4 depicts the calculation of the values of hidden and output units. The calculations are always weighted sums of the incoming units

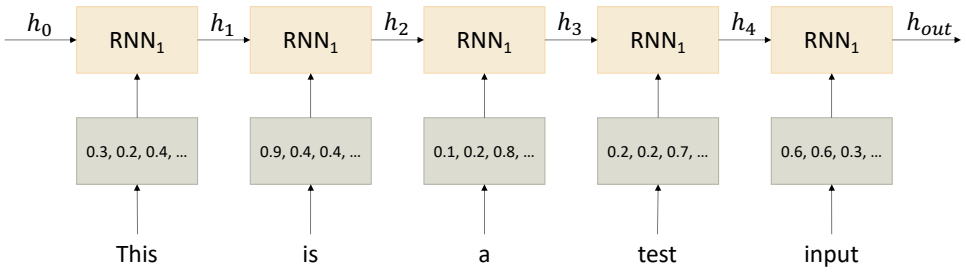

Figure 2.5: The RNN processes input sequentially. At each time step, the RNN is fed with an input vector x_t and a hidden vector h_{t-1} and produces a new hidden vector h_t. Ideally this hidden vector captures the content of the input processed so far.

with an added unit-specific bias b. The sum is passed through a function g, also called the activation function. There are various activation functions and they often differ in each layer. To enable the network to approximate arbitrary functions often non-linear activation functions are used.

Recurrent Neural Networks

Feedforward networks assume the same input size across samples. Especially in the text domain, this is an assumption which does not hold. Recurrent neural networks aim to solve this issue by processing inputs sequentially. To achieve this, recurrent neural networks make use of a hidden state h which serves as a sort of short-term memory, preserving information about already processed inputs. Given an input sequence $x = (x_1, x_2, x_3, ..., x_T)$ of length T, the RNN processes each element of the input sequence sequentially, together with the respective hidden vector at the current time t. Note, that x_1 does not refer to a single input unit, but a sample of the dataset, i.e. x_1 is typically a vector passed to the network. The RNN function for an input x_t, with a hidden vector h_t is then $RNN(h_t, x_t)$ and produces the hidden vector for the next time step h_{t+1}. This is depicted in Figure 2.5 with a toy example.

A typical task for recurrent neural networks is language modeling. The standard problem of language modeling is predicting the next word in a sequence, an application commonly used in smartphones. At each time step a logistic classifier can be applied to predict the probability of the next word in a sequence. Given a logistic classifier with weights H and bias b, the prediction function is

$$p(V_{t+1|v_1,...,v_t}) = softmax(H * h_{t+1} + b) \tag{2.1}$$

When optimizing the RNN parameters via stochastic gradient descent, the vanishing and exploding gradient phenomenon can be observed. This phenomenon causes the RNN to not be able to carry information about distant elements in a sequence. RNN variations such as the LSTM [28] have been proposed to mitigate this effect. Often variations of these architectures are used. The BiLSTM for example, processes input from left to right and right to left. The hidden vectors produced after processing each input can be regarded as contextualized representations of the input. In the case of BiLSTMs the representations from both directions are often concatenated to form a new input

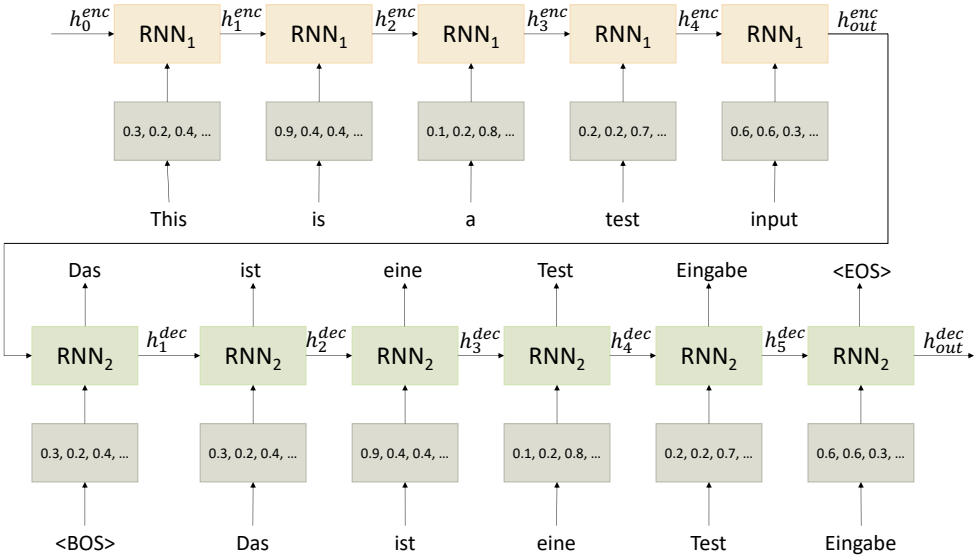

Figure 2.6: The encoder produces a hidden vector h_{out}^{enc} after processing all inputs, which ideally captures the meaning of the complete input sequence. The decoder has the task to take this hidden vector and use it to produce all outputs of the target sequence.

representation.

The Attention Mechanism

Solutions to solve translation problems often involve using two separate RNN networks. An encoder network, which encodes text in the input language, as well as a decoder network, which takes the output of the encoder network and decodes it into the target language. This is depicted in Figure 2.6.

A problem with this approach is that the hidden vector which is passed from layer to layer has to capture all of the information about the input as well as information about the already translated words. To enable translation of long sequences, the attention mechanism was proposed [54]. The attention mechanism essentially computes correlations between the current hidden vector in the decoder and all previous hidden vectors of the encoder. After normalizing these correlations, they can be regarded as weights which represent how important each hidden vector is for the current output. A weighted mean of these vectors is then used to improve the prediction of the next word in the decoder. A toy example is shown in Figure 2.7.

Transformer Networks

In 2017 Vaswani et al. introduced the transformer network [26]. The model was used for machine translation and consisted of attention-only blocks, i.e. there is no recursion. Similar to the aforementioned RNN architecture for translation, the transformer consists of an encoder and a decoder. The key component of the network are self-attention blocks as seen in Figure 2.8.

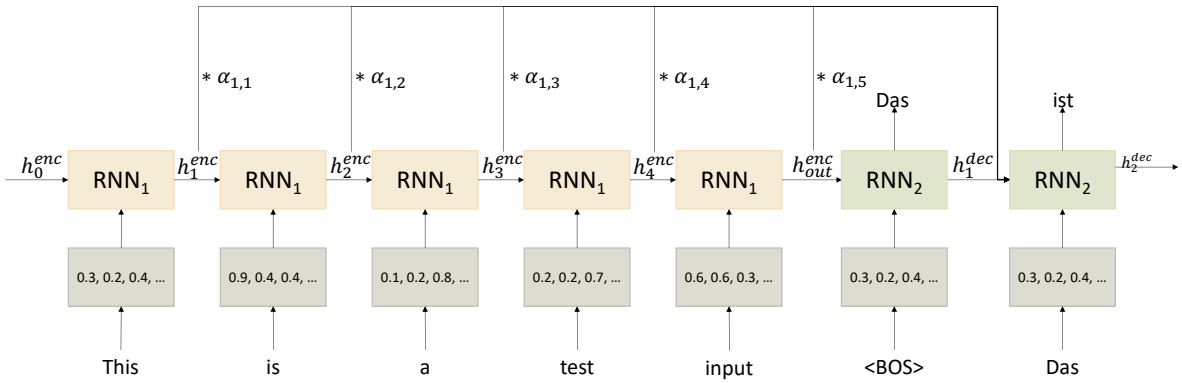

Figure 2.7: The attention mechanism allows the decoder to attend to all hidden vectors in the encoder at each time step. It mitigates the problem of the hidden vector being an information bottleneck.

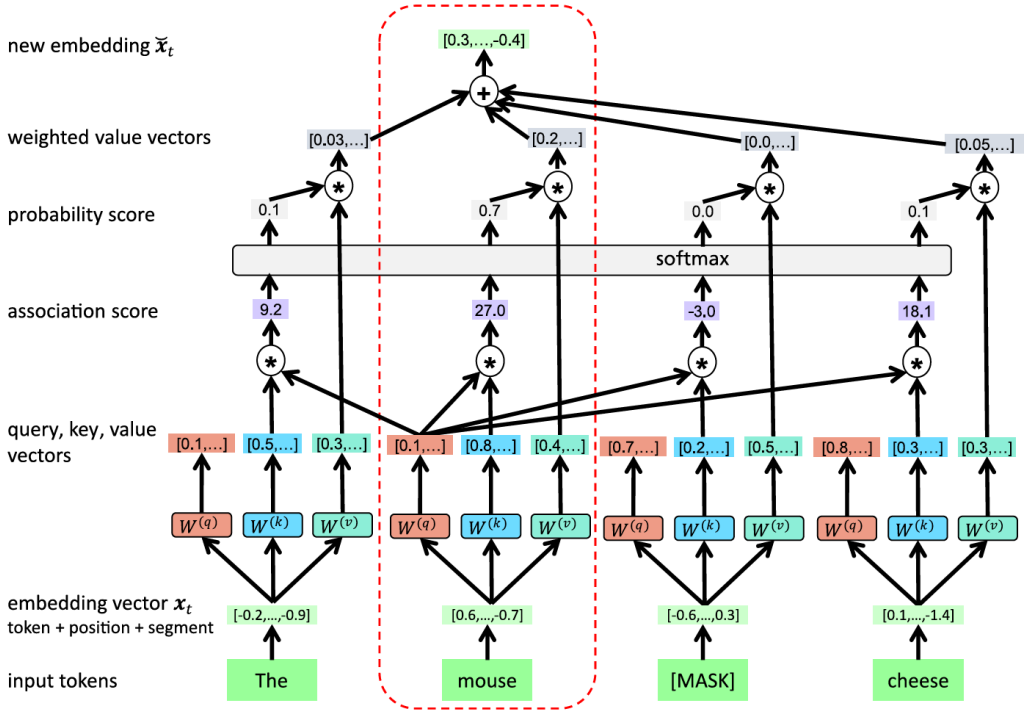

Figure 2.8: The calculations in a self-attention block [12].

The input of the transformer are token embeddings, position embeddings and segment embeddings. The token embeddings should reflect the semantical and syntactical information of the token, which is often a subword-element. The positional embeddings reflect the position of the token in the input sequence. This is necessary because recursion is omitted and with it all sequential information. Finally, the segment embedding specifies which tokens belong to the same segment, e.g. the same sentence. Typically, these embeddings are all learned from scratch during the training of the transformer network. The sum of these three embeddings yields a token representation t with dimensionality d_t.

This token representation is transformed to a query-vector q_t, key-vector k_t and value-vector v_t via linear mappings. These mappings are learnable parameters, represented with the matrices W_q, W_k and W_v. The purpose of the transformer network architecture is to enrich each token with information from its context. To enrich the first token in the sequence with contextual information, its query representation q_1 is compared to the key representations of all tokens in the sequence $k_1, ..., k_n$ via scalar products, i.e. $q_1 \times k_1$, $q_1 \times k_2$ and so on. The scalar products are then normalized with the factor $\frac{1}{\sqrt{d_k}}$ and finally the softmax over all normalized scalar products is calculated to obtain a weight $a_{1,t}$ for each token.

$$(a_{1,1}, ..., a_{1,n}) = softmax\left(\frac{q_1^\intercal k_1}{\sqrt{d_k}}, ..., \frac{q_1^\intercal k_n}{\sqrt{d_k}}\right) \tag{2.2}$$

The value representation of each token is now multiplied with its respective weight and the weighted sum of all tokens is used to create a new, contextualized token representation for the first token.

$$x_1^1 = a_{1,1} * v_1 + ... + a_{1,n} * v_n \tag{2.3}$$

Finally, each self-attention block also contains a feed-forward layer with rectified linear unit (ReLU) activation, to introduce a non-linearity in the model, and a linear feed-forward layer, projecting the new token representation to a desired embedding size. This process is performed for all tokens and repeated for all L layers, yielding L representations $x_t^1, ..., x_t^L$ per token. When multiple matrices K, Q and V are used per layer, it is called a multi-head self-attention block. Note, that the calculations in a self-attention block can be parallelized across all elements in the sequence. However, the memory demand is quadratic in the size of the input $O(n^2)$, i.e. the number of tokens in the input. In the encoder, multiple such encoder blocks are stacked over each other, with the goal that the last block produces useful representations for each token.

The task of the decoder is to produce a sequence, given an input sequence. In the case of machine translation, the decoder translates an input sequence to another language. The decoder consists of self-attention blocks as well, however they can only attend to words which the decoder has already produced. Further, the decoder consists of cross-attention blocks. Computationally these are the same as self-attention blocks, however the attention goes from the decoder towards all the tokens in the last layer of the encoder. The complete architecture is sketched in Figure 2.9.

Transformer networks have shown remarkable efficiency. While achieving state-of-the-art results in many translation tasks, the transformer required 50 times less computations than comparable solutions at the time [26]. In addition, the computations in each layer of the network can be parallelized, i.e. all words can be passed through a layer as a complete sequence, whereas recurrent networks need to process sequences word by word. Both, the omittance of a hidden state and allowing each word in a sequence to attend to all others, give the transformer the capability to better capture long-range

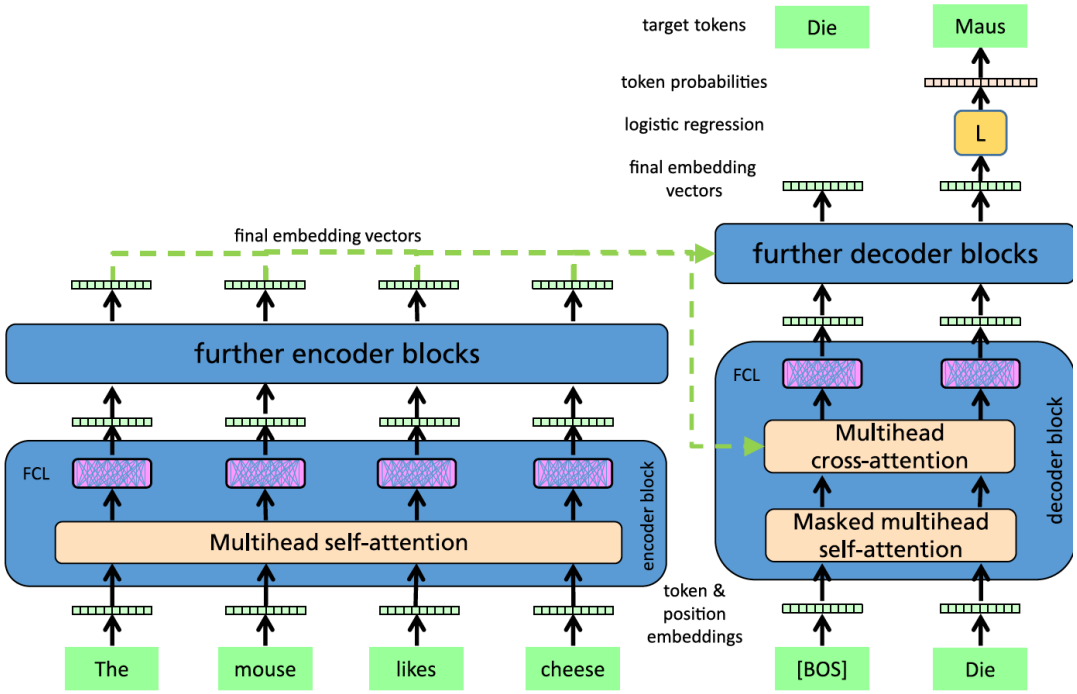

Figure 2.9: The transformer architecture with its encoder and decoder parts performing a translation task [12].

dependencies. However, with the aforementioned downside of a quadratic complexity in terms of memory.

2.3 Text Representations in Natural Language Processing and Understanding

Natural language understanding and natural language processing are subfields of data science. A key differentiation factor of NLP and NLU to other subfields of data science is that the input consists of natural language, i.e. speech or text. While natural language processing focuses on the processing of natural language inputs, natural language understanding focuses on deriving meaning from texts. Often methods of NLP are used to process text before NLU methods can be applied. For the sake of simplicity and because we are mainly interested in tasks which require natural language understanding, we will refer to both of them as NLU for the remainder of this thesis. Further, we limit the scope of NLU to textual data and omit speech.

While the same types of machine learning algorithms were used across modalities, feature representations were typically learned individually. In contrast, foundation models are able to learn meaningful representations across modalities. To highlight the benefit of foundation models, compared to traditional feature learning approaches, we will provide a short overview of the development of the NLU field with regards to text representations.

Ideally, textual features capture the context-dependent meaning of a word based on semantic,

syntactic and morphological information. A lot of the most relevant research in natural language understanding is not on how to create new machine learning models for certain predictions, but rather on how to best represent text. We will briefly show how text representations have evolved and give explanations about the downsides and benefits of the various representations. Note, that this is only a partial overview and does not aim to provide a complete survey of text representation models.

2.3.1 Bag-of-Words Representations

To make text processable by statistical methods and machine learning algorithms, we need to transform text into a numerical representation. One of the earliest numerical representations used for text is the bag-of-words (BoW) representation. Given a vocabulary V of size $|V|$, i.e. the set of all words in a document collection D, each document is assigned a vector d of dimensionality $|V|$. Each component of d corresponds to a distinct word in the vocabulary, i.e. V_i corresponds to the i-th word in V. Now the frequency of each word w in the document is counted and assigned to the respective component of the vector: $v_i = c_{d,w}$. An example can be seen in Figure 2.10.

Bag of Words

Document 1: He drove to the bank in his car.

Document 2: He drives his cabriolet to the river bank.

Vocabulary:

"He", "drove", "to", "the", "bank", "in", "his", "car", "drives", "cabriolet", "river".

	He	drove	to	the	bank	in	his	car	drives	cabrio let	river
Doc 1	1	1	1	1	1	1	1	1	0	0	0
Doc 2	1	0	1	1	1	0	1	0	1	1	1

Figure 2.10: A simple example for a BoW representation, where the document collection consists of two documents. First the vocabulary is formed as the set of all words in the documents. Then a vector with the word count per document of all words in the vocabulary is assigned to each document.

This form of representation was used for different use cases such as text classification or clustering of documents. To this end, similarity functions such as the cosine similarity can be used. However, the BoW representation poses some problems. Words without important meaning, so called stop words, such as "a" are shared across documents and could lead to a higher document similarity. This problem motivated another form of representation called "term-frequency inverse document-frequency"(tf-idf). In tf-idf, words are divided by a factor that reflects across how many documents a word occurs, the inverse document frequency.

$$tfidf(w,d) = tf(w,d) * idf(w) = tf(w,d) * \log\left[\frac{n}{df(w)} + 1\right] \tag{2.4}$$

Here tf is the term frequency, which is essentially the same as a standard BoW representation, n is the total number of documents in the document collection and $df(w)$ is the amount of documents in which the word w occurs in.

The intuition behind this form of representation is that words are considered more important if they appear frequently in a few documents but are rare overall. The tf-idf representation is often preferred over the BoW representation as it shows better performance in benchmarks and applications. However, it faces some similar issues as BoW. Since each word in these representations is assigned to a different component of the document vector, relationships among words such as hypernymy, synonymy and homonymy are disregarded.

Bag of Words

Document 1: He drove to the bank in his car.

Document 2: He drives his cabriolet to the river bank.

Vocabulary:

"He", "drove", "to", "the", "bank", "in", "his", "car", "drives", "cabriolet", "river".

	He	drove	to	the	bank	in	his	car	drives	cabriolet	river
Doc 1	1	1	1	1	1	1	1	1	0	0	0
Doc 2	1	0	1	1	1	0	1	0	1	1	1

Figure 2.11: An example of the limitations of BoW-like representations. Words with similar meanings like "drove" and "drives" and hypernyms like "car" for "cabriolet" lead to underestimated similarity, as they are represented by different components of the BoW vector. Homonyms like "bank" lead to an overestimated similarity of the documents.

This leads to wrong estimates of similarity, either overestimating (in the case of homonymy) or underestimating the similarity (synonymy or hypernymy) as can be seen in Figure 2.11. Another problem of the BoW-like representations is their sparse high-dimensional representations.

2.3.2 Topic Models

Topic models mitigate some of the problems with BoW representations. The most popular algorithm for topic modeling is the Latent Dirichlet Allocation (LDA) introduced in 2003 by Blei et al. [55]. The intuition behind topic models is that documents can be represented as mixtures of topics, where topics are mixtures of words. Some properties of the LDA algorithm are that every word belongs to

every topic with a certain (mostly small) probability and every document can contain multiple topics. In vanilla LDA, topics are not named and are typically represented by their most probable words, either the top-k words or words which pass a probability threshold.

Topic models have different uses. They can be used to represent documents for tasks such as classification, calculating semantic similarity of texts or exploratory analysis of large document collections. In the LDA algorithm, the number of topics has to be specified a priori as a hyperparameter of the model. While large numbers of topics (e.g. 500-1000) can benefit classifiers, they are not suitable for exploratory analysis as the number of topics is no longer interpretable for humans.

Since words can belong to multiple topics, homonyms are likely to appear in multiple topics with higher probability and together with different words, enabling disambiguation. Synonyms and hypernyms are more likely to appear in similar topics together. Hence, representing documents as distribution of topics can lead to better semantic similarities than BoW representations.

2.3.3 Distributed Word and Document Embeddings

In 2013, Mikolov et al. presented the Word2Vec algorithm [56]. In contrast to the aforementioned representations, Word2Vec aimed at learning vector representations for words. The idea of Word2Vec is that "a word is characterized by the company it keeps", which traces back to the idea of distributional semantics in the 1950s by Zelig Harris [57] and John Firth [58]. More precisely, the idea is that the meaning of a word can be understood by analyzing with which words it cooccurs in a certain neighborhood. The neighborhood is specified in a distance d of words to the left and right of a word. Word2Vec assigns two vectors to a word, an input and an output representation, which are merged after training, e.g. via averaging or summation. There are two variants of Word2Vec, the Continuous Bag-of-Words (CBOW) representation and the Skip-Gram (SG) representation. In the case of CBOW, the algorithm tries to predict a central word, based on its context words. The input vectors of the context words in the d neighborhood of a central word c are averaged. The resulting averaged vector a is than compared to the central word and n randomly sampled negative words. The comparison is performed via a softmax function over the dot product of a and the output representation of c and the dot products of c and the output representation of the negative samples. The loss function is designed such that the softmax of the actual central word should be maximized and the softmax of the negative samples should be minimized. The respective input and output vectors are then optimized via stochastic gradient descent. SG modifies the training task by using c to predict the $2d$ words in its neighborhood. During training e passes or epochs are made through the document corpus. In this process, every word serves as the central word for each context it appears in e times. Subsequently, the input and output vectors of each word are averaged to form a new representation for that word.

It is important to note that Word2Vec represents each word as one vector, which is essentially a representation averaged over all contexts that the word appears in. Word2Vec vectors showed some interesting properties. When comparing the neighborhoods of the word vectors, one can find that they form semantic clusters. Further, the authors discovered0 that simple arithmetic over the vector space is possible and allowed for the calculation of analogies such as $v_{Berlin} - v_{Germany} + v_{England} = v_{London}$.

Most interestingly, Word2Vec popularized the paradigm of self-supervised training of word representations over millions of unlabeled documents and then feeding the pre-trained vectors into neural networks to fine-tune them on tasks such as document classification [59] or named entity recognition [60]. This new way of training led to faster convergence and better prediction accuracy on smaller datasets. Note, that the pre-training only needs to occur once, and the vectors can afterwards be

fed into arbitrary other neural networks or machine learning models to fine-tune them on task-specific labeled datasets. More so, this led to a trend of sharing pre-trained vectors for different languages, reducing the modeling efforts to choosing the vectors for the right language, creating a suitable task-specific neural network, feeding the vectors into the first layer and then fine-tuning on tasks without further pre-training [61].

Later modifications of Word2Vec included the ability to generalize to words unseen in training or embeddings by including morphological information [62] and the representation of whole documents by introducing document or paragraph vectors [63]. Word2Vec and the related approaches represented words with a single vector, which again led to problems with homonymy. Some approaches tackled this problem by learning multiple vectors for a word based on word senses obtained from structured resources such as WordNet [64]. However, these approaches solved the problem only partially.

2.3.4 Contextualized Word Embeddings

In 2018, Peters et al. popularized deep contextualized word representations with the "Embeddings from Language Modeling" (ELMo) approach [65]. In contrast to distributed embeddings, contextualized embeddings assign vectors to words based on the current context they are in. This leads to an individual word representation per context, rather than single distributed representations per word, and mitigates the problem of resolving homonyms. ELMo is a deep neural network which learns to represent words via language modeling. It consists of k stacked BiLSTM layers. On the lowest layer, embedding vectors (often character-based) are fed into ELMo. The first BiLSTM then processes these vectors from left-to-right and right-to-left. The hidden vectors after (respectively before) every word are then fed into the next BiLSTM layer. In the final layer, the model tries to predict the next (previous) word of a sentence, a task referred to as language modeling. Just like Word2Vec, this way of training requires no annotations, thus it can be referred to as self-supervised. This self-supervised task can be performed on arbitrary amounts of text and leads to a pre-trained network. The hidden vectors that ELMo generates in each layer are concatenated for both directions of the BiLSTM and can be regarded as contextualized word representations of the words in the input sequence. After finishing pre-training, ELMo embeddings can be used to perform arbitrary NLU tasks. To that end, the representations of each layer are combined as a weighted average and then used to solve downstream tasks such as classification or named entity recognition. The weight assigned to each of the representation layers is a learnable parameter. When released, ELMo showed remarkable transfer learning capabilities. The same pre-trained ELMo network could be fine-tuned to multiple NLU tasks, achieving state-of-the-art (SotA) performance in many of them, beating specialized architectures.

However, ELMo was soon surpassed by BERT, which was released later in 2018 by Devlin et al. [66]. BERT is based on the encoder of the transformer architecture and proved to be even more capable and scalable than ELMo. The original BERT paper introduced two different versions, $BERT_{small}$, which consists of 12 layers and $BERT_{large}$ with 24 layers. Both were pre-trained on a corpus of 3.3 billion words. BERT uses a vocabulary based on wordpieces which is built by starting with single characters and counting their cooccurrences. The most frequent characters are grouped and then their concatenation is added to the vocabulary. This is repeated until a desired vocabulary size is reached (e.g. 64.000 tokens). Words in the input are split into the largest word pieces in the vocabulary. This way of vocabulary building eliminates the need of specific word splitting methods in preprocessing, as each word can be put together by substrings of the vocabulary. The words in the vocabulary are represented by adding a token embedding, a positional embedding and a segment embedding.

The final token representations are then fed into the transformer encoder. BERT follows a similar pre-training and fine-tuning scheme as ELMo. However, it introduces two different pre-training tasks. In masked language modeling (MLM), tokens in the input of a sequence are replaced with a [MASK] token. The embeddings in the uppermost layer of BERT are then used to predict the masked words. Additionally, for the second pre-training task, two additional tokens are introduced: the [CLS] token, which marks the start of an input sequence, and the [SEP] token, which marks a sentence break. In the second pre-training task BERT uses the embedding of the [CLS] token to predict whether two sentences, split by [SEP], follow each other semantically or are randomly chosen. After pre-training, BERT can be fine-tuned for various NLU tasks by using it as the lower layers in different neural networks. For example, in classification tasks, a linear layer with softmax activation can be added on top of the [CLS] token. Either the entire BERT network or just the final layer can be fine-tuned using a labeled dataset. BERT achieved SotA performance in various NLU benchmarks.

Another transformer-based model which uses pre-training is the generative pre-trained transformer (GPT) [67]. While BERT corresponds to the encoder of the transformer architecture, GPT corresponds to its decoder, without the cross-attention to the encoder. GPT is a language model, which processes texts from left-to-right. In contrast to BERT, GPT does not predict masked words but the next word of a sequence. This again represents a self-supervised task, enabling pre-training on arbitrary amounts of unlabeled texts. GPT was shown to achieve state-of-the-art performance in language modeling tasks. Due to the attention-mechanism, it achieved especially good performance on generating long texts. While older models often showed topic drift, essentially forgetting earlier contexts, GPT referenced contexts over wider spans.

BERT and GPT marked the start of a remarkable development in the field of NLU. Various variants of them have been developed, featuring different pre-training tasks, attention mechanisms, model sizes, and training dataset sizes. [12]. It has been shown that increasing the model size and the size of the pre-training corpus yields better performance across tasks.

2.4 From Task-Individual Models to Transfer Learning - A New Paradigm in ML

The way NLU models are trained has evolved together with the text representations. Using machine learning methods such as support vector machines [68] with BoW-representations requires a lot of feature engineering. These representations carry no knowledge about language. In order to reduce the number of features to a subset of relevant features, many preprocessing methods have to be applied. Tokenization and sentence splitting are used to find word or sentence boundaries. Lemmatization or stemming normalize different word forms to a basic form. NGram or Phrase detection methods group words which either statistically co-occur very often and/or form meaningful phrases such as ”New York City”. In addition to the feature engineering, machine learning models are tailored exactly towards the preprocessed dataset. Topic models reduced the complexity slightly by representing texts as topic distributions. However, the creation of useful topics is also reliant on the preprocessing of the texts they were learned on.

Distributed embeddings mitigate the problem of repeated feature engineering and selection. Text representations by models such as Word2Vec are trained on large amounts of texts in a self-supervised manner and then used across projects. This enables good results with less training examples on downstream tasks because the embeddings already carry a lot of knowledge about syntax and semantics

[56]. Since the feature representations can be used as input across tasks, only the final machine learning model, i.e. often the architecture of a neural network, was constructed specifically for the task. Still, distributed embeddings required preprocessing methods such as sentence splitting, tokenization and NGram detection.

Finally, contextualized embeddings are obtained by deep neural networks which encode text and create vector representations based on the individual context a word appears in. Approaches like BERT [66] and ELMo [65] have shown that only minor alterations of such models are necessary to perform certain NLU tasks. Rather than sharing embeddings across projects, the whole network architecture is now reused across projects and only fine-tuned on a small amount of task-specific training data. These models are often referred to as language models and they have proven to beat the state-of-the-art across NLU tasks.

2.4.1 Foundation Models

Brown et al. have shown that increasing the parameter size and the amount of training texts during pre-training of models such as GPT is strongly correlated with the performance of language models in downstream tasks [69]. Language models with very large parameter sizes are referred to as large language models (LLMs). More generally, models of arbitrary modalities, even multi-modal ones, which are trained in a self-supervised manner, are often called foundation models. This name implies two properties, the models are built upon the same underlying architecture, independent of the data modality, and they show transfer learning capabilities.

Generative large language models show an intriguing property. Rather than having to fine-tune them on specific tasks, they can be instructed via natural language to perform tasks such as programming, question answering and classification [69]. This way of adaptation is called prompting and requires no specific retraining, hence no alteration of the model parameters. A term used to describe the ability of models to perform tasks without being trained on them is zero-shot learning. Brown et al. further show that models benefit from example solutions in their prompts, which is referred to as in-context learning [69]. For certain tasks, foundation models have been proven to be on par or even better when prompted, than specifically fine-tuned smaller models [70]. However, increasing the parameter sizes of large models comes with a price. Training these models often requires hundreds or even thousands of GPUs for months. The largest of these models cannot even fit onto a single GPU for inference. Simply scaling the model size is not sufficient for getting the best performing models. Studies have shown that there are optimal ratios of dataset, model sizes and compute, i.e., training models longer and on more data is also beneficial [71]. This effect is so strong that it has led to 70 billion parameter models outperforming 530 billion parameter models across tasks. However, using generative foundation models and prompting leads to novel challenges in solution design. Since these models solve tasks by providing textual answers, one has to make sure that these answers are correct, non-toxic and come in the right format, e.g., when structured answers are required. Controlling the answer generation, such that it can be used to automate processes, is still a topic of research. Recently, variants of such models have been specifically tuned to follow human instructions, making it easier to adapt these models to arbitrary tasks via prompting with natural language instructions [72].

Foundation models have led to a new paradigm in machine learning. Models like BERT and GPT are pre-trained on very large amounts of data with general text understanding tasks, such as predicting masked or next words in a sentence. These self-supervised tasks require only automated alterations of texts and can hence be performed without human annotations and on masses of texts.

Adapting these models to a task is as easy as prompting them to perform the task or adding only an additional task-specific layer or adapter and fine-tune them on a small amount of task-specific annotated data. This drastically reduces the modeling complexity and reduces the importance of steps such as neural architecture search, feature engineering, data annotation and even training. However, task- and domain-specific training can still be beneficial. It has been shown that language models improve their performance if they are pre-trained or fine-tuned with domain- and task-specific texts [73]. The pre-training and adaption scheme leads to a new machine leaning paradigm which requires rethinking the management of model and data artifacts, i.e., the storage, usage, versioning and creation of models and data. Models are now not only project-specific artifacts but can be reused across projects.

2.4.2 Impact on Natural Language Understanding

To understand the impact of foundation models on fields like NLU, we reviewed their impact on common benchmarks in prior work [12]. There is a wide variety of tasks which require natural language understanding. We analyzed 10 groups of tasks with respect to the state-of-the-art models used to solve them. We found that across all of these tasks and for most of the datasets transformer-based architectures yield the best performance, i.e., the current state-of-the-art in NLU is dominated by pre-trained language models. Further, the same large language models are often leaders in benchmarks across tasks, emphasizing their transfer learning capabilities.

1. **Text Classification**: The classification of text is a very common task in NLU. One or more labels should be automatically assigned to documents based on their content. Exemplary use cases for classification in which transformer-based models are the best models are the categorization of news articles [74], sentiment detection for customer reviews [74] or the assignment of medical diagnosis and treatment codes to patient documentation [75].

2. **Word Sense Disambiguation**: In word sense disambiguation algorithms are used to distinguish the different senses of words, i.e. words which are written in the same way but have different meanings depending on their context. Typical applications are word sense disambiguation or dialogue. Currently, a BERT variant performs best in multiple benchmarks [76].

3. **Named Entity Recognition**: Named Entity Recognition (NER) is the task of identifying named entities in documents, such as person names, organization names, medications, etc. It is commonly used to automatically fill databases with information from documents and is used for applications such as invoice processing [77].

4. **Relation Extraction**: Relation extraction is often used in addition to NER. It describes the task of understanding whether entities discovered in a document are also described to be in a relation. In the sentence ”Steve Jobs founded Apple”, the relation ”founder of” is described for the person entity ”Steve Jobs” and the company entity ”Apple”. Recently, large language models have shown great zero-shot capabilities in relation extraction and together with fine-tuned models, they beat several benchmarks [78].

5. **Document Retrieval**: Document retrieval describes the task of identifying relevant documents, given a search query. The query can be in the form of a short search string or in the form of a query document. One example of document retrieval is the automated comparison of news

articles for validation [22]. A popular BERT variant for document retrieval is SentenceBERT [79].

6. **Question Answering**: Question answering enables users to receive answers to their queries. Models used for question answering can access structured resources like knowledge graphs, perform reading comprehension tasks to extract answers from texts, or utilize retrieval techniques to find answers in text collections or on the web. Certain models also encode answers in their parameters. Question answering is used within assistants such as Siri or Google Assistant. Recently, question-answering approaches combine retrieval of documents which might contain the answer, with models which extract the answer out of the retrieved documents [80].

7. **Neural Machine Translation**: Neural machine translation is used to generate translations of texts from one language to others. Popular systems for machine translation are Google Translate and DeepL. Transformers have proven to beat the state-of-the-art in most benchmarks such as [81], with large language models showing state-of-the-art performance that outperforms production systems even in few-shot settings [82].

8. **Text Summarization**: Whereas information extraction methods such as NER and relation extraction provide a structured summary of key information in texts and documents, often textual summaries are required. Extractive summaries select the most important content from one or more documents and use them as a summary, whereas abstractive summarization are newly generated texts. Example usages for summarization methods are the automatic summary of sport events or news articles. A popular transformer model for summarizations is PEGASUS [83].

9. **Story Generation**: Story generation describes the ability of programs to write coherent, long texts and stories. This technology can be used to automate the creation of reports, describe football matches, or even create computer programs. A popular transformer-based model for text generation is InstructGPT, which is also the basis for ChatGPT [72]

10. **Dialog Systems**: The purpose of building dialog systems is to answer questions to users, entertain them or perform tasks for them. Dialog systems traditionally combined multiple of the aforementioned tasks, such as the classification of a user's intent, the recognition of a topic, the detection of a sentiment or the recognition of key entities in dialog processing pipelines. However, there are also end to end machine learning models without intermediate pipelines. Most recently, ChatGPT has shown the potential of transformers to act as smart assistants. ChatGPT is at the time of writing based on a variant of the GPT-4 model [13].

Public Model Hubs

As of July 2023, the Huggingface model hub [84] contains over 240,000 pre-trained and sometimes even fine-tuned models, available for download. Many of these publicly available models are state-of-the-art in common benchmarks, such as the aforementioned NLU tasks. Further, many of the models have licenses which allow their usage for commercial purposes. This provides companies with access to reusable models, i.e. in the best-case data scientists do not need to adapt models at all or just have to worry about fine-tuning them on task-specific datasets or engineering appropriate prompts.

2.5 Project Methodologies

The core contribution of this thesis is a project methodology which is tailored towards projects in which foundation models are used. To understand how such a project methodology should look like, we must first understand what a project methodology is, which purpose it serves and how it is related to other concepts in project management. We start by introducing relevant terminology from project management in Subsection 2.5.1. Then, we proceed with a definition of project methodologies in Subsection 2.5.2. Since, software development methodologies have an influence on data science methodologies, we introduce heavyweight and agile software development methodologies in Subsections 2.5.3 and 2.5.4. Then, we proceed to introduce data science project methodologies and propose a way of structuring them in Subsection 2.5.5. Lastly, we present CRISP-DM, the most popular data science methodology, in Subsection 2.5.6.

2.5.1 Project Management Terminology

We start with the definition of a project, project management and project life cycles. An understanding of these terms is necessary to understand the definition of project methodologies.

Project

According to the project management body of knowledge (PMBOK), a project is a "temporary endeavor undertaken to create a unique product, service, or result. [...] Projects are undertaken to fulfill objectives by producing deliverables" [85]. The authors of the PMBOK further define an objective as "an outcome toward which work is to be directed, a strategic position to be attained, a purpose to be achieved, a result to be obtained, a product to be produced, or a service to be performed" and deliverables as "any unique and verifiable product, result, or capability to perform a service that is required to be produced to complete a process, phase, or project" [85]. Further, the PMBOK elaborates that temporary endeavor means projects have a beginning and an end, where the end of project can be determined for multiple reasons, such as the achievement of the project objectives or the exhaustion of project funding.

According to the PMBOK, projects can be embedded in project programs and portfolios. Programs group multiple projects and even subprograms and portfolios group programs, subportfolios and projects to achieve strategic goals of an organization [85]. A more holistic view reveals that projects can be part of a bigger picture, i.e., their goals can contribute to greater goals of an organization and projects within an organization compete for resources [85].

The PMBOK also defines the success criteria of a project: "Success is measured by product and project quality, timeliness, budget, compliance, and degree of customer satisfaction"[85]. In addition several key components of projects are listed and described: Project life cycles, project phases, phase gates, project management processes, project management process groups and project management knowledge areas.

Project Management

Project management is used by organizations to assure the achievement of the aforementioned success criteria of projects. The PMBOK defines project management as "the application of knowledge, skills,

tools, and techniques to project activities to meet the project requirements" [85]. They further specify that project management involves the application and integration of project management processes.

Project Life Cycle

A project life cycle is a collection of project phases which are connected and have criteria such as milestones and deliverables which determine when to transition between them [3, 85, 86]. The life cycle covers the phases of a project from beginning to end [85]. It also specifies the order in which phases are traversed [86].

Phases are project activities which are related and have specific attributes such as names, durations, resource requirements, entry and exit criteria [85]. Phase gates specify how to proceed at the end of a phase and typically involve the comparison of the progress with respect to the business goals and can determine the start of the next phase, a modification of a phase, the remainder in a phase, the repetition of it or parts of it or even the end of a project [3, 85].

Series of such project management activities are also known as project management processes [85]. Given an input, these processes yield deliverables or outcomes using various tools and techniques. These outcomes can then serve as inputs for other processes [85]. Processes can be further grouped into process groups for initiation, planning, execution, monitoring and controlling and closing and are independent of phases [85].

There are various life cycle models such as predictive life cycles, where a lot of planning is done upfront, iterative life cycle, containing multiple cycles of the project phases incrementally adding to the desired outcome, incremental life cycles, where the deliverable is only finished after the final iteration, adaptive life cycles, also known as agile, where the scope is always determined before an iteration and hybrid life cycles, which are combinations of predictive and adaptive life cycles [85].

Project life cycles should be closely tied to project methodologies, i.e. project methodologies should be tailored towards the phases in the life cycle [3].

2.5.2 Definition of Project Methodologies

There are multiple definitions of project methodologies in literature. Some of them are very generic, while others are specific and more detailed. The PMBOK contains a general definition of methodologies: "A methodology is a system of practices, techniques, procedures, and rules used by those who work in a discipline" [85]. This definition is applicable to many domains and leaves large room for interpretation.

Charvat suggests multiple alternative definitions in his book "Project Management Methodologies: Selecting, Implementing, and Supporting Methodologies and Processes for Projects" [3]. He highlights that there is not one unanimously agreed on definition. He deems the following definition as the most appropriate: "A methodology is a set of guidelines or principles that can be tailored and applied to a specific situation. In a project environment, these guidelines might be a list of things to do. A methodology could also be a specific approach, templates, forms, and even checklists used over the project life cycle" [3]. Among his alternative definitions, there are two which complement this definition: 1) "A collection of methods, procedures, and standards that define a synthesis of engineering and management approaches designed to deliver a product, service, or solution" [3]. 2) "An integrated assembly of tasks, techniques, tools, roles and responsibilities, and milestones used for delivering the project" [3]. These definitions are all more specific than the definition of the PMBOK.

Marbán et al. provide yet another definition [86]. According to them, project methodologies describe how to perform tasks, while project processes define what to do. They further mention that methodologies should include tools and techniques that help to improve task performance.

Finally, Saltz et al. provide a definition which is tailored towards data science: "A system of practices techniques, procedures, and rules used to guide a temporary team-based endeavor that collects and analyzes data to solve problems by developing actionable insights" [87]. This definition introduces the aspect of data and actionable insights. The results of data science projects are often referred to as actionable insights.

Examining meta studies about data science project methodologies, it becomes clear that many of the methodologies mentioned in them contain only parts of what is suggested in the more specialized methodology definitions [88]. This leads us to the conclusion that for the remainder of this chapter and thesis, we will adapt the definition of PMBOK as it is the most inclusive of the definitions: A data science project methodology is a system of practices, techniques, procedures, and rules used by those who work in the discipline of data science.

2.5.3 Heavyweight Software Development Methodologies

Data science and software development are undoubtedly closely related. Due to the maturity of research on project management in software development, many of the methodologies in data science are built upon software development methodologies or at least borrow from their ideas. To understand data science methodologies, we first examine methodologies in software development. We start with so called heavyweight software development methodologies. With these methodologies, projects follow mostly linear trajectories and phases follow one another. Heavy planning is done in early phases of the projects, which makes it hard to adapt to an evolving understanding of what customers need.

The Waterfall Methodology

Although it is debated whether something like the waterfall methodology really exists in software development [89], it is often referenced as a traditional approach. Traces of the waterfall model go back to the 1950s when Benington introduced the nine phase stage-wise model [90]. This model was used to build an air defense software system called SAGE and consisted of 9 phases which were all dependent on the respective previous phase: Operational plan, operational specifications, program specifications, coding specifications, coding, parameter testing, assembly testing, shakedown, evaluation. Although described as a straight-forward top-down waterfall approach, Benington later added that they followed a more iterative approach of first building a prototype and then the full system.

The model closest to what is called waterfall model today was first presented by Royce in 1970 [91]. In his publication Royce shows multiple project life cycles for software development. The simplest form resembles a similar structure as the one originally described by Benington. The phases specified by Royce are: System requirements, software requirements, analysis, program design, coding, testing and operations. Even though his model is seen as a prototype of the waterfall model, Royce emphasizes that the model should not be applied as is and that it needs iterations between phases. The phases in his model are ordered in a way that iterations are mostly between neighboring phases and seldom between distant phases. However, he acknowledges that there are cases which lead to larger steps back in the project cycle, e.g. from the testing phase all the way back to the software requirements

phase. To avoid costly steps back in the life cycle, he suggests adding a preliminary design phase in between software requirements and analysis, to document design decisions across the whole life cycle, to build pilot models before building the whole system, to plan, control and monitor testing and lastly to involve the customer even in early stages of the project.

Finally, Boehm presented a version of the waterfall model based on Royce version [91] which consists of the following 8 phases: System feasibility, software plans and requirements, product design, detailed design, code, integration, implementation, operations and maintenance. In his model successive phases are linked in both directions and he adds validation, verification or test activities to the phases.

The Spiral Model

In 1986 Barry Boehm suggested the spiral model as a new software development methodology [92]. A major differentiation factor to previous models is the iterative risk handling by assessing and then eliminating risks in each loop through the model. As the name suggests, the spiral model is spiral-shaped, with each circle of the spiral growing in radial dimension. The spiral model is split into four quadrants, namely:

1. Determine objectives, alternatives, constraints
2. Evaluate alternatives, identify, resolve risks
3. Develop, verify next-level product
4. Plan next phases

One pass through all of the quadrants is considered a project phase. Each cycle begins in the first quadrant, with the goal to determine objectives, assess alternatives and identify constraints of the alternatives with regards to factors such as cost, schedule, interface and so on. Afterwards, each phase contains a risk analysis, uncovering sources of uncertainty. Various methods, such as prototyping, are suggested to resolve risks. In which way the spiral is traversed is dependent on how well the risks can be mitigated. The complete model, as depicted in Figure 2.12 suggests following typical software development steps such as conceptualizing, requirements elicitation, design, development, testing and implementation. Note that each phase ends with a validation and Boehm suggests having review meetings at the end of each phase with all important stakeholders. In this review meeting the previous cycle is evaluated and all parties agree on the plans for the next cycle, including a resource plan.

Boehm suggests that projects can contain multiple spirals, e.g. a second spiral can be started after a finished product has been used practically and the user experience leads to new feature demands or change requests.

2.5.4 Agile Software Development Methodologies

In 2001 the agile manifesto was created by 17 software engineers in response to the problems with heavyweight methods [93]. The agile manifesto emphasizes four values, based on 12 principles. The values of the agile manifesto are:

1. Individuals and interactions over processes and tools

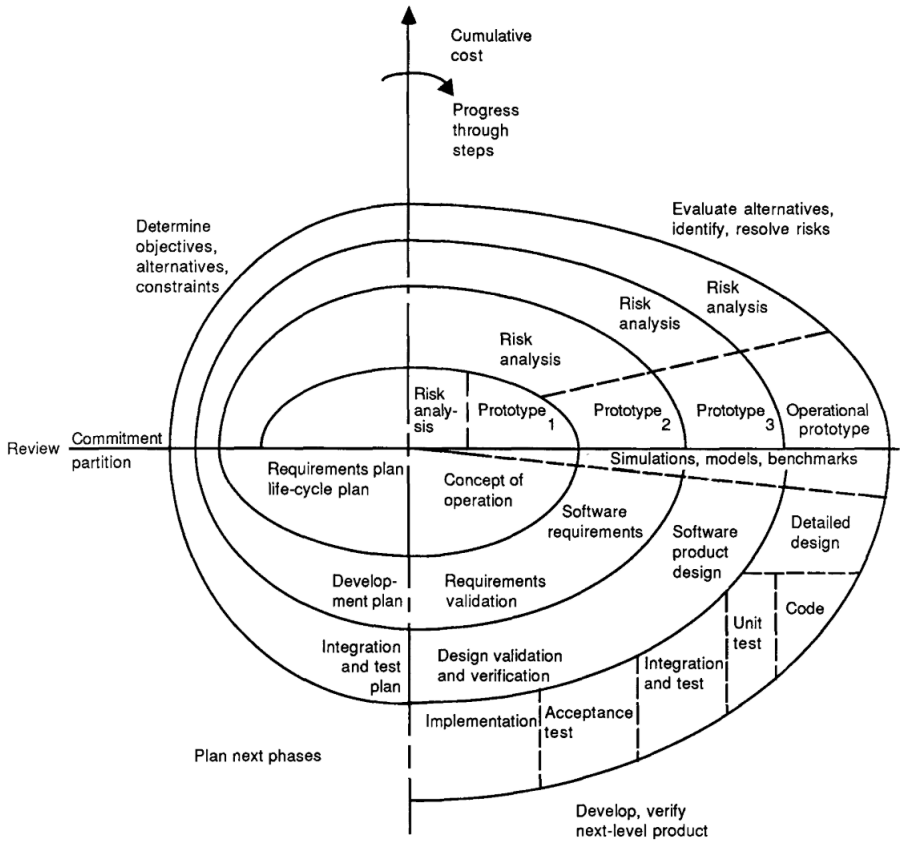

Figure 2.12: The spiral model by Barry Boehm [92]

2. Working software over comprehensive documentation

3. Customer collaboration over contract negotiation

4. Responding to change over following a plan

The values and principles embrace customer involvement and feedback, flexibility, producing working software, efficiency, communication and self-optimization.

There are many lightweight methodologies for software development, some of which were even published before the agile manifesto. We will focus on Scrum, Kanban and Extreme Programming next, as they are often referenced by data science methodologies and have partially been evaluated for use in data science projects [94].

Scrum

The Scrum methodology is still very popular today but was originally already presented in 1987 by Ken Schwaber [95]. It aims to tackle problems of methodologies which follow a linear project order such as the waterfall and the spiral model. One of their major downsides according to Schwaber is the assumption that software development is a defined, predictable process. He therefore proposes Scrum

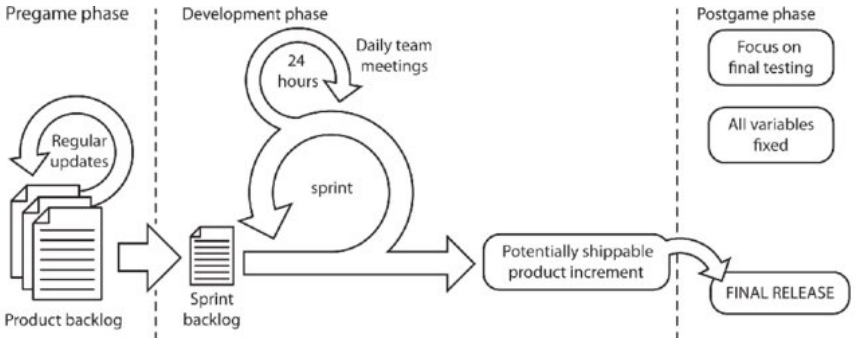

Figure 2.13: The Scrum phases as depicted in [96]

as a methodology with a lot of flexibility and appropriate control, based on theories from industrial process control and systems development.

Scrum assumes that teams have to adapt to changing environments and have to deal with imprecision. In contrast to the waterfall and spiral methodology, Scrum only starts with an initial plan and a broad deliverable definition. Both evolve during the development process, based on environment changes. It provides control mechanisms to account for an incompletely defined development process.

Scrum contains three groups of phases called Pregame, Game and Postgame. The Pregame group contains the Planning and Architecture phases. In the Planning phase, a new release of the software system is defined based on a known backlog of features and improvements. This includes the estimation of costs and a schedule. This phase contains different activities depending on whether a new system is built or an existing system is altered. In the Architecture phase, the implementation of items from the backlog is designed including adjustments on a system level to account for the new developments. The Game or Development group consists of development sprints. In these sprints, the actual development takes place in iterations, where each iteration should evolve the system. The duration and frequency of the sprints is controlled by time, requirements, quality, cost and competition constraints. Typical sprint lengths are one to four weeks. A review takes place after each sprint, including the development team and management but also optionally customers, sales, marketing and others. The Postgame group consists of the Closure phase. Here the system is prepared for release, changes are documented and tested and finally the system is released.

Scrum gives detailed task descriptions for each phase and provides multiple means of control for the process. It also specifies project teams and allows for multiple teams to work on the same product. Each team should consist of management and developers. The development team should ideally consist of three to six people. The whole project is managed by a product manager.

Kanban

Kanban shares many similarities with Scrum. Work is broken down into small tasks which are handled by self-organizing teams. Communication and meetings are central elements of both methodologies with the goal to deliver working software frequently.

However, there are some key differences. In Kanban there are no iterations but rather a continuous stream of work and tasks. This allows for rapid adjustment to new requirements and reprioritization. Instead of sprints, Kanban focuses on tasks and measures lead-time, i.e. the time it takes to finish a task. Additionally, there are no predefined roles in Kanban and the whole methodology aims to fit to

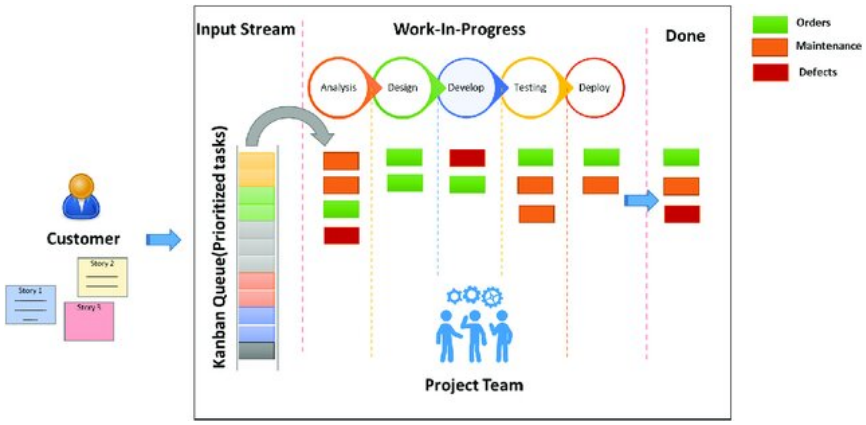

Figure 2.14: The Kanban board as depicted in [97]

working processes and structures.

Kanban is centered around a ticket board where tasks are documented and subsequently assigned to different columns indicating whether something is planned, in development or finished.

Extreme Programming

Extreme programming is another agile methodology for software development which puts even more emphasis on reacting to changing customer requests and satisfying their needs. According to Kent Beck it focuses on five main values: communication, simplicity, feedback, respect and courage [98]. It embraces constant communication between programmers and customers, encourages frequent and early deliveries of updates to get rapid feedback, simplicity of solutions and high quality of work. Essential elements of extreme programming are prioritization of features, honest planning involving all stakeholders, daily communication with team empowerment and producing working software.

Don Wells lists some rules for extreme programming, grouped according to planning, managing, designing, coding and testing. These rules include the creation of written user stories, frequent communication with the customer, pair programming, writing unit tests before coding, frequent small releases, iteration planning at the beginning of each iteration and so on.[1] Together these rules produce the workflow for project execution displayed in Figure 2.15.

Planning and feedback loops in the project are used in multiple granularities as can be seen in Figure 2.15. Extreme programming advises to execute monthly release plans, weekly iteration plans, acceptance tests with the customer every couple of days, daily stand-up meetings, hourly pair negotiations in pair programming, unit testing every couple of minutes and communication in pair programming down to a level of seconds.

Applying Software Development Methodologies in Data Science

The application of software development methodologies in data science has been studied previously [18]. However, there are multiple factors which lead to the need for specialized data science methodologies.

[1] For a detailed description we refer the reader to: http://www.extremeprogramming.org/rules.html

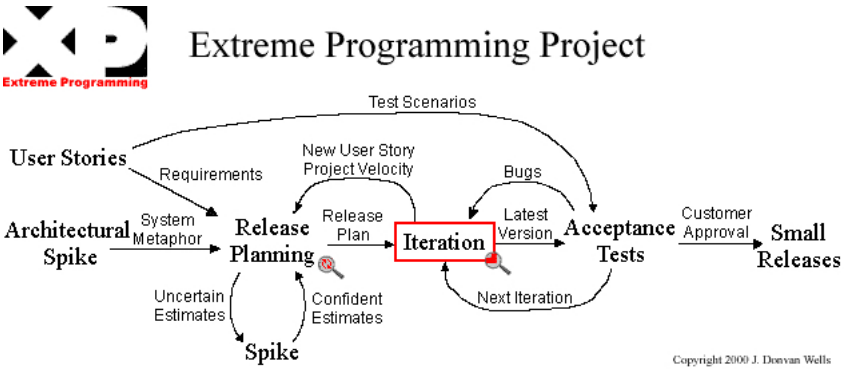

Figure 2.15: The Extreme Programming Flow Chart. Source: `http://www.extremeprogramming.org/rules.html`

Saltz argues that there are major differentiation factors between software development and data science [11]. One differentiation factor is data, as it has a different role in software development than in data science. In data science, obtaining, cleaning validating and storing data plays a different role and there are no phases in software development projects which account for this. He further argues that data quality differs between the two fields since data science projects often have to deal with noise in data. Sculley et al. also identify factors such as data dependencies which are significantly different to software development and argue that those lead to technical debt in machine learning systems [99]. According to Jurney a key differentiation factor between data science and software development is that data science is an applied research process, centered around data [6]. He elaborates that analytics products have a conflict between application development and research timelines and that the research timeline depends on the data.

Both, application development and research, are important aspects of data science projects. While the research activities yield insights about the data and which models are appropriate to solve a certain machine learning problem given the project data, the application development ensures that the results of the research are usable for an end-user. According to Jurney, not accounting for the research activities in data science leads to a degradation of all data science projects to waterfall-like projects [6].

To quantify the usefulness of different methodologies in data science projects, Saltz et al. performed a controlled experiment to benchmark Agile Scrum, a variant of Kanban adapted to data science phases, the CRISP-DM methodology and as a baseline, no explicit methodology [18]. In the experiment, multiple teams worked on an identical data science assignment, using one of the aforementioned methodologies. After the completion of the assignment Saltz et al. measured the willingness to work together in the future, satisfaction of team members, output quality and team dynamics and status updates. Both, the Kanban variant adapted to data science and CRISP-DM outperformed Scrum and the baseline. The teams which worked with the baseline methodology converged to a quasi-CRISP-DM methodology, i.e. they identified the need for data science specific project phases. The results clearly indicate that project methodologies tailored towards data science can benefit project teams in data science projects.

2.5.5 Data Science Project Methodologies

Over the course of the last 30 years, many project methodologies have been suggested for data science. The Cross Industry Standard Process for Data Mining (CRISP-DM) [2] remains the most popular methodology to date [100]. Most of modern methodologies adapt or at least reference parts of CRISP-DM in their design choice [101]. Some reference its predecessor, the Knowledge Discovery on Data Mining (KDD) [29] methodology. We present the KDD methodology and other, more recent, methodologies in Section A.1 in the appendix and review them in Chapter 4.

Structuring Data Science Project Methodologies

In Subsection 3.3.1 we discuss the results of eight meta-studies which analyze data science methodologies. All of them present methodologies in an unstructured way. The methodologies they analyze do not all contain the same components and structure. To make methodologies systematically comparable, both qualitatively and quantitatively, we propose our own structure. It is based on an analysis of literature about project methodologies and aligns with the key information that a methodology should provide to help with executing data science projects.

We propose to structure the methodologies into four distinct components:

1. **Life cycle, processes and tasks:** This component includes the life cycle, project phases, processes, methods and procedures. In essence, this component structures projects into typical phases and subphases, establishes typical activities through these phases and proposes standard transitions between them. Charvat emphasises that methodologies should include concrete project implementation details in the form of methods, procedures and standards [3]. He adds that methodologies often contain project phases. The PMBOK further states that methodologies should help with the appropriate selection of processes and life cycles [85] and Charvat emphasizes that methodologies should be tailored towards the phases in the project life cycle [3].

2. **Team Organization:** This component encompasses team organization aspects, including project roles and responsibilities. According to the Charvat, methodologies should help project managers to manage their resources, such as the personnel, and ease the communication between team members.

3. **Artifacts:** This component includes all the artifacts which are used and created in a project such as all types of templates, forms, checklists, to-do lists, milestones, deliverables, inputs and outputs. Important artifacts in data science projects are the data, machine learning models, visualizations, documentation of results and many more.

4. **Tools and techniques:** This component summarizes the tools and techniques suggested by a methodology.

To verify the usefulness of this structure we examine the definitions presented in subsection 2.5.2. Charvat's definition highlights that methodologies provide guidelines and principles and should be adaptable towards specific project contexts. It adds that there are certain artifacts such as to-do lists, templates, forms and checklists which are used over a **project life cycle**. The second definition highlights that methodologies bridge the gap between management and engineering and adds that the

output of a project are often products, services or solutions. The last definition adds the aspects of roles, responsibilities and milestones.

Charvat gives some recommendations on how methodologies should be used and what they should achieve. He states that all team members should be aware of the methodology, it should guide them through the whole life cycle of projects and should not only consider a single project, but all projects in the team, i.e. take a holistic view [3] of an organization to account for shared resources. In addition, Charvat lists four common parts of methodologies: they often contain project phases, measure the project progress, guide the team to take corrective actions based on problems and assign resources to the different project phases [3]. Finally, Charvat also discusses responsibilities of project methodologies. According to him, they should enable project managers to manage the project performance, the project life cycle, the resources and the communication between them [3].

The PMBOK states that good methodologies should be flexible and allow for tailoring towards individual projects, to accommodate the uniqueness of each project [85]. They state that that the selection of appropriate processes, inputs, tools, techniques, outputs and life cycle phases is project-individual.

To understand whether these components are helpful in data science contexts, we analyze literature about data science methodologies.

In his position paper "The Need for New Processes, Methodologies and Tools to Support Big Data Teams and Improve Big Data Project Effectiveness", Saltz discusses the needs and goals of data science methodologies as well as approaches to define them, to assess a team's performance and what kind of tools could be useful to support teams [11]. Saltz names multiple reasons why project methodologies are helpful for data science teams. He calls for well-defined team roles to assure correct and complete task-assignment and coordination (component 2). He adds that methodologies are helpful to assure that teams think about all necessary tasks in data science projects and to share best practices (component 1). Further, he emphasizes the need for well-defined processes and phases to assure good communication, prioritization, problem understanding and so on (component 1). Lastly, he calls for more focus on appropriate tools and techniques as a part of methodologies to support data science teams (component 4).

Martinez et al. present a study in which they assess 19 data science project methodologies with respect to data science project challenges [88]. In the study they discover three main groups of challenges, team management challenges, project management challenges and data & information management challenges. For each methodology the authors determine how well the individual challenges are tackled. They claim that a data science project methodology should be integral, i.e. include aspects of project management, team management and data & information management. Such a methodology should provide information about project phases and tasks (component 1), should provide information about the roles and their responsibilities (component 2), should contain information about goals, outcomes and deliverables and give guidance on how to manage code, data and models (component 3) as well as recommend and provide tools for this purpose (component 4). They call such integral methodologies a holistic approach to data science and argue for its necessity.

2.5.6 CRISP-DM

The Cross Industry Standard Process for Data Mining (CRISP-DM) [2] was introduced in 2000 and succeeded the KDD process. Recent polls show that even 20 years after its introduction, it is still considered the most widely adopted data science methodology [100]. CRISP-DM structures data

Figure 2.16: The Generic Process CRISP-DM cycle, showing the highest level with its six phases. [2]

mining projects into four levels: Phases, generic tasks, specialized tasks and process instances. The first level, the phases, provide a generic view on data mining projects. They are divided into business understanding, data understanding, data preparation, modeling, evaluation, and deployment. All phases are centered around the available data. Each phase is comprised of generic tasks, which should be independent of the application and models used, and specialized tasks, which describe how actions should be carried out in specific situations. Lastly in the process instance level, particular instances are described based on the tasks defined on the generic and specific level.

The goal of the original CRISP-DM paper was to establish a standard for data mining projects, to transition the field of data mining into a more mature state [2]. To this end, the authors present a rather generic methodology and emphasize the need for specialization to specific domains and projects. The description of the phases and the generic tasks in [2] is rather high-level with a brief mention of outputs of all tasks. This is enriched with a user guide and more detailed descriptions of artifacts and tasks in [102]. The phases of CRISP-DM are embedded in a project life cycle which highlights the most common phase transitions.

Life Cycle, Tasks and Processes

The CRISP-DM phases, displayed in Figure A.2, follow a cyclic nature. The outcome of each of the phases defines which phase should be visited next. According to CRISP-DM, typically more than one iteration through the cycle is performed in a project. The authors justify this with the not straightforward nature of data mining projects. They further emphasize that transitions between phases are project-dependent and do not always follow a fixed order. They add that phase transitions other than the ones specified in their CRISP-DM cycle can occur. In the following subsections we will shortly summarize the six phases of CRISP-DM:

Phase 1: Business Understanding The authors suggest several generic tasks in the business

understanding phase. One of the goals of this phase is to understand the background of the project, the business objectives as well as the business success criteria. The project situation is assessed to derive the available resources, project requirements, underlying assumptions and constraints. Risks and contingencies are determined, a project terminology is created and costs and benefits are listed. With the now available knowledge about the is-state and the business objectives, data mining goals are determined along with data mining success criteria. Given all this information a first project plan is produced with an initial assessment of the tools and techniques that should be used.

Phase 2: Data Understanding According to the authors the data understanding phase is tightly coupled to the business understanding phase. This intuitively makes sense, as an understanding of the business objective will help to determine the right data, while an understanding of the data should lead to a better project plan. This phase starts with an initial collection of data, which amounts to an initial data collection report. The collected data is then described in a data description report. Next, the data is explored and the data quality is evaluated which leads to a data exploration and data quality report.

Phase 3: Data Preparation The third phase proposed in CRISP-DM is the data preparation phase. Its goal is the creation of the final data set and its description. The phase begins with the selection of the data via a rationale for inclusion and exclusion. The selected data is then cleaned and the cleaning steps are reported in a data cleaning report. Next is the construct data task, in which attributes are derived and new records are generated. Finally, data from different sources is integrated and reformatted to assure compatibility with the modeling tools. Garcia et al. further describe a data reduction step in which irrelevant records are removed [103].

Phase 4: Modeling The modeling phase starts with the selection of modeling techniques based on modeling assumptions. Next, tests are designed to assess the quality of the model. The main objective in this phase is to build and assess the model. Building the model requires picking appropriate parameters and should not only result in the models but also in a description of them. During the model assessment it might be necessary to revise the parameter settings. The authors note that models and their assessment often lead to the discovery of problems with the data, which might amount to a transition back to the data preparation phase and the construction of new data.

Phase 5: Evaluation While the model quality has been assessed in the prior phase, in this phase the models are assessed as to how well they meet the business objectives. This assessment involves comparing the outcomes of the data mining models with the predefined success criteria. If the models meet the success criteria, they are approved. The whole process of the project is reviewed and the next steps are determined, yielding a list of possible actions. Finally, a decision is made regarding the next steps to be taken in the project.

Phase 6: Deployment The last project phase is the deployment phase. Initially, the deployment is planned. If the project requires running the data mining solution in a productive environment, monitoring and maintenance of the deployed solutions is planned. Depending on the needs of the user, deployment can range from a final report to software integration. The authors claim that the deployment is often performed by the user, but knowledge about the deployment nature is needed upfront. Lastly, the project is reviewed and the experience of everyone is documented.

Team Organization

CRISP-DM mentions 16 roles along the project life cycle. There are no explicit descriptions of all of the roles and their responsibilities and only for some of them the tasks are mentioned in the phase descriptions. The following roles are mentioned: Customers, Data Analysts, Business Experts,

Internal Sponsors, Data Experts, Technical Support, Data Mining Personnel, Data Mining Engineers, Business Analysts, System Administrators, Database Administrators, Market Analysts, Statisticians, Domain Experts, Project Leaders and End Users.

Artifacts

CRISP-DM provides extensive descriptions of artifacts which are generated throughout a project lifecycle. They suggest 41 artifacts which are grouped according to the phases they are produced/used in. For easier readability we offload the detailed listing and descriptions of the artifacts to the appendix and just provide a general overview here.

- **Business Understanding**: The business understanding phase contains 12 different artifacts, which provide important project information for both the client as well as the data mining team. The artifacts in this phase are documents which mainly describe the goal and background of the project, the available resources for the project, project constraints and the project plan.
- **Data Understanding**: This phase contains four reports which describe how data is obtained, where it is stored, provides information about the data itself, interesting findings in the data and a summary of its quality.
- **Data Preparation**: Aside of the documents describing the data preparation methods and decisions, this phase also contains the dataset itself and its modified versions as artifacts. In total it contains seven different artifacts.
- **Data Modeling**: The data modeling phase contains eight artifacts which range from descriptions of modeling choices to the models and parameter choices themselves and result assessments.
- **Evaluation**: In the evaluation phase CRISP-DM introduces five artifacts. These artifacts contain the evaluation results, the approved models, a review of the whole data mining process, lists of possible next actions and a documented decision on how to proceed.
- **Deployment**: The deployment phase also contains five artifacts. These artifacts describe how the results of the project are deployed, they include a plan for maintenance and monitoring of the result, final reports and presentations as well as an experience documentation for knowledge dissemination.

Tools and Techniques

In contrast to other methodologies, CRISP-DM does not provide a list of tools and techniques.

2.6 Conclusion

The data analysis field has evolved from exploratory knowledge discovery in databases to big data analysis, data science and MLOps. Today, data science projects often produce models and software which are deployed as solutions and have special requirements for operation and maintenance. Since more and more unstructured data, such as text, is available, subfields of data science such as NLU have become more important.

Recently a new machine learning type called self-supervised learning has led to a new paradigm in training models. Models are now pre-trained on large volumes of data in a self-supervised manner and then adapted to specific tasks, either via fine-tuning, i.e. a separate training step, or via prompts in natural language. So called language models are pre-trained on masses of texts in a self-supervised manner, learning general knowledge about the semantics and syntax of languages, but also capturing general knowledge. These models are then adapted to perform specific tasks, such as named entity recognition. The adaptation requires less feature or model engineering than prior approaches as well as less training data to get better results.

The transformer architecture and its encoder-only and decoder-only variations have replaced RNNs as dominant machine learning model in various subfields of data science such as NLU. They have proven to be very efficient for training and scaling models to more parameters. Very large models, which are trained with the pre-training and adaptation scheme, sometimes across modalities, are called foundation models. Foundation models, such as large language models have shown great transfer learning capabilities and beat benchmarks across NLU tasks. Since the transformer architecture can be efficiently parallelized across GPUs, larger models have high hardware requirements, which need to be accounted for.

Experiments have shown that the performance of such models scales with their size and the amount of data they have been trained on. Encoder-like foundation models are pre-trained once and can then be fine-tuned to perform arbitrary tasks with only few training examples. Large decoder- or transformer-like models can be instructed to perform tasks via natural language prompts in a zero-shot manner, in some tasks even beating models specifically fine-tuned to the task. However, the reliable application of such models for NLU tasks such as information extraction and classification is still a topic of research as the answers of the models may vary in content, length and format. In NLU projects, the pre-training and adaptation paradigm is the state-of-the-art solution and is hence the focus of this thesis.

Foundation models have significantly improved the state-of-the-art across various media. They beat every benchmark for NLU tasks and enable new applications. On the other hand, training large language models is prohibitively expensive in terms of resources and time [69]. This development has led to a paradigm-shift, making it more and more important to re-use models and version them with respect to the datasets and tasks they have been trained on.

Data science project methodologies provide guidance for data science projects. Research on project methodologies in software development has strongly influenced the development of data science methodologies. Software development methodologies can be classified as heavyweight or agile methodologies. Heavyweight methodologies often require a lot of planning upfront and are less flexible with concern to the order of phases and tasks. In contrast agile approaches promote short iterative planning cycles and are generally tailored towards incorporating feedback from users and domain experts frequently. Due to the data- and research-driven nature of data science projects, software development methodologies cannot simply be reused in data science projects. Dedicated data science methodologies should provide guidance with respect to the lifecycle, processes and tasks, roles and responsibilities, tools and techniques as well artifacts of data science projects. CRISP-DM, which is more than 20 years old, remains the most popular data science methodology to date.

CHAPTER 3

An Evidence-Based and Comprehensive Framework for Assessing and Comparing Data Science Methodologies

In this chapter, we establish a catalog of requirements that a project methodology should meet, in order to be suitable for foundation model projects. In Section 3.1, we show that existing methodologies have always been tailored and adapted towards changing project characteristics, i.e. challenges, project requirements and success factors. Then, in Section 3.2 we describe a systematic literature study, which we use to derive a catalog of such project characteristics in Section 3.3. The catalog is tailored towards today's business needs and the usage of foundation models. In our literature study we follow a multi-step approach. We start by analyzing general data science project management literature. This provides us with a broad theoretical overview of project characteristics. Unsurprisingly, we find that project management literature has not yet addressed the influence of foundation models on data science projects. Hence, we study state-of-the-art publications about foundation models and enrich and adapt the characteristics according to our insights. We structure the characteristics into a hierarchical catalog, consisting of 165 aggregated groups. Finally, we review our catalog from a practical perspective. In Section A.2 we analyze 26 case studies of NLU projects with respect to the catalog. The case studies provide us with practical insights into the importance of the characteristics and help us to condense the catalog to a project-relevant subset. Both, the complete catalog and its project-relevant subset will serve as catalog of requirements which a foundation model project methodology should address. A summary of the process is depicted in Figure 3.1.

3.1 Adapting and Tailoring Data Science Project Methodologies

Methodologies should always be adapted according to the evolution of the data science field, i.e. technical developments, changing challenges and customer needs. Project management literature primarily discusses three types of project characteristics with regards to analyzing how to execute data science projects:

1. **Challenges**, which should be tackled by a project methodology and could lead to a project failure. A typical example would be a customer with unrealistic expectations of a project

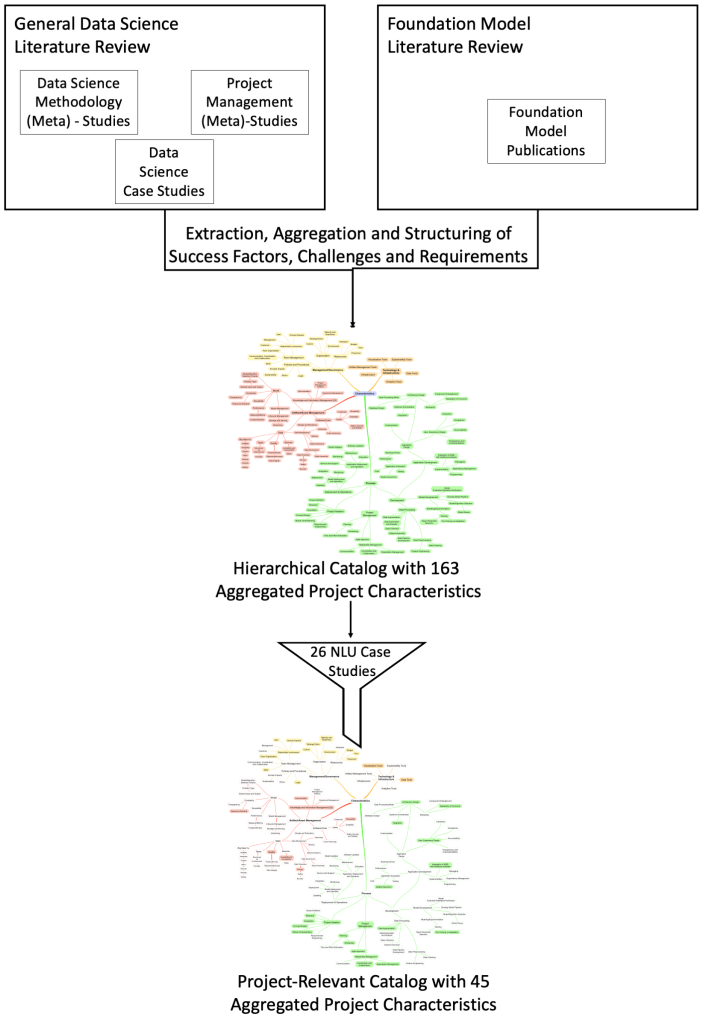

Figure 3.1: The process of deriving the complete catalog of characteristics and the project-relevant subset.

result. A case of adapting methodologies to challenges is [104], which specifically tailored the ASUM-DM methodology towards addressing challenges in multi-stakeholder projects.

2. **Success factors**, which increase the likelihood of project success, such as executing a feasibility study before starting large projects. Soukaina et al. analyze methodologies with respect to such factors [105].

3. **Project requirements**, i.e. specific conditions or functions that need to be accounted for, such as hardware limitations on the customer side. Schroer et al. discovered that methodologies like CRISP-DM were adapted towards changing requirements due to novel technology [106].

3.1.1 Comparing Foundation Model Projects to Traditional Projects

In Chapter 2 we explained that foundation models induce a novel paradigm of pre-training and adaptation. This has radical implications on the execution of data science projects.

Pre-training foundation models amplifies requirements of traditional machine learning training. Vast amounts of data need to be gathered, cleaned, preprocessed and generally managed. Handling these amounts of data requires specific infrastructure. It is beneficial to establish data storages which allow the reuse of datasets for later training runs without having to repeat the complete data processing steps. Further, many foundation models are so large that they cannot be trained on a single machine. Their training requires large amounts of compute power and effective parallelization across machines. Training such models on hardware from cloud providers such AWS or Azure can easily cost millions of dollars [107].

Foundation models are typically pre-trained on as much general data as possible. However, it has been shown that it is beneficial to take pre-trained models and continue training them on domain- and task-specific data [108]. This way domain- or task-specific derivatives of foundation models can be created, which calls for the establishment of model management infrastructure and processes.

An unparalleled benefit of pre-trained foundation models is their adaptability. Once pre-trained, the models can either be adapted via fine-tuning, i.e., training them on smaller, task-specific datasets, or by prompting them with natural language instructions. This reusability and adaptability significantly change the modeling phase of data science projects. Especially, when considering that many pre-trained models are available for download from public sources, even with licenses that permit commercial use [84]. These models come with their own tokenizers and often require only minimal data preprocessing for fine-tuning and inference. This is displayed in Figure 3.2 by comparing various representations in NLU, starting from BoW-like representations over distributed embeddings, contextualized embeddings and finally generative large language models.

Finally, models such as GPT-3 or GPT-4 are available via REST-API, i.e. companies which want to use these models do not have to worry about training or even hosting their own models. The work is then reduced to building applications around the API which handle the inputs and outputs of the model as well as defining the right prompts, i.e., instructions for the model. However, this requires careful engineering as these models tend to hallucinate, i.e. provide wrongful answers [109].

We conclude that foundation models represent a paradigm shift in machine learning and in the execution of data science projects. Even though data remains a vital artifact, data science projects using foundation models require the management and handling of model artifacts across projects. To motivate the need for a foundation model specific project methodology we first show that paradigm shifts in data science have always led to novel project methodologies. Afterwards, we empirically study the characteristics of projects which make use of foundation models.

3.1.2 The Evolution of Data Science Project Methodologies

Over the last 30 years, as the characteristics of data science projects have continuously evolved, so too have data science methodologies. To study their evolution, we identify relevant methodologies in literature. We use the meta-studies by Saltz et al. [110] and Martinez et al [88] to identify a set of data science project methodologies. We enrich this set via literature search, searching for a conjuncture of ”data science” and ”project methodology” in the Google Scholar, IEEE, Elsevier Scopus, ACM and arxiv databases. Our final set contains 27 different methodologies. Our study reveals that these methodologies can be categorized according to the project characteristics which they primarily address. We propose to align them in five different groups which represent general shifts in data science paradigms:

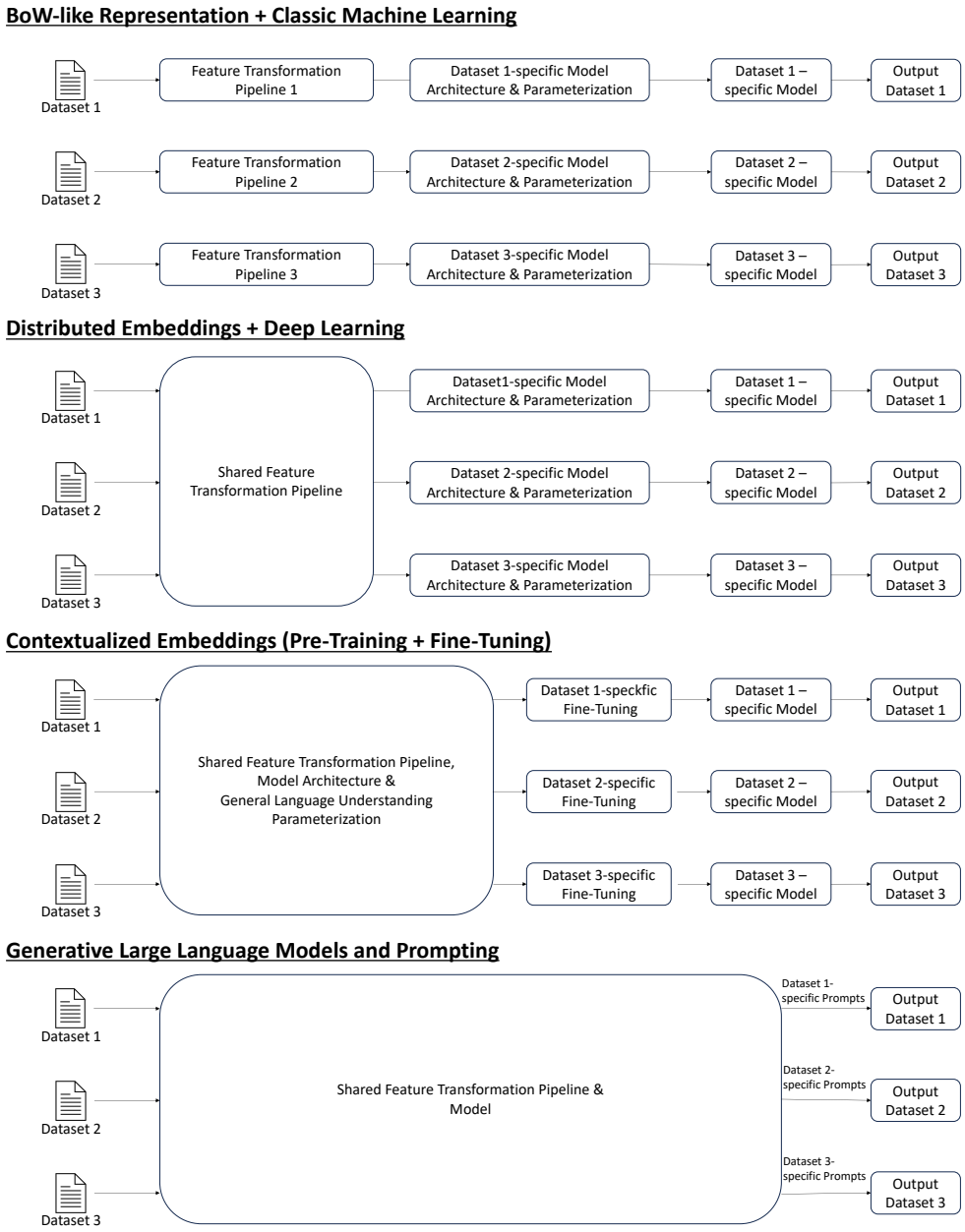

Figure 3.2: The evolution of text representations gradually allowed sharing more information across projects. Earlier representations required completely new feature and model engineering for each project. Distributed embeddings introduced feature sharing across projects. With contextualized embeddings, whole pre-trained models are shared across projects and only slightly modified for task-specific outputs and fine-tuned on training data. Finally, large generative language models only require specific prompts for adaptation.

1. **KDD/Data Mining Group:** The first methodologies in this group, KDD [30] and CRISP-DM [2], recognized the need for specialized data science project methodologies. In particular they acknowledged two differences between software engineering and data science: First, that data is the central artifact in data science projects and second, that data science projects are more like research projects by nature. We also place direct extensions of CRISP-DM in thuis group.

2. **Big Data Group:** These methodologies were tailored and adapted to project characteristics, specific to big data. In particular, the "Big Data V" challenges. The methodologies in this group also particularly address organizational characteristics which are important when executing big data projects.

3. **Agile Group:** Methodologies of this group introduce agile methods in data science projects. Their goal is to improve team communication, coordination and collaboration as well as to achieve a better adjustment to changes and closer cooperation with customers. This became particularly important because of growing data science teams and increasing project complexity.

4. **Integrated Group:** These methodologies integrate software and often cloud frameworks with data science methodologies. They tailor their software frameworks to their proposed project life cycles.

5. **Software Development and Operations Group:** This group of methodologies focuses on the development of functioning data science applications as well as their deployment, maintenance and operation. This group stems from the shift of the data science field towards productive solutions which are integrated in other systems.

Many of the methodologies fit to more than one of the aforementioned groups. It makes intuitive sense for newer methodologies to adapt practices from older methodologies. For each methodology which fits into multiple groups, we select its group based on the primary challenges that it aims to tackle, e.g. a lot of agile methodologies also mention big data but rather focus on challenges in team collaboration and life cycle management.

In addition to the five groups, we introduce a sixth group for methodologies, tailored towards NLU. Since foundation models are predominantly used in NLU, we assume that these methodologies provide valuable insights for a foundation model project methodology.

We will now describe each of the groups in more detail and highlight the methodologies which we assigned to them.

KDD and Data Mining Group

Early methodologies for data science recognized that data science projects have distinct characteristics which differentiate them from software development. In particular, they recognized that data is the central artifact in data science projects and that data science projects have a strong research focus and hence a high degree of uncertainty. The first notable data science methodology is the "Knowledge Discovery in Databases" methodology [29, 30]. It recognized the need for a structured process for knowledge discovery projects to guide data miners in their interactions with large databases. CRISP-DM succeeded the KDD methodology and aimed at setting an industry-wide standard process for data mining [2]. It presented a project methodology with phases similar to the ones in KDD but placed an emphasis on business understanding. Further, CRISP-DM came with a user guide including

detailed descriptions of tasks and artifacts [102]. One of the goals of CRISP-DM was to transition the data mining field into a more mature state. It was created by a consortium of experts with industrial and research experience in data mining. Aside of the detailed guidelines that CRISP-DM provides, another indicator for its more mature nature are the roles mentioned in the methodology. While the project team in KDD consists of a single data miner that interacts with business users, CRISP-DM mentions a lot more roles including technical support, project leaders, database administrators, etc., albeit with no clear descriptions of the roles and their responsibilities. Another noticeable methodology from this group is RAMSYS [111]. RAMSYS is an adaptation of CRISP-DM to settings in which teams collaborate on data mining projects remotely. The authors recognized a need for information management, data management and collaboration. They define guiding principles to ease remote collaboration and introduce tools and artifacts to support the methodology. The ”process model for data mining engineering” acknowledged the need for combining software engineering and data mining activities [86]. Finally, CASP-DM is an extension of CRISP-DM, which adds the aspect of context-awareness [112]. It recognizes that models have to be adapted to changing domains, e.g. due to data shift, and introduces relevant aspects with regards to model management.

Big Data Group

Starting from 2014 a lot of methodologies began to tackle the specific challenges of big data. In their review from 2019, Plotnikova et al. identify big data challenges as one of the main reasons for adaptation of data science methodologies [101]. In the ”Big Data Managing Framework”, Dutta and Bose recognize the need for a cultural change in companies to adapt to big data technologies [113]. They argue that the involvement of management from all levels is a key success factor and call for cross-functional workshops including business stakeholders as well as users and IT staff. Vanauer et al. argue similarly in their ”Big Data Ideation, Assessment and Implementation” methodology [114]. Their methodology involves cross-functional teams in all phases and strong involvement by management. The methodology tackles two different cases of introducing big data in a company. One for companies which want to change their own processes based on big data and another for companies which want to provide big data solutions. The ”Foundational Methodology for Data Science” calls on the aspect of emerging tools and techniques to handle the volume challenge of big data [115]. The Big Data Management Canvas [34] does not only focus on the volume of big data but rather on the 5Vs model of big data. The methodology puts an emphasis on the scalability of systems.

Agile Group

Saltz et al. recognized that there are notable similarities when comparing the evolution of data science projects with software development projects [11]. They argue that software and data science projects evolved similarly: from small solitary projects to large team-based efforts. Agile methodologies yield a lot of benefits when dealing with the challenges of large projects, especially when it comes to coordination of teams and reacting to changing requirements. Hence it is only natural, that Saltz et al. also observed a spike in methodologies and case studies which deal with agile methods for data science [110]. ASUM-DM is a methodology by IBM which embraces agile development [116]. The authors incorporate continuous project management in the methodology and suggest using prototyping sprints in the beginning of projects when requirements are still unclear. In their ”Towards Methods for Systematic Research on Big Data” methodology, Das et al. argue that agile development benefits the

research nature of data science projects [5]. They call for user-driven development with small iterations which yield improvements iteratively and incrementally in chunks of user-valued functionality. Ideally this functionality can be visualized as this eases the communication with business stakeholders. Grady et al. argue similarly in their "Knowledge Discovery in Data Science" methodology [31]. They add that data science projects often turn into waterfall-like methodologies and that agile mitigates a lot of the waterfall problems by encouraging collaboration, frequent and early delivery of software and iterative development. The "Agile delivery framework for BI, fast analytics and data science" methodology shares this sentiment and even incorporates the agile manifesto [117]. The authors recommend iterative development, with timeboxed iterations during modeling and close iterations during visualization and validation. Jurney goes one step further and adapts the agile manifesto to data science, i.e. he creates the agile data science manifesto [6]. His agile data science methodology emphasizes the difference between software development and data science caused by the uncertain nature of data science projects. Jurney argues that one problem, which causes data science projects to succumb to waterfall projects, is having too many different people involved, i.e., each person is involved with a different role. His methodology mitigates this problem by utilizing people in different roles in the same project. Lastly, we add the MIDST [118] and "Analytics Canvas" methodologies [119] to this group as they tackle aspects of team communication and collaboration. MIDST borrows ideas from open source development and introduces a tool for stigmergic coordination on projects, i.e. providing stimuli to other team members to pick up unfinished work and continue. A previous study by Martinez et al. classified MIDST as a methodology [88], whereas we would rather name it a tool to support methodologies. The analytics canvas provides an easy-to-understand UML-like visualization for data science projects and can be useful for communication between stakeholders in a project. The publication also sketches a process of a data science project and can hence be considered a methodology, however the focus is on introducing the analytics canvas as artifact to support a project.

Integrated Framework Group

We identify a group of methodologies which are tightly integrated with software frameworks and sometimes specific infrastructure, such as cloud platforms. SEMMA, which is often referenced as a methodology in literature, can be placed into this category. Originally SEMMA was not intended to be a methodology but rather an organization of the functional toolset of the SAS Enterprise Miner [120]. SEMMA contains five phases, namely sample, explore, modify, model and assess, and provides tools and methods for all of these phases. Although there is no explicit phase for deployment, SAS Enterprise Miner also provides tools for this aspect of data science projects. Guo presents the data science workflow, originally dubbed the research programmers workflow [121]. Along with the workflow, Guo implemented tools for various tasks of data science, such as deployment and packaging, data and code management, debugging and experiment tracking. The data analytics lifecycle methodology [122] describes a joint methodology and cloud framework to automate all phases of the project lifecycle. In contrast to this more theoretical description, Microsoft provides a practical implementation of a joint methodology and cloud platform with TDSP and Azure [7]. TDSP provides a comprehensive methodology with detailed task descriptions, role and responsibility descriptions and artifact descriptions.

Software Development and Operations Group

With rising maturity of the data science field, the expectations of customers also increased. While it was enough to present valuable insights in the form of slides in the early stages of data science, customers have come to expect working solutions which yield insights and automate processes even after the finalization of a project. To this end data science software solutions need to be developed and deployed on the customer side. Software Development and Operation Methodologies set a focus on the challenges of software development, maintenance and operation and newer methodologies also on the maintenance and operation of machine learning models. All of these methodologies contain roles for product management and software development as well as recommendations for tools for the collaborative development of software and its deployment. The Domino DS lifecycle puts an emphasis on building reusable software components and datasets [123]. In the "development workflows for data scientists" methodologies, Byrnes reiterates the importance of sharing code within teams [124]. She adds the sharing of knowledge and models via knowledge repositories, which should be accessible to everyone within an organization. More recently, Thomas suggested an end-to-end workflow of data science in the data science & AI lifecycle, spanning from business understanding to the operationalization of AI [47]. This is the first methodology to explicitly describe the running, monitoring and management of models after deployment of solutions. He further puts an emphasis on the automation of large parts of data science projects and calls for the utilization of tools such as continuous integration and deployment. Most recently the term machine learning operations became more popular to describe end-to-end conceptualization of data science solutions [125]. MLOps combines elements from machine learning, software engineering and data engineering. The aim of MLOps is to build and maintain product-grade data science solutions.

NLU Group

Only two of the 27 methodologies that we discovered are tailored towards the specific challenges and requirements of textual data. Both of these methodologies are very recent, from 2021 and 2022. The "Cross-Industry Process Standardization for Text Analytics" methodology is a close adaptation of CRISP-DM for text analytics [126]. It recognizes specific project requirements created by the textual modality of the data and the specific methods developed to handle this sort of data. The biggest contribution of the methodology is the addition of text-specific methods and tools to the different project phases. However, it ignores recent developments such as the monitoring and operation of models. The Data2Value methodology by Leidner introduces more NLU specific changes to the project lifecycle [17], such as specific phases for the annotation of textual data. It also accounts for phases from MLOps and further introduces phases for ethics reviews in the beginning and end of projects. Both methodologies disregard the recent developments in the NLU field described in Chapter 2 and Subchapter 3.1.1. None of them account for the differences in project execution introduced by the use of foundation models or even the pre-training/adaptation paradigm.

3.2 A Systematic Study of Foundation Model Project Characteristics

To understand the requirements of a foundation model project methodology, we want to create a catalog of project characteristics in terms of challenges, success factors and requirements, which influence the execution of data science projects and, in particular, foundation model projects and hence

should be considered in a foundation model project methodology. Due to their recent development, the impact of foundation models on project execution is not yet well-studied. Hence, we build a catalog of NLU project characteristics by combining the results of two literature studies. To understand the requirements which a foundation model methodology should meet, we need to understand what the characteristics of projects involving foundation models are. There is already a lot of literature on requirements, success factors and challenges for general data science projects. However, to the best of our knowledge, there is none concerning projects which make use of foundation models. We argue that a modern methodology needs to reflect both, the characteristics of general data science projects and the characteristics specific to foundation models. To get a broad overview of general data science project characteristics, we harness existing knowledge in project management publications. To understand the specific characteristics of foundation model projects, we analyze literature about foundation models. Since most of the foundation model literature is about pre-trained language models, the majority of the publications we analyzed is related to NLU. To build the catalog, we follow the following steps.

1. We perform a systematic literature review of data science project management literature to gather requirements, challenges and success factors, which we will summarize as project characteristics. We build queries to discover publications which explicitly discuss such project characteristics. This includes (meta-)studies about methodologies and project management as well as data science case studies. We do not limit our search to foundation model related publications, as we expect there to be only limited literature available about case studies or meta-studies for foundation model projects. Further, we argue that there are general data science project characteristics, which every data science project methodology should address. These characteristics are not limited to foundation model usage.

2. To account for the lack of publications about projects using foundation models, we screen the current state-of-the-art literature about foundation models. We begin by screening surveys about foundation models and publications which describe the state-of-the-art methods in NLU, i.e. the methods which beat the most popular benchmarks for NLU tasks. Since foundation models are predominantly used in NLU, we expect to find the most literature in this discipline. Based on the publications, we extract challenges, success factors and requirements.

3. We thoroughly analyze the individual success factors, challenges and requirements, deduplicate and group them. We create a hierarchical catalog of characteristic groups.

The complete process of obtaining the general data science project characteristics is depicted in Figure 3.3.

3.2.1 Systematic Literature Review of General Data Science Project Characteristics

We design our study of general data science project characteristics based on the Kitchenham approach for systematic literature reviews [127], which is common for systematic literature reviews in data science. We mimic the structure of the studies of [128] and [129].

Based on our research question we obtain an initial list of publications which are relevant for our goal. We then analyze these publications and their references to obtain a set of relevant search terms. We validate the quality of the search terms by defining a set of search strings which are used to query common literature databases and verify that a set of held-out validation publications are included in

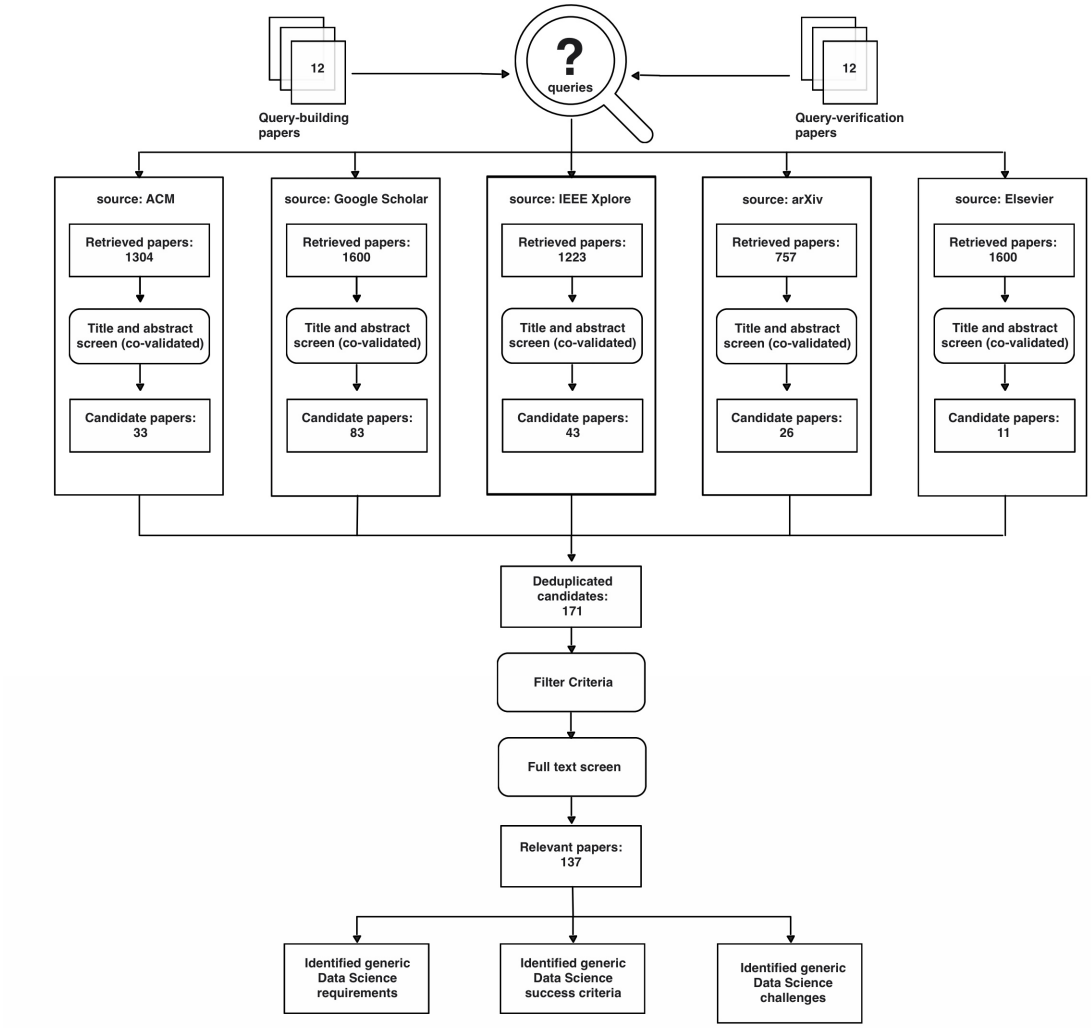

Figure 3.3: The process of the systematic literature review for obtaining challenges, success factors and project requirements of data science projects, discussed in literature.

the result set. If not all are included, we refine or expand the set of search queries. With our final set of queries, we query five different literature databases: IEEE Xplore, Elsevier Scopus, ACM library, arXiv and Google Scholar. Finally, we clean the results obtained by the different sources by eliminating duplicates. We filter publications which are more than five years old, as we are especially interested in understanding recent developments in data science, which account for developments such as MLOps and potentially even foundation models. We further filter out publications which are more than a year old and have no citations. In a next step we assess the remaining publications in a multi-staged voting process, starting with a title and abstract screening by two independent reviewers. Next, we perform a full-text screening of the publications which were deemed relevant. We include publications which explicitly discuss success factors, challenges or requirements of data science projects. We exclude publications which are specifically tailored towards a certain domain or technology other than NLU or foundation models. Finally, we perform an in-depth analysis of the remaining publications and extract and harmonize the resulting characteristics of data science projects.

Research Question

Our overall goal is to obtain an overview of data science project characteristics. We identified different types of publications which provide us with defining characteristics of data science projects:

- (Meta-)Studies about data science methodologies (DSM)
- (Meta-)Studies about project management including challenges, success factors and project requirements (PM)
- Case studies (CS)

Our research question is posed accordingly:

- **RQ** Which data science project characteristics are described in literature? ‘

To answer our **RQ** we proceed as follows: We obtain keywords via an initial set of publications and use them to form queries for an automated literature search. We validate and optimize these queries based on a second set of relevant publications. Finally, we query literature databases with the optimized set of queries.

Initial Set of Publications

To obtain relevant publications we follow the same procedure as described in [128]. We start with an initial set of publications for the aforementioned categories: DSM, PM and CS. We use this initial set and publications which are referenced in them to find relevant keywords. Our initial set of publications is listed in Table 3.1. We use this set also to validate that our automated search is working properly.

Data Collection

We perform an automated database search in several literature databases common for computer and data science. We base the selection of the sources on common sources taken in similar studies. Additionally, we use arXiv and Google Scholar to enable cross-database searches and to include recent

	Query Publication Set
DSM	[88], [2], [104], [87], [130]
CS	[131], [132], [133]
PM	[134], [135], [136] , [137]

Table 3.1: Our initial set of publications

	Search string
S1	(data science **and** methodology)
S2	(crisp-dm **and** methodology **and** data science)
S3	(data science **and** methodology **and** project)
S4	(crisp-dm)
S5	(requirements engineering **and** data science)
S6	(data science **and** project **and** characteristics)
S7	(data science **and** case study **and** project)
S8	(data science **and** methodology **and** adaption)
S9	(data science **and** methodology **and** selection)
S10	(data science **and** success factors)

Table 3.2: Search strings used for querying the literature databases

relevant preprints. Based on an initial set of relevant articles we identify keywords and formulate queries for our automated search. The databases we choose for the search are: ACM Digital Library, IEEE Xplore, Elsevier Scopus, arXiv and Google Scholar. If available, we collect all of the following information about each publication: authors, title, year, abstract and citation count.

Query Definitions

Our queries are composed of keywords that we extract from the initial set of relevant publications. We refine the queries by validating against a held-out set of publications which are relevant but not part of the query publications used for constructing the queries. We select an initial set of keywords for all of the different types of publications we are looking for as listed in Table 3.1. Our final set of queries is listed in Table 3.2. ”OR” statements are used to extend the queries with related terms. The related term sets are listed in Table 3.3.

We limit the results for each query to the top 160 search results per database-query pair. This gives us an initial set of 6,493 publications across all five sources. The abstracts and titles of these publications are screened to determine the relevance of these publications. Two experts vote on the relevance of each publication. If both experts agree that a publication is relevant or irrelevant they are included or respectively excluded for further analysis. In case of a conflicting decision by the two experts, a third expert breaks the tie. The publications which remain after this filtering step are

	Related Terms
Set 1:	(data science **or** big data **or** data analytics **or** artificial intelligence **or** AI **or** data mining **or** machine learning)
Set 2:	(methodology **or** process model **or** team process **or** life cycle)
Set 3:	(characteristics **or** factors **or** criteria **or** challenges **or** success factors)
Set 4:	(tailoring **or** adjustment **or** adaption **or** customisation **or** customization)
Set 5:	(crisp-dm **or** crisp dm **or** asum-dm **or** tdsp **or** domino ds **or** ramsys **or** ai ops)

Table 3.3: Related terms which are used in disjunctions in the search queries.

more thoroughly investigated by screening their full-text. Papers which are still considered relevant after full-text screening form the final ”relevant publication” set. From these publications we extract requirements, success factors and challenges in data science. The complete process is depicted in Figure 3.3.

Result statistics

In total we retrieved 6.484 publications with our queries. 171 candidate publications remained after title and abstract screening, expert voting, filtering and deduplication. After full-text screening 62 publications remain which contain relevant success factors, challenges and requirements for data science projects. 9 of these publications are surveys. However, we disregard [138] as it is essentially identical to [139] with only minor adjustments such as reordering of the characteristics. The surveys contain 515 characteristics, while the remaining publications contain 836 characteristics, yielding a total of 1.351 characteristics, i.e. success factors, challenges and requirements.

3.2.2 Review of Foundation Model Literature

To obtain relevant literature for NLU specific project characteristics we follow a two-fold approach. First, we gather survey publications which describe the technology behind foundation models and examine them as well as the publications they reference. Next, we study the website ”paperswithcode” which lists the current best methods for various benchmarks in NLU. We summarize all of our findings in [12] and scan the relevant publications for project characteristics most relevant to NLU projects.

Research Question

Our research question is the following:

- **RQ** Are there any characteristics specific to NLU methods which complement or alter the characteristics from general data science projects? ‘

Result Statistics

We analyzed a total of 897 publications to get a comprehensive understanding of foundation models, especially in the context of NLU. We discovered 46 characteristics that are of particular interest for projects using foundation models.

3.3 A Catalog of NLU Project Characteristics

It is obviously unfeasible to consider 1,396 project characteristics when managing projects. Therefore, we aggregate the characteristics. Firstly, many of the characteristics are duplicates. We summarize such characteristics under one name. Further, most of the characteristics belong to the same overarching categories. Hence, we structure the characteristics into categories and subcategories on up to six different levels. Figure 3.4 shows the whole tree of characteristics, with the leaf nodes representing the distinct characteristics, while Figure 3.5 shows the tree without the leaf nodes. By reducing the tree to 163 aggregate categories, i.e., the characteristics without the leaf nodes, and structuring them hierarchically we reduce its complexity.

We propose structuring the characteristics into four main categories:

1. **Artifact/Asset Management**: Artifacts and assets describe virtual and physical objects which are either given to a project as input, used within the project or produced as an output. They comprise various things such as data, machine learning models, code or documents. We place characteristics in this category if they can be linked to such artifacts but not if they describe processing these artifacts. The four second-level subcategories of this category are ”Data”, ”Model”, ”Knowledge and Information Management” and ”Software/Code”.

2. **Management & Governance**: This category comprises characteristics specific to team management, characteristics which describe the management and structure of organizations, policies and procedures which need to be established as well as resources which should be available for successful projects. The four second-level subcategories of this category are ”Team Management”, ”Organization”, ”Resources” and ”Policies and Procedures”.

3. **Process**: This category contains characteristics which are specific to project processes. Characteristics are assigned to this category whenever they describe activities in the project design, development, deployment and operations phases or project management. This category also contains four subcategories, namely ”Project Design”, ”Development”, ”Deployment and Operation” as well as ”Project Management”.

4. **Technology & Infrastructure**: Characteristics are placed in this category when they address the technologies used or required in projects and the project infrastructure. This category comprises six second-level subcategories: ”Explainability Tools”, ”Artifact Management Tools”, ”Data Tools”, ”Analytics Tools”, ”Visualization Tools” and ”Infrastructure”.

The first three levels of our categories are depicted in Table 3.4. Notice that the four top-level categories are generalizations of the four components of methodologies we identified in Chapter 2. The ”artifacts” component is included in the ”artifact/asset management” category, ”process” in ”lifecycle, phases and process”, ”tools and technologies” in ”technology and infrastructure” and ”roles and responsibilities” is part of the ”management & governance” category.

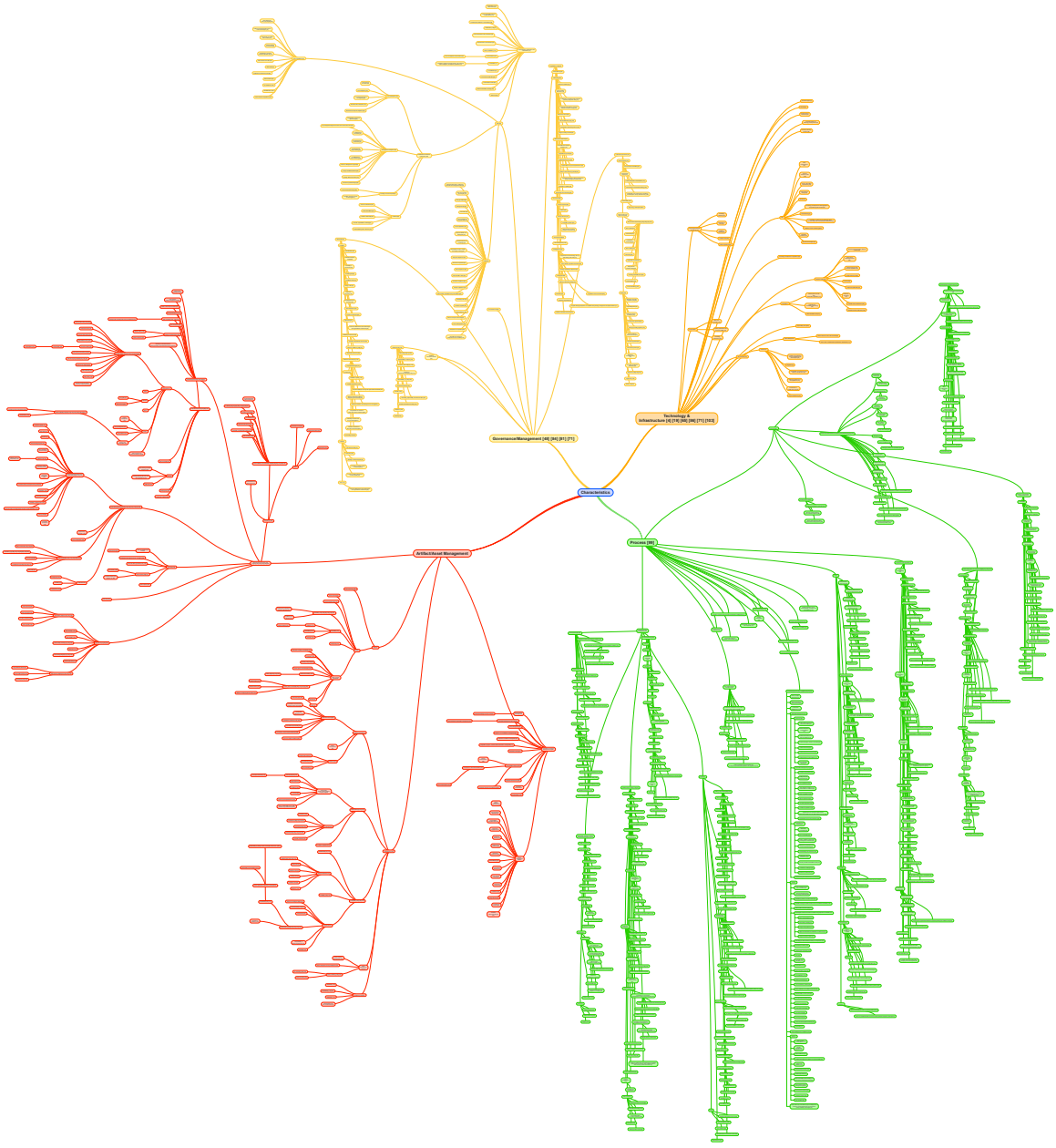

Figure 3.4: The complete catalog of characteristics, including all individual mentions of success factors, challenges and requirements.

Figure 3.5: The hierarchical catalog of characteristics except for the leaf nodes.

We find that the categories of characteristics do not simply arrange in a tree form with no cross-connections between nodes. They rather form a matrix-like structure with categories such as ”data” spanning across nodes. ”Data” is clearly a project artifact and belongs to the category of ”Artifact/Asset Management”. However, data science projects also revolve around processes which define how to handle data, hence we also assign data related processes to the ”Process” Category. Another cross-category topic is ”trustworthiness” which affects every category, in particular the (sub-)categories ”Process”, ”Policies and Procedures”, ”Data” and ”Model” as well as the choice of ”Technologies & Infrastructure” used in projects. We provide a more detailed description of the catalog in Subsection 3.3.2.

3.3.1 Comparison to Eight Meta-Studies

Naturally, we are not the first to propose categories for such characteristics. Our literature study contains eight survey publications which also propose their own categorization of project characteristics. We want to briefly present these surveys and compare their structure to ours.

Study on Requirements Engineering by Villamizar et al.

Villamizar et al. argue that requirements engineering is still an underdeveloped activity when developing machine learning enabled systems [140]. They perform a literature review and identify five main perspectives, i.e., categories in our case, on machine learning which they deem relevant and concerns, i.e., characteristics, which are relevant for a perspective. The ”ML Objective” category centers around the evaluation of ML-enabled systems from technical as well as business perspectives. The ”User Experience” category lists characteristics centered around the user experience when interacting with an ML-enabled system. The ”Infrastructure” category describes characteristics of the system in which the ML components are integrated. The ”Model” category describes characteristics of machine learning models ranging from the model size to selection of an appropriate algorithm. Finally, the ”Data” category lists characteristics of data as well as characteristics of data handling. Figure 3.6 depicts the mapping of the categories by Villamizar et al. to our categories. The ”Data” and ”Model” categories mainly map into our ”Artifact/Asset Management” category. However, Villamizar et al. also describe characteristics of the processes for handling data and models, which maps into our ”Process” category. The ”User Experience” category completely maps into our ”Process” category and belongs to our ”Development” subcategory. We argue that ”User Experience” design is part of the application design process. Finally, the ”ML Objective” design describes characteristics which fit into our ”Artifact/Asset Management” and ”Process” categories, since we again split the process of evaluating machine learning models from the characteristics of the model artifact, such as model performance. Since the survey focused on requirements engineering, it is not a surprise that there are no characteristics belonging to the ”Management/Governance” category. Nonetheless, the absence of characteristics regarding the operation and deployment of models is unexpected, given their crucial role in the development of machine learning enabled systems.

Study on Success Factors by Soukaina et al.

Soukaina et al. present a study of success factors in big data projects [105]. Via literature search they obtained publications discussing success factors and categorized them. They also present a

Top-Level	Sub-Level 1	Sub-Level 2
Artifact/Asset Management	Model	Complexity
		Reusability
		Performance
		Model/Algorithm Selection Criteria
		Model Management
	Data	Big Data V's
		Types
		Quality
		Sources
		Data Management
	Knowledge and Information Management	Documentation
		Experiment Management
		Project Management Artifacts
	Software/Code	Quality
		Code Versioning
Management/Governance	Policies and Procedures	Societal Impacts
		Ethics
		Legal
	Team Management	Skills
		Communication, Collaboration and Coordination
		Stakeholder Involvement
		Team Organization
	Organization	Culture
		Strategy/Vision
		Maturity and Readiness
		Environment
	Resources	Hardware
		Budget
		Time
		Personnel
Technology & Infrastructure	Explainability Tools	
	Infrastructure Tools	
	Artifact Management Tools	
	Data Tools	
	Analytics Tools	
	Visualization Tools	
Process	Project Design	Scope Definition
		Research
		Acquisition
		Concept Design
		Mutual Understanding
		Requirements Engineering
	Development	Data Processing
		Application Development
		Model Development
	Deployment and Operations	Model Deployment and Operation
		System Deployment and Operation
	Project Management	Planning
		Agile Approach
		Stakeholder Management

Table 3.4: The first three levels of our catalog.

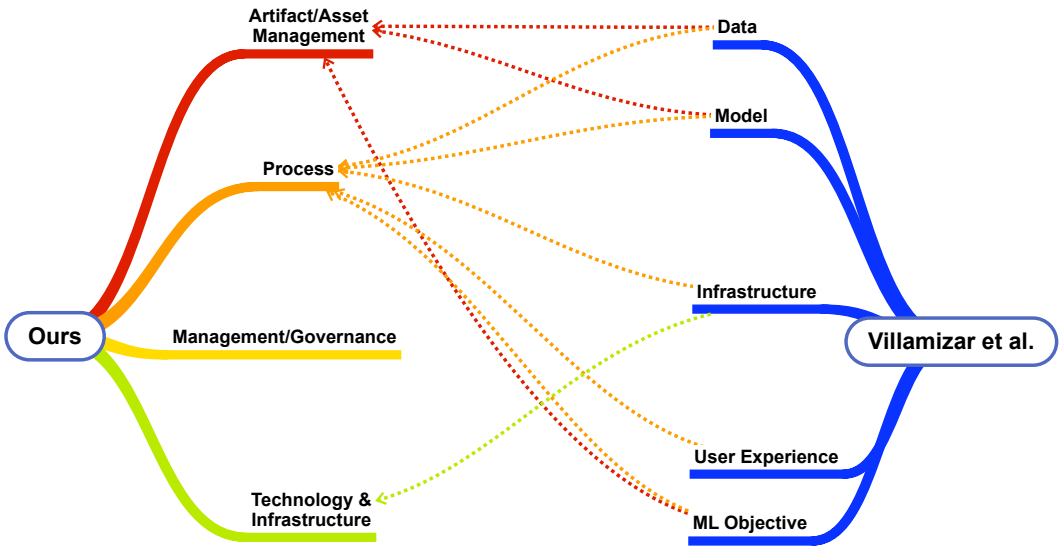

Figure 3.6: Comparison of the catalog by Villamizar et al. with our catalog.

discussion of similarities and divergences to previous publications about success factors. They propose six categories: ”Governance”, ”Process”, ”Technologies”, ”Team”, ”Data” and ”Third Party Management”. Their ”Governance” category discusses characteristics of organizations, management and strategy, however they also assign ”project feasibility studies” to this category. Their ”Process” category describes categories which are mostly linked to project management activities and the overall design of a data science process, e.g. choosing an ”iterative development approach”. The ”Technologies” category contains characteristics about tools to be used as well as characteristics describing the application that is to be developed and the infrastructure in which it will be integrated. Their ”Team” category focuses on staffing and stakeholder management. The ”Data” category lists characteristics about data and their sources as well as the data processing process. Finally, the ”Third Party Management” category lists characteristics of project partners. Figure 3.7 depicts how their categories align with ours. Generally, their study shows a broad focus, aligning well with our structure and covering subparts of all of our categories, however in lesser detail. As with the previous study the ”Data” category splits into our ”Artifact/Asset Management” category as well as our ”Process” category. Their ”Process” category completely maps into our ”Process” category and is best represented by our ”Project Management” subcategory. The ”Technologies” category they propose best fits into our ”Technology & Infrastructure” category as well as our ”Process” category because relevant characteristics for application development are also listed in it. Their ”Team”, ”Third Party Management” and ”Governance” categories all belong into our ”Management/Governance” category as well as our ”Process” category.

Study on Challenges by Martinez et al.

Martinez et al. present challenges in data science projects and, in addition, performed a survey with 237 professionals to gain insights into data science project success factors [141]. They propose

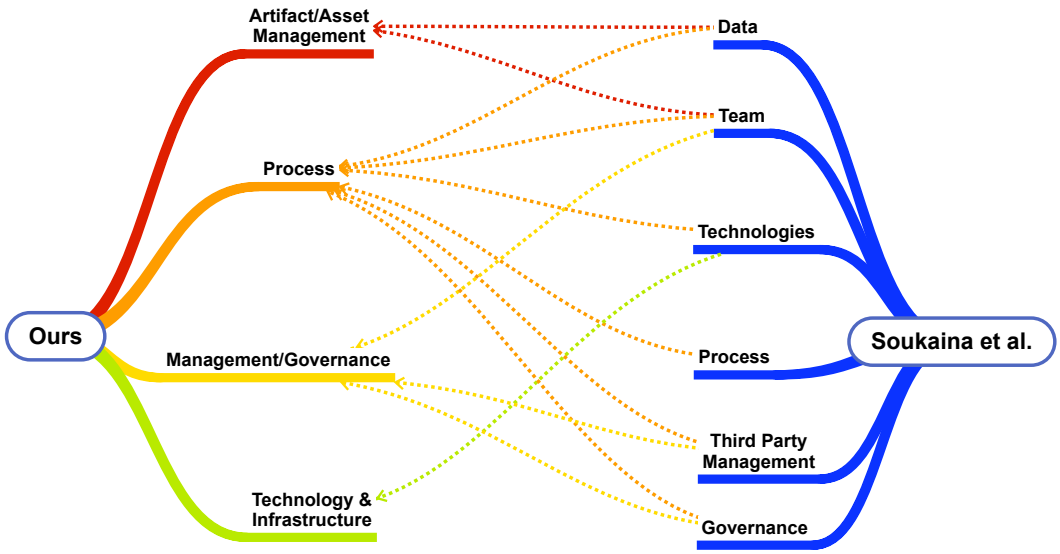

Figure 3.7: Comparison of the catalog by Soukaina et al. with our catalog.

three main categories of characteristics, namely ”Team Management”, ”Project Management” and ”Data & Information Management”. Their ”Team Management” category focuses on characteristics about staffing, collaboration, communication and coordination. This aligns very well with our ”Team Management” subcategory in the ”Management & Governance” category. Their ”Project Management” category presents challenges in the data science process as well as business understanding. Hence, it aligns well with our ”Project Management” and ”Project Design” subcategories in the ”Process” category. Their ”Data & Information Management” category describes characteristics of data and information artifacts and best belongs to our ”Artifact/Asset Management” category, apart from the ”investment in IT infrastructure” characteristic which would map into our ”Infrastructure” subcategory in the ”Technology & Infrastructure” category. The alignment of both categorizations is displayed in Figure 3.8. Their study covers only few of our subcategories.

Study on Machine Learning Software Engineering Challenges by Lwakatare et al.

Lwakatare et al. present a taxonomy of software engineering challenges for machine learning systems [142]. They analyzed case studies of six different companies across domains to identify the software engineering challenges which they are facing when building machine learning systems. Their taxonomy has two dimensions. One addresses the maturity of a project, i.e. whether the project is in a prototyping phase or already in deployment, while the other one addresses specific activities in the project. The categories which they propose along the second dimension are: ”Assemble Dataset”, ”Create Model”, ”Train and Evaluate Model” and ”Deploy Model”. Their ”Assemble Dataset” category combines characteristics from our ”Project Design” subcategory such as ”issues with specifying desired outcome”, our ”Data” subcategory such as ”Imbalanced training set” and our ”Development” subcategory and, more specifically, the activities centered around data processing, which includes a correspondingly named characteristic group ”Assemble dataset”. Their ”Create Model” category can

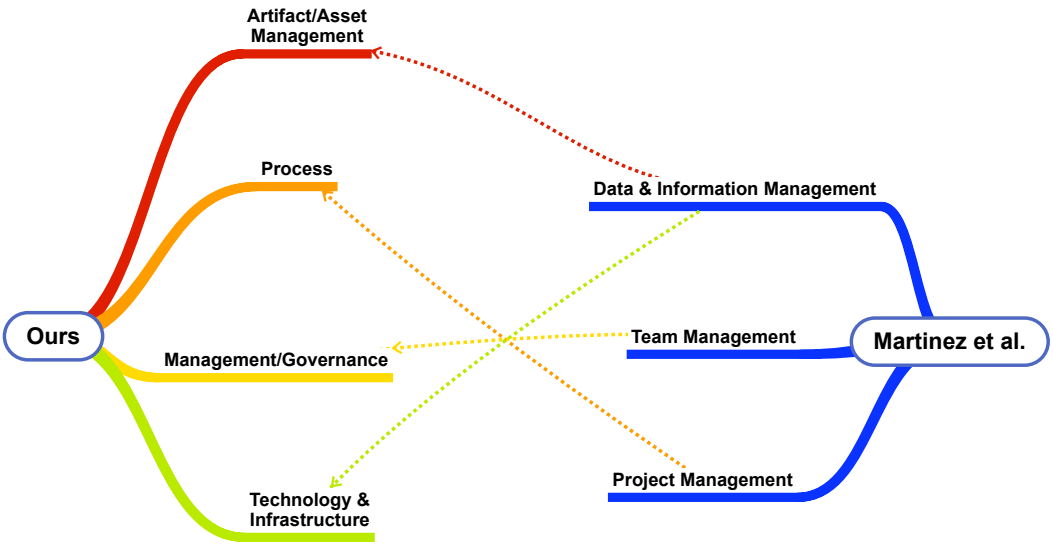

Figure 3.8: Comparison of the catalog by Martinez et al. with our catalog.

be split into characteristics concerning the ”Data” artifact as well as the process of developing a model, listed under our ”Model Development” category below ”Development” and ”Process”. The ”Train and Evaluate Model” category also completely fits into our ”Model Development” subcategory. Finally, the ”Deploy Model” category is mapped to our ”Model Deployment and Operation” subcategory under ”Deployment and Operations” and ”Process”. The mapping of their categories to our top-level categories is depicted in Figure 3.9.

Study on Success Factors focused on Ethical Principles by Miller

Miller uses a systematic literature review to find success factors of AI projects with an emphasis on ethical principles and moral decision-making with algorithms [139]. She identified 17 categories of characteristics on up to two levels. Her top-level categories are ”Project Governance”, ”Product Quality”, ”Usage Qualities” and ”Benefits and Protection”. The ”Project Governance” category comprises the subcategories ”Project Management”, ”Ethical Practices” and ”Investigation”. The subcategories ”Project Management” and ”Investigation” contain characteristics discussing project management artifacts fitting in our subcategory ”Project Management Artifacts”. ”Ethical Practices” belong to our subcategory ”Ethics” in the ”Policies and Procedures” subcategory within the ”Management/Governance” category. The ”Product Quality” category contains the subcategories ”Source Data Qualities”, ”Training Data Qualities”, ”Model and Algorithm Qualities”, ”User Interface Qualities”, ”System configuration” and ”Data and Privacy Protections”. It covers characteristics which describe how to assess the quality of ”Data”, ”Model” and ”Software/Code” artifacts, as well as characteristics about ”Project Management Artifacts” and the ”Process” of developing and evaluating models and systems as well as how to cope with data. The ”Usage Qualities” category comprises characteristics about ”System Transparency and Understandability”, ”Usage Controls” and ”Decision Quality”. These are all subcategories which we assign to our ”System Development” subcategory. Lastly, the ”Benefits

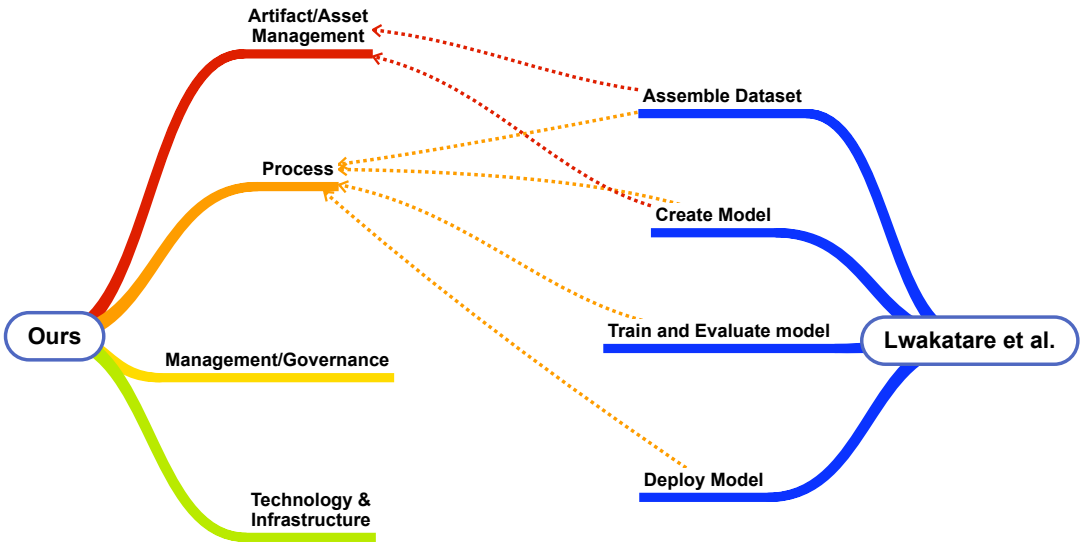

Figure 3.9: Comparison of the catalog by Lwakatare et al. with our catalog.

and Protections" category can be mapped to our "Management/Governance" category and covers characteristics which we would assign to the "Resources" and "Policies and Procedures" subcategories. The list of characteristics suggested by Miller is quiet comprehensive, however it lacks details about categories such as the development process, operation and deployment of systems and models, team management and others. It also completely omits characteristics about "Technology & Infrastructure". The top-level mappings are depicted in Figure 3.10

Study on Challenges in Deploying Machine Learning by Palayes et al

Palayes et al. analyze case studies to understand the challenges in deploying machine learning [143]. They structure the characteristics which they discover according to the data science process but also introduce a category for cross-cutting characteristics. Their main categories are "Data Management", "Model Learning", "Model Verification", "Model Deployment" and "Cross-cutting Aspects". The "Data Management" category contains subcategories describing parts of the "Data Processing" activities such as "Data Collection", "Data Preprocessing", "Data Augmentation" and "Data Analysis". The "Model Learning" category is divided into "Model Selection", "Training" and "Hyper-Parameter Selection". "Model Verification" contains the "Requirement Encoding", "Formal Verification" and "Text-based Verification". Under the "Model Deployment" category they cover "Integration", "Monitoring" and "Updating". All aforementioned categories belong into our "Process" category and partially resemble our subcategories very well. However, their subcategories also mix in characteristics which we map to "Management/Governance", such as the "Mixed teams dynamic", and to the "Artifact/Asset Management" category, such as "Model Complexity". Their "Cross-cutting Aspects" characteristics contain the "Ethics" and "Law" subcategories, which belong to our "Management/Governance" category, "End User's Trust" which belongs to "Management/Governance" and "Application development" under "Process" and "Security" which we split into the "Model"

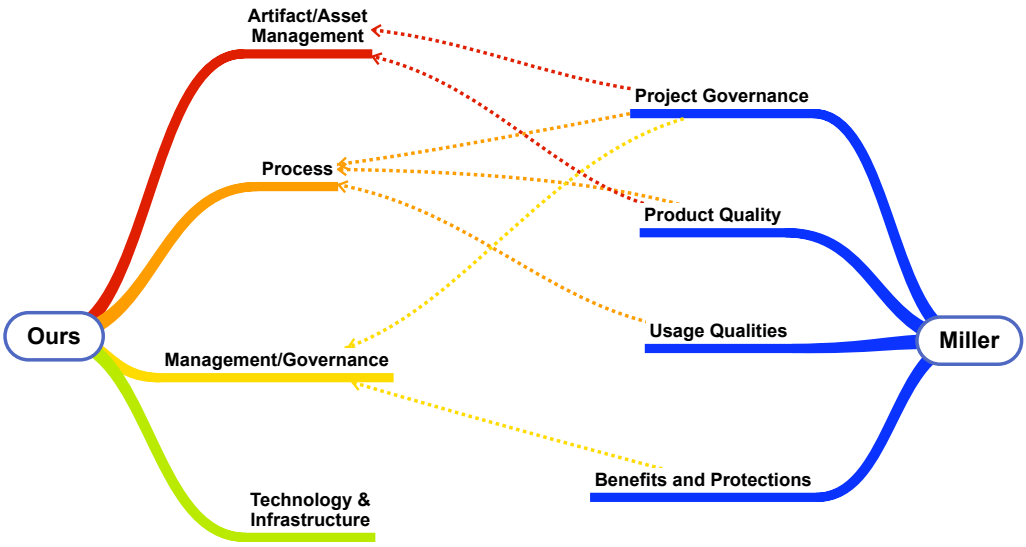

Figure 3.10: Comparison of the catalog by Miller with our catalog.

and ”Data” artifacts in ”Artifact/Asset Management”. The ”Cross-cutting Aspects” category fits our observation that our characteristics do not strictly resemble a tree structure but have matrix-like cross-connections. Figure 3.11 depicts the top-level mapping to our categories.

Study on Big Data Success Factors by Al-Sai et al

Al-Sai et al. study success factors for big data via a systematic literature review [144]. They propose four main categories. The ”Organization” category is well-aligned with our ”Governance/Management” category and especially the ”Organization” subcategory. However, their ”Project Management” characteristic maps into our ”Process” category. The second category they define is ”Data Management” which maps into our ”Data” subcategory in ”Artifact/Asset Management” and the ”Process” category. Their third category is ”People” which maps well to our ”Management/Governance” category and especially the ”Team Management” and ”Organization” subcategories and the ”Project Management” subcategory in the ”Process” category. The ”Technology” category describes characteristics which we map to the ”Application Development” in ”Process” and ”Technology & Infrastructure”. Lastly, their ”Governance” category contains characteristics about the ”Data” artifact, the ”Data Processing” and ”Application Development” subcategories in ”Process” and the ”Policies and Procedures” subcategory in ”Governance/Management”. Although the category names match our top-level categories very well, Figure 3.12 clearly shows that the characteristics scatter around all of our categories.

Study on AI-Based Systems Software Engineering Challenges by Martínez-Fernândez et al

The last survey we analyzed is by Martínez-Fernândez et al. and focuses on challenges around software engineering for AI-based systems [145]. The focus is clearly visible from the category names alone: ”Software Requirements”, ”Software Design”, ”Software Construction”, ”Software Testing”, ”Software Engineering Process”, ”Software Engineering Models and Methods”, ”Software

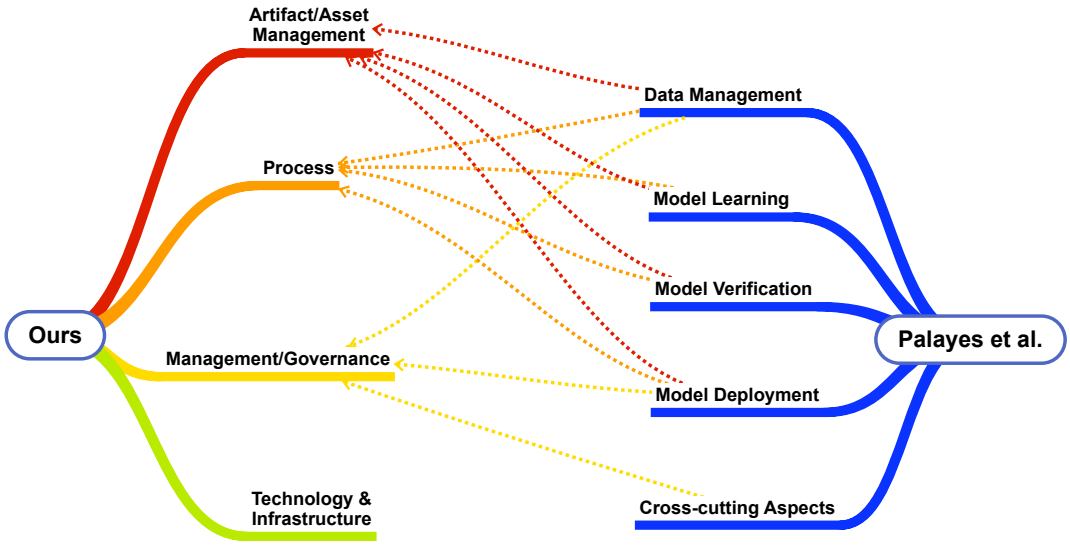

Figure 3.11: Comparison of the catalog by Palayes et al. with our catalog.

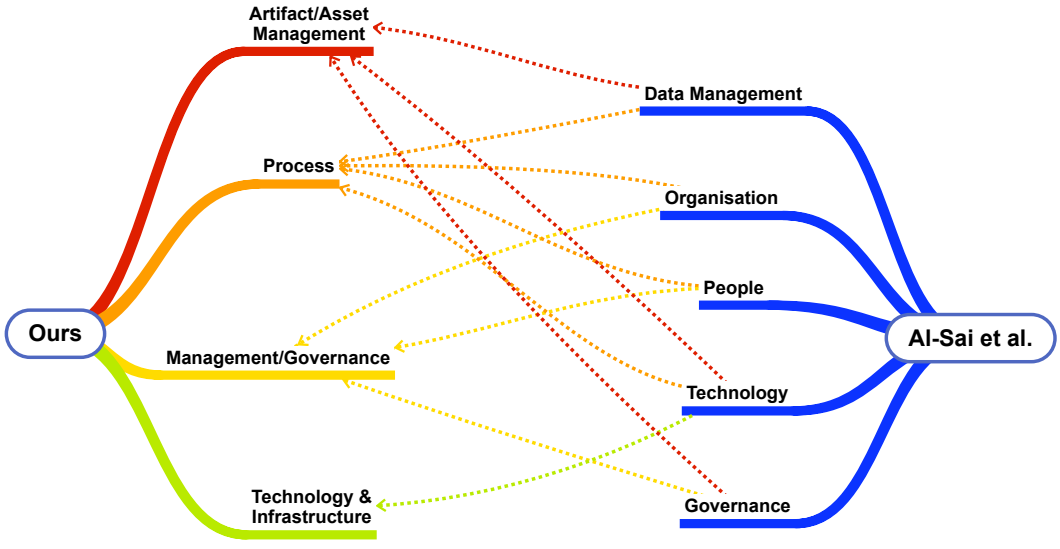

Figure 3.12: Comparison of the study by Al-Sai et al. with our study.

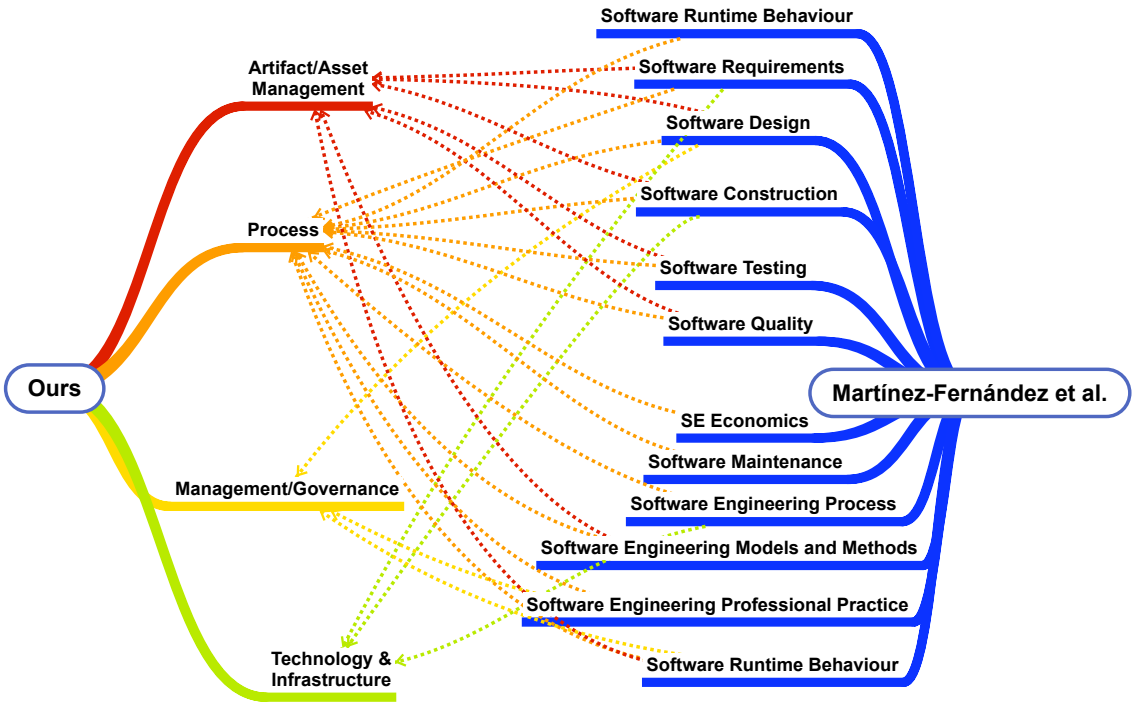

Figure 3.13: Comparison of the study by Martínez-Fernândez et al. with our study.

Quality", "Software Engineering Professional Practice", "Software Runtime Behaviour", "Software Maintenance", "Software Configuration Management" and "SE Economics". Figure 3.13 shows that there are many cross-mappings between our and their categories. However, every single category of the survey also maps to our "Process" category, while there are only few cross-connections to our "Governance/Management" and "Technology & Infrastructure" categories.

Overall Comparison

Overall, the different surveys all cover different parts of our categories. However, our catalog is more than just a superset of their categories, as we also introduce characteristics not covered by any of the other surveys. While most of the surveys cover our "Process" and "Artifact/Asset Management" categories to some extent, fewer address "Management & Governance" characteristics and even fewer consider characteristics about "Technology & Infrastructure". It is also noteworthy that five out of eight surveys have a top-level category related to data and data management. This is not surprising as data is an integral part of data science projects.

3.3.2 Detailed Analysis of the Characteristic Categories

Our analysis shows that most characteristics describe parts of the data science process. The "Process" category contains 566 distinct characteristics. The characteristics in the process category align well with typical phases in software development and data science projects. More interestingly, they

cover the recent topic of machine learning operations and the topic of application development, which is not discussed by most data science methodologies. ”Management and Governance” are the second largest groups with 212 characteristics. The third largest group is the group of ”Artifact/Asset Management” with 205 characteristics. Not surprisingly, the ”Model” and ”Data” subcategories are the most frequently mentioned subcategories of this group, as they play a vital role in data science and are distinguishing factors between data science and traditional software development projects. To our surprise, information and knowledge management, software/code management and especially project management artifacts are underrepresented. Finally, the ”Technology and Infrastructure” category is the smallest with only 53 characteristics. We expected a more detailed view on this category based on the recent focus of the community on standardized tools.

We will now provide a deep-dive into the different subcategories. For the sake of readability, we will focus on interesting and surprising findings and emphasize parts which are most relevant for foundation models.

Artifact and Asset Management

Based on the characteristics which we place into this category, we can identify four subcategories: Model, data, information/knowledge, software/code, project management artifacts. The Artifact/Asset Management category with its subcategories is depicted in Figure 3.14. Not surprisingly data is the most referenced artifact, as it obviously plays a crucial role in data science. Similarly, the ”model” artifact contains many references, which is also not surprising, as machine learning models and machine learning in general play a crucial role in data science and foundation model projects.

Data: The ”Data” subcategory further splits into the categories ”Big Data V’s”[146–148], ”Types”[149, 150], ”Quality”[48, 88, 134, 139, 141, 144, 151–154], ”Sources” [140, 150] and ”Data Management”[88, 144, 145, 155]. ”Data quality” is mentioned as a key concern by many publications. This is in accordance with the focus of current publications on large language model training, e.g. the GPT-2 and GPT-3 publications describe their data collection and cleaning process in as much or even more detailed than the modeling approaches [69, 156]. Other common characteristics are the ”Big Data V’s”. Volume, variety and velocity are mentioned most frequently. Particular challenges with respect to volume are the lack of data [157] and the need for high volumes of data for some learning algorithms [158]. This characteristic is particularly important for foundation models, e.g. in the context of pre-training large language models. Recent models have been trained on approximately 3 billion documents [159]. In addition it was shown that pre-training language models on more data can be even more beneficial than scaling models to larger sizes [160, 161]. Another related characteristic is the size of single samples e.g. in sub-word tokens, i.e. parts of words into which word tokenizers split words in preprocessing. Due to the quadratic memory complexity with regards to the length of sequences, many transformer-based models are restricted to sequence lengths of 512 tokens [66]. However, there are variations of models which specifically tackle the memory consumption of the attention mechanism to enable processing of longer sequences [162–164]. With regards to data types, the question whether data is structured or unstructured as well as the format in which the data is available, is discussed in detail. While in projects using foundation models most of the data is unstructured, documents can be considered semi-structured as they contain spatial information in addition to text. Sometimes structured data is combined with unstructured data to achieve better results [165]. Often documents not only have mixed structures but naturally also have multiple modalities, as they combine image data with textual data. This affects the choice of model, as some models are

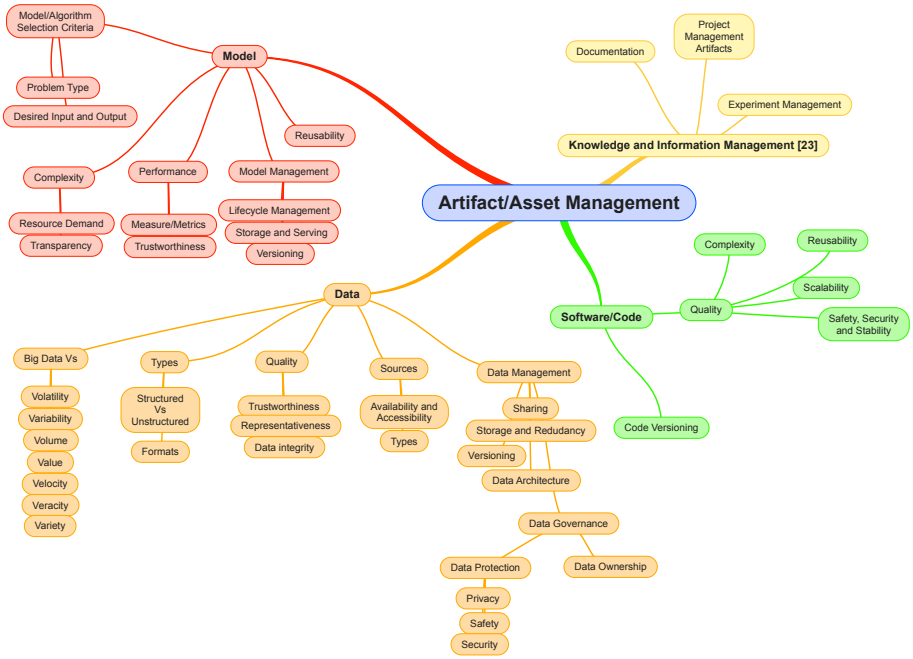

Figure 3.14: The subcategories of the Artifact/Asset Management category. It is evident that most of the characteristics center around the model and data artifacts.

particularly designed to process multiple modalities [166, 167]. Data formats can impact the data preprocessing process, as parsing can become increasingly difficult depending on the format. On the other hand, some formats offer valuable information, such as layout information in documents, which can be used in addition to the plain text they contain. A special attribute of texts is that they are written in various languages. This is another characteristic of data which influences the choice of models, as some models are explicitly trained to tackle multilingual inputs [168]. With respect to data sources, the availability, accessibility and type have a strong influence on the project outcome. Foundation models rely heavily on public data sources to amass a sufficiently large amount of training data [69]. Finally, a lot of characteristics describe data management in projects. These characteristics focus on the topics of data sharing, storage and redundancy, versioning, data architecture and data governance, where the latter can be further split into data protection and ownership. Considering the paradigm of pre-training foundation models and then adapting them to domains and tasks, data management will become even more important, including saving meta-data about the time when the data was obtained, its domain and so on.

Model: The subcategory ”Model” [137, 140, 158] is divided into several categories, namely ”Complexity” [48, 152, 169], ”Reusability”[169], ”Performance”[48, 140, 170], ”Model/Algorithm Selection Criteria”[140], and ”Model Management” [145, 171]. The ”Complexity” category addresses characteristics which describe the ”Resource Demand” and the ”Trustworthiness” of a model with respect to understandability, transparency, interpretability and explainability. The characteristics belonging to ”Resource Demand” describe the size of the model, how scalable it is and the time it takes for training and inference. There is a wide variety of foundation models with different

resource demands. Considerations like available hardware and acceptable training and inference time are affected by these characteristics. While very large models might need multiple GPUs even for inference [172], there is also a trend towards reducing model complexity e.g. via distillation methods [173]. Regarding "Trustworthiness" it is evident, that deep learning models are not easily understood or explained. In contrast, models such as logistic regression offer better explainability with the cost of sacrificing predictive performance. The choice of model leads to a trade-off between performance and explainability. However, there are methods to help end-users understand the output of even large language model, e.g. via visualization of attention-scores [174], via computing multiple gradients for different inputs [175], by making generative models explain their chains of thoughts [176] or by retrieving evidence for generated sequences [177]. The "Performance" category obviously lists characteristics regarding the selection of appropriate measures for assessing the quality of a model. However, especially with regards to certain NLU tasks like text generation, novel metrics such as BERTScore, which are specialized on pre-trained language models, should be considered [178]. The performance of models should further be evaluated with regards to trustworthiness, i.e. characteristics including robustness [48, 145, 158, 179], consistency [139] and uncertainty of models [145, 155]. The "Reusability" of models is especially important in the context of foundation models which embrace transfer learning. Reusability can be achieved in different ways, e.g. by using the same model in different environments [158], modifying an existing model [170] or retraining it [169, 170]. However, especially for foundation models, adaptability plays a crucial role [169]. Foundation models are typically pre-trained on general tasks such as masked language modeling and then adapted via methods such as fine-tuning [66], prompt-tuning [180] or prompting [69]. Given the reusability of foundation models, "Model Management" naturally becomes a consideration [145, 171]. Models should be managed over their complete lifecycle [145], i.e. in the case of foundation models even across projects. They should be stored in an accessible way, making it easy to serve them [171], and versioning them becomes a crucial part [137, 181]. This is also relevant with regards to linking models to the respective datasets they have been trained on or adapted to. Further characteristics for "Model/Algorithm Selection" can be grouped into the subcategories "Problem Type" and "Input and Output". In Chapter 2 we introduced various different types of problems or tasks which can be tackled with NLU models. There are models which were explicitly fine-tuned on each of these tasks. However, large language models show great performance across some of these tasks without explicit fine-tuning. Nevertheless, different models still prove best for different tasks. While decoder-based models may be the best choice for tasks with sequential output, encoder-based models still prove better performance in many tasks which require robust, structured output [182].

Software/Code: "Software/Code" represents another artifact in data science and foundation projects. Machine learning models in production environments are often just small parts of a whole application [99]. Hence, the success of a machine-learning based application is also determined by the software which embeds the models. The "Quality" of the software splits into various categories of characteristics: "Complexity" [137, 151, 169], "Reusability"[183], "Scalability"[150, 155, 169, 181, 184, 185] and "Safety, Security and Stability" [140, 143, 146, 150, 151, 154, 169, 179, 184, 186–189]. Reduced complexity and higher scalability are beneficial also in terms of reusability. Reusability of software becomes easier when embedding foundation models. A huge benefit of these models is that they are built with similar architectures, mostly based on the transformer architecture, allowing for an easy exchange of models with the same input and output types and encouraging building software that relies on standard frameworks and libraries such as the Huggingface library [84]. The subcategory "Safety, Security and Stability" contains general characteristics which should

be considered for any software. However, novel security treats such as ”Model Stealing” [143] and ”Model Inversion” [143] need to be considered. This becomes even more important if these models encode sensitive information. Finally, ”Code Versioning” [188] should be considered for professional software development.

Knowledge and Information Management: The ”Knowledge and Information Management” [141] category includes characteristics which consider artifacts that help to document, share or specify knowledge and information. Important characteristics belonging to this category are e.g. to acquire knowledge [110], to reuse it [157], to harness it for future work [141] and to share it [88]. Subcategories are ”Experiment Management” [134, 137, 171], ”Documentation” [88, 147, 185] and ”Project Management Artifacts” [139]. ”Experiment Management” is crucial during the development of machine learning models to keep track of promising hyper-parameter configurations [171]. However, it remains vital when updating a model already in production. ”Documentation” not only serves the users of a system [185] but is also essential for internal purposes [185], e.g., to draw lessons learned from project efforts to improve the efficiency of a team or to maintain software. Finally, the usage of model cards can help to provide transparency over the properties and training setting of models [190] and is already in use on the Huggingface platform [84]. Lastly, various characteristics regarding ”Project Management Artifacts” can be drawn from [139]. These characteristics describe artifacts ranging from auditing to technical deployment records and responsibility alignment matrices.

Management/Governance

The ”Management/Governance” category contains characteristics considering the project team, the overarching organization and the project partner/customer. It can be further structured into the ”Team Management”, ”Resources”, ”Organization” and ”Policies and Procedures” subcategories. Recent studies call for a more holistic approach to data science projects and argue that not addressing this is a shortcoming of many methodologies [88]. This aligns with our observation that we can find many characteristics describing challenges in this category. The whole subcategory without leaf nodes can be seen in Figure 3.15

Team Management: The ”Team Management” subcategory contains characteristics which can be further arranged into the subcategories ”Team Organization”, ”Stakeholder Involvement”, ”Communication, Coordination and Collaboration” and ”Skills”. The ”Team Organization” characteristics describe staffing. Team Roles should be clear and teams staffed accordingly [88, 141, 191]. Further, teams benefit from multi-disciplinary, diverse team members [105, 141, 144, 192]. Teams should be balanced [105, 193] and should not rely on lead data scientists [141]. They should also account for fluctuation of team members and reliances outside of the team [194, 195]. ”Communication, Coordination and Collaboration” summarizes characteristics which describe inter-team and intra-team concerns. The collaboration, coordination and communication across teams [110, 141], especially with IT personnel [88, 105, 144, 193], is frequently mentioned. Communication is a key concern [196] and should be transparent [88], focused [193] and proactive [141]. Further, information exchange should be fostered [181]. The ”Stakeholder Involvement” subcategory encompasses characteristics regarding the involvement of various stakeholders outside of the development team. It is further partitioned into subcategories revolving around different stakeholders: ”Domain Expert”, ”Customer”, ”User” and ”Management”. The involvement and commitment of various levels of management is considered a key success factor by many studies [105, 144, 151, 193]. Customers should be committed to a project [193], be consulted and included for decision making [151, 187]. User involvement should

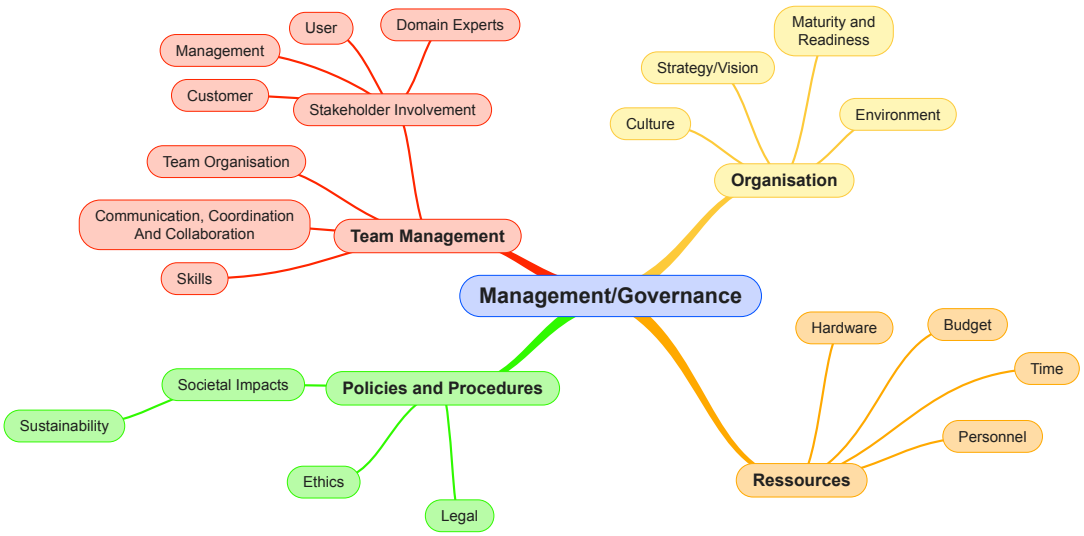

Figure 3.15: The subcategories of the ”Management/Governance” category. It is evident that most of the characteristics center around the model and data artifacts.

already start in early stages of projects [157] and should last throughout the project [143, 144, 151] with the final goal of achieving user acceptance [144, 151]. Lastly, domain experts should be involved, which is a concern due to their availability [158]. They provide valuable domain knowledge which many projects lack [137]. In the context of NLU projects especially labeling data is a task for which domain expertise is required, particularly if texts from specific domains need to be annotated. The ”Skills” category lists characteristics describing the various skills needed in data science projects. They range from management [144] to machine learning skills [188]. Some characteristics describe the need for skill gap analysis in teams [105, 144] and the need to educate teams to gain new skills [141, 144, 155]. In particular deep learning skills play an important role [158]. We argue that, with the rise of foundation models, this will be amplified. More so, machine learning engineering will become more important, especially the skill of scaling up training processes to distribute training over large amounts of hardware and building systems and guardrails around large language models to control their input and output. Further, novel skills such as prompt engineering will grow in significance.

Organization: The subcategory ”Organization” includes characteristics describing not only the project team but the organization in which the team is embedded. The characteristics form further subcategories such as ”Culture”, ”Strategy/Vision”, ”Maturity and Readiness” and ”Environment”. The characteristics under ”Culture” describe the data-driven mindset or culture that companies should have to successfully conduct data science projects [105, 144, 149, 193]. Further, the culture should foster collaboration and sharing [105, 155]. ”Strategy/Vision” includes characteristics which link the success of companies to defining a strategy [149, 151] and aligning actions, goals and projects with the strategy [105, 110, 144]. With respect to foundation models, such strategies may include how data should be acquired and managed, as well as how foundation models should be trained, adapted and managed. The ”Maturity and Readiness” category contains characteristics which describe the readiness of organizations to perform data science projects. Characteristics include effective processes

[144] and readiness for innovation and changes [105]. The ”Environment” characteristic splits further into the actual ”Work Environment” [193] as well as the ”Industry” [144] in which an organization and the competition operate [141] as well as other market developments [105].

Resources: The ”Resources” [88] category covers characteristics about resources which are available to an organization and need to be considered in projects. Naturally, organizations require adequate resources to execute projects [134, 137, 158, 184]. These resources can concern ”Personnel”, ”Budget”, ”Time” and ”Hardware”. The ”Personnel” subcategory contains characteristics such as the scarcity of skilled IT personnel, data science personnel and domain experts [137, 158]. Krasteva et al. further describe the reassignment of personnel during projects [146] as a challenge. The characteristics in the ”Budget” subcategory are mainly concerned with financial aspects [139, 145, 151]. However, they also take into consideration the energy cost of training models [139]. This is a large concern with respect to training of foundation models, which sometimes use thousands of GPUs for a span of months. Obviously, ”Time” is also a relevant characteristic and a constraint with regards to project management [144, 170]. Finally, hardware plays an important role [88, 155]. Characteristics in this subcategory are concerned with storage [150, 170], network capacity [186] and processing power [147]. Pre-training foundation models requires immense processing power and, depending on the size of the model, even thousands of GPUs [69]. Thus, it is important to distinguish whether a project will benefit from pre-training a model from scratch or whether a team should take a pre-trained model and only adapt it to a task at hand. A crucial consideration here should also be the available hardware in the operating environment and requirements such as processing time. It should be noted again that even for inference, the largest models need multiple state-of-the-art A100 GPUs [172].

Policies and Procedures The ”Policies and Procedures” [197] category can be further divided into ”Societal Impacts” [139], ”Ethics” [139, 145, 154, 169, 188, 198] and ”Legal” [134, 198] characteristics. It contains characteristics which concern internal policies of an organization regarding the aforementioned aspects. ”Legal” characteristics are addressing laws [144], regulations [137, 143, 152] and norms [188]. The GDPR is addressed explicitly by two sources [152, 179]. Further characteristics describe that organizations should comply to regulations and laws [139, 186] and explicitly implement safeguards to assure compliance [139, 199]. This topic is of great importance for large language models. At the time of writing, the European Union is explicitly discussing regulation of large language models given recent developments in the field. The characteristics in the ”Ethics” subcategory address the implementation of ethics policies and the according training of employees [139]. Human rights and human well-being should be considered when executing projects [154]. With the advent of foundation models that can generate images in styles of various artists, the topic of authorship has become more prominent [143]. Regarding ethical concerns, Miller suggests implementing an Ombudsman [139]. The last subcategory of ”Policies and Procedures” is ”Societal Impacts”. Characteristics which belong to this subcategory address environmental issues [139], sustainability [170] but also the consequences of automatisation for society [139] and the well-being of society [154]. The impact of foundation models is large on all of these topics. The carbon footprint of a single training run of large language models is immense. For example, for training the largest LaMDA model it was approximately 25.2 tCO2e [159]. Note, that the authors highlight that this is low in contrast to other large language models.

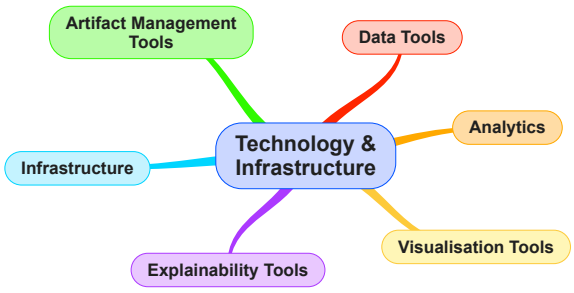

Figure 3.16: The subcategories of the ”Technology & Infrastructure” category.

Technology & Infrastructure

The ”Technology & Infrastructure” category groups characteristics which describe technological concerns as well as concerns about the project infrastructure. The characteristics are further partitioned into the subcategories ”Artifact Management Tools”, ”Data Tools”, ”Analytics Tools”, ”Visualization Tools”, ”Explainability Tools” and ”Infrastructure” as can be seen in Figure 3.16. To our surprise, this is by far the smallest category with only 53 characteristics. Generally, the characteristics of this subcategory call for an adequate selection of technology with high readiness and maturity [144]. Tools and infrastructure should ideally be capable [144] and easy to use [151]. However, their complexity seems to be a prominent issue [145].

Artifact Management Tools: The ”Artifact Management Tools” subcategory contains only two distinct characteristics since some of the characteristics are already assigned to the ”Data Management” and ”Model Management” subcategories in the ”Artifacts/Asset Management” subcategory. However, the characteristics belonging to this subcategory emphasize cross-cutting capabilities. Version control tools should be used to manage code, data and models [151]. Further, Saltz et al. call for continuous integration and deployment, maintenance as well as coordination tools [110].

Data Tools: The ”Data Tools” subcategory is the largest subcategory of ”Technology & Infrastructure” with ten assigned characteristics. An appropriate selection of tools for data storage is mentioned as a concern [140, 144]. Further, connecting to data sources and especially to data streams is mentioned by three publications [140, 142, 144]. Other characteristics discuss the capability of handling big data [31], integration of data from different sources [144] and data documentation [144]. Martinez et al. list the need for data annotation tools [145]. This is particularly important for NLU projects, as many tasks require labelled data, often in specific formats. Named entity recognition, for example, uses the so-called BIO-notation, in which words are either annotated as the beginning or intermediate word of an entity, or with the ”other” category. Tools which automatically create annotations in this format can help experts save time and make labeling more accessible for them.

Analytics Tools: Characteristics in the ”Analytics Tools” subcategory concern the analytical abilities of tools in various stages of data science projects [105, 144, 151]. These tools can support the data discovery [144], data analysis and machine learning [155, 193] or e.g. parameter management for distributed training [200, 201].

Visualization: This subcategory comprises characteristics which concern tools for visualization. Such tools can e.g. be used to visualise data [105, 141] or help with reporting and visualising results

[144].

Infrastructure: This subcategory includes characteristics about the infrastructure of the organization, that is used for projects. Considering the ever-growing hardware demand of models and the rapid developments in processing power, companies should invest into their IT infrastructure [105, 144]. The infrastructure needs to be appropriate for the applications and tools which should run on them [105, 144]. Finally, companies should consider using infrastructure from cloud providers and integrating them with their current tools and technologies [140, 144].

Explainability Tools: This subcategory is concerned with characteristics for explainability tools. Surprisingly, no characteristics for this category were found in our systematic literature review. However, the foundation model literature revealed tools particularly designed to visualize explanations of transformer-based models [174, 202]. Further tools to interpret the results of transformers are "transformers-interpret" [203] and the "Captum" library which works with deep learning models based on pytorch [204]. Next to these tools are also other ways to make systems more explainable by design, however we place them in the "System Design" subcategory in the "Process" category.

Process

The "Process" category [205] is by far the largest category with 566 characteristics arranged in many subcategories. This category includes all characteristics which concern the process of executing data science and, more specifically, foundation model projects. On the second-level, this category is subdivided into the "Project Design", "Development", "Deployment & Operations" and "Project Management" subcategories as shown in Figure 3.17. The "Development" subcategory is further subdivided into three large subcategories, concerning "Model", "Data" and "Application"-related tasks. Having a well-defined process is considered important [105, 141] and its absence is considered a key factor for project failure [48, 110]. Processes should provide guidance and account for project realities [181] without sacrificing flexibility [106]. They preferably take a business-driven approach [105] and implement scientific methods to enable reusability [88, 154, 181]. Further important characteristics related to processes are that adapting them should have a manageable cost and should not introduce too much complexity [105]. They should interleave experimentation and results interpretation [145], embrace a business-driven approach [105] and collaboration [110]. Saltz et al. further suggest that project processes should integrate project, team, data and information management [110].

Project Design: The "Project Design" subcategory includes characteristics describing phases and tasks in the initiation of a project. It is further divided into the "Acquisition", "Scope Definition", "Research", "Concept Design", "Mutual Understanding" and "Requirements Engineering" subcategories. A common theme of characteristics in the "Acquisition" subcategory is dealing with risk and uncertainty [145, 193] related to the impossibility of giving prior guarantees about project success [145, 187]. Further characteristics concern the creation of contracts and the estimation of costs and effort [185]. Projects should ideally yield long-term benefits [145] and should only be executed when needed, to not waste resources [186]. The "Scope Definition" subcategory includes characteristics which describe how to set the scope of a project [137, 185]. Scopes should be well-defined [105, 193] to prevent delivering something other than what the customer needs [141]. Objectives and outcomes should be measurable [105], modest, progressive [105] and also well-defined [193]. The "Research" subcategory was added based on characteristics from our foundation model study. The reusability of large language models, their public accessibility through platforms such as Huggingface [84] and the availability of benchmarks and datasets [206] allow data scientists to not only search for publications

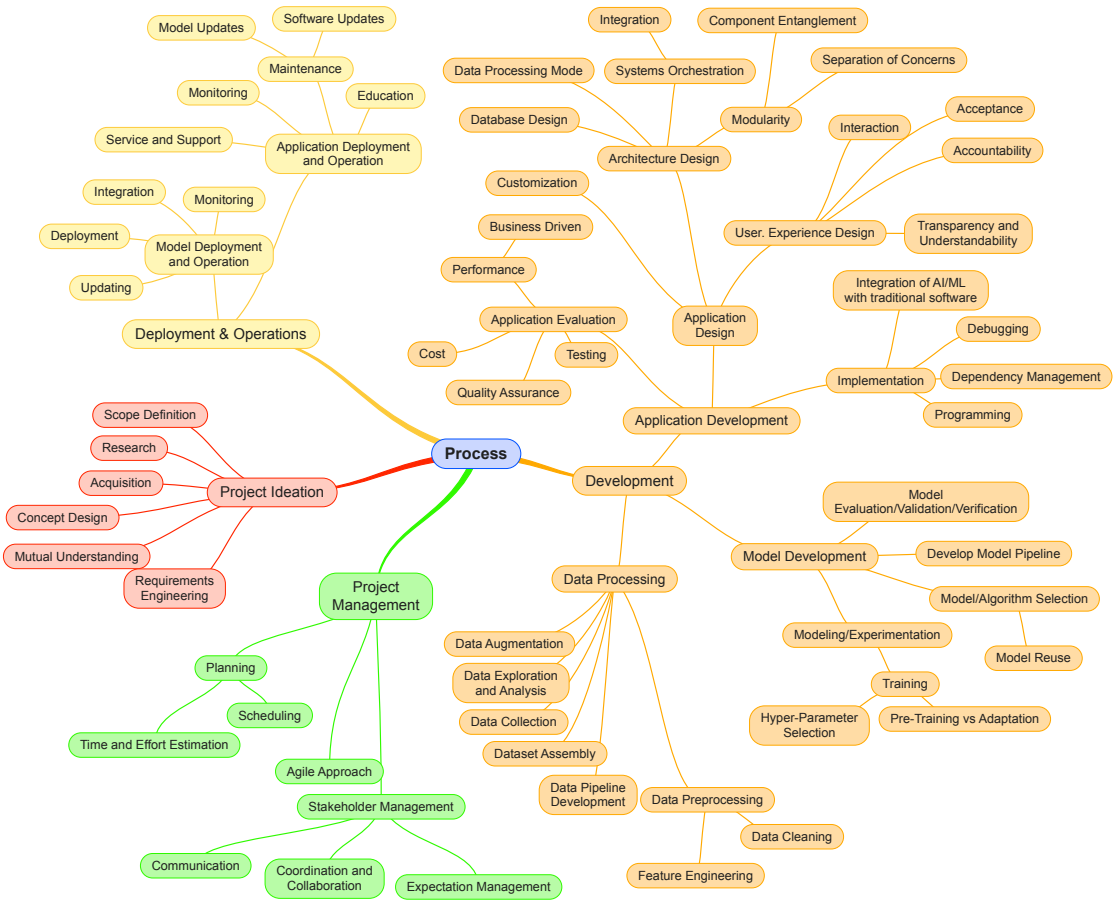

Figure 3.17: The subcategories of the ”Process” category.

discussing similar work, but also for models which might already be suited to solve certain problems or datasets to improve the likelihood of success of projects. We argue that this is a characteristic which has gained a lot more relevance since the introduction of foundation models. The ”Concept Design” subcategory encompasses characteristics which are concerned with designing the approach to be taken to reach the project objectives. A common theme of the characteristics is the translation of business objectives and requirements to data science problems and technologies [141, 145, 185, 207]. Martinez et al. also suggest identifying project risks and potential pitfalls early on [141]. A potential way of mitigating risks is described by characteristics which concern feasibility studies [17, 105] such as preliminary experiments [205], ”Proof of Concept” projects [185] or the development of an MVP solution which tackles a smaller problem [110]. Leidner suggests to sub-sample data for such purposes to make the analysis more feasible with reasonable effort [17]. The ”Requirements Engineering” [185] subcategory lists characteristics regarding project requirements [193, 205]. Requirements can for example be functional [134, 150], non-functional[158, 169, 185, 188], data-driven [205] or domain-specific [150]. A particular challenge of requirements in data science is that they are often more uncertain [205] and ambiguous [150, 195] than in traditional software development, requiring a better understanding [110]. Once understood they need to be mapped to techniques and solutions [145,

208]. Lastly, processes to measure requirements should be implemented [145]. The characteristics in the ”Mutual Understanding” subcategory are concerned with the understanding of the business side and the development side. We chose to name this category ”Mutual Understanding” rather than ”Business Understanding” as there are not only problems understanding the business [145, 187] but also misunderstandings on the business side regarding the technology [131]. Wrong business understanding can lead to data science teams delivering solutions to wrong problems [110] and customers not using the solutions [141]. To avoid such pitfalls, some characteristics describe the need for clear goal and business case specifications [88, 105, 149, 157, 158, 185, 193]. Finally, these goals should also be well-documented [106].

Development: ”Development” is the largest subcategory of the ”Process” category and contains all characteristics describing the main phases and tasks in the execution of data science projects. The subcategory further splits into ”Data Processes”[209], ”Model Processes” and ”Application/System Development Processes”[137]. We choose to describe these subcategories separately to enhance readability.

Data Processes: The ”Data Processes” subcategory splits into further subcategories concerning the lifecycle of data in data science projects. Many characteristics address the collection of data [143, 144, 185, 210]. These characteristics include data discovery [143, 155], data selection [153], quality assessment [185] and reviewing whether the stakeholders have a correct data domain understanding [185]. Miller further adds that projects benefit from documenting the data collection process [139]. Other characteristics describe the assembly of a dataset after collection [142]. The dataset assembly should be according to the specification of the problem formulation and the desired outcome of a project [142]. When assembling datasets, appropriate data architectures need to be chosen [142] and systematic methods for splitting data into training and testing sets should be utilized [145]. The subcategory of ”Data Analysis” [143, 181] contains characteristics such as the exploration and understanding of data [31, 158, 185] and steps such as data profiling [143]. Data analysis alone can lead to project outcomes such as actionable insights [195]. Characteristics in the ”Data Augmentation” [141, 143] category mostly concern labeling of data [158]. Typical challenges are the scarcity of labeled data [142], the cost and difficulty of labeling data [137, 145, 155, 157, 205], the access to experts which are able to label data [143, 157] and the amount of labeled data which is available and required [142, 143]. Data labeling is a typical challenge of projects that deal with unstructured data, e.g. in NLU. Annotating texts for tasks such as named entity recognition is costly and tedious, as they require annotating individual words in texts, sometimes in large amounts of texts, i.e. domain experts need to mark the exact position in a text that corresponds to an entity. A common practice in NLU is the design of annotation guidelines to assure consistency in the labels by different labelers [211]. To avoid high costs, it might be beneficial to utilize structured resources such as ontologies or knowledge graphs to automatically annotate entities [212]. It might also be helpful to use crowdsourcing for labeling if large amounts of text should be annotated [211] or to leverage large language models as annotators [213]. The final subcategory of ”Data Processes” is ”Data Preprocessing”[143]. Characteristics in this subcategory address data preparation [185, 210], cleaning [143, 185] and feature engineering [158, 210]. In NLU one common time-consuming task is the extraction of text from different data formats. Formats like PDF, HTML, and more require dedicated parsing scripts to clean metadata. Foundation models have fundamentally changed certain aspects of ”Data Preprocessing”. Tokenization of texts for instance is reduced to splitting text into common sub-tokens [214]. However, some models still require certain preprocessing steps such as sentence splitting [66]. Feature engineering is less important, since pre-trained language models are essentially feature generators. In contrast to this complexity

reduction, exhaustive data cleaning might be beneficial to reduce bias present in data [159, 215]. To make the whole data process repeatable and reusable, [185] suggest the development of data pipelines.

Model Development: The "Model Development" subcategory contains characteristics concerning the complete model lifecycle in a project. The subcategory can be further divided into "Model/Algorithm Selection" [143, 185], "Modeling/Development/Experimentation" [158], "Model Evaluation/Validation/Verification" [139, 143, 145, 155, 158, 185], "Model Deployment and Operation" [143, 155, 158, 187, 200] and "Develop Model Pipeline". "Model/Algorithm Selection" entails characteristics regarding the choice of a proper model or algorithm to tackle a data science task. John et al. describe the uncertainty in selecting an algorithm and that data scientists are biased towards preferring algorithms which they already have experience with [158]. Various characteristics describe influences on the algorithm selection, such as resource-constraints [143], model complexity, especially of deep learning approaches [158], and the variety of contexts in which a model needs to be applied [112]. The exploration of models should be reproduceable and data scientists have to be aware that research results are not always applicable in practice [145]. However, they might benefit from assessing related work [158]. Important foundation model related characteristics for model selection are the task that needs to be solved, as there are specific architectures or modifications of models which benefit different tasks. As described in the "Model" subcategory of "Artifact/Asset Management", other selection criteria are the language of texts, the size of inputs and the available computational resources. Rather than training models from scratch, model reuse [155, 185] might be a viable option. This is particularly important in projects for which foundation models could be available inside of an organization or on public model hubs. Transfer learning plays a crucial role for foundation models, as pre-trained models can be adapted to various tasks [66]. Other important characteristics are "Domain Adaptation" [145] and various forms of it such as reframing or revisioning [112]. Domain-specific pre-training has proven to be a valuable option for large language models and has shown boosts in performance across tasks [73]. Another form of model reuse is retraining of models [112]. The characteristics assigned to the "Modeling/Development/Experimentation" subcategory are mostly concerned with model training [143]. The modeling process should be repeatable [169], comparable [179] and reproducible [140–142, 158, 169, 179]. Belani et al. suggest working towards a minimal viable model [137]. Characteristics in training concern the computational cost, the environmental impact and privacy concerns [143]. To train models in a privacy aware way, federated learning is suggested by multiple publications [189, 200, 201]. A tedious and resource-heavy task during training is hyper-parameter selection [143, 210]. With foundation models it is rarely the case, that hyper-parameter selection is an issue, as most of the model architecture stays the same during adaptation. However, the choice between pre-training a specific model and adapting a pre-trained model is important in this context. During pre-training, various pre-training tasks can be chosen that depend on the type of input that is expected, e.g. DocFormer for multi-modal documents [166], and the type of tasks that should be performed, e.g. generative tasks for which one would use a next-token prediction tasks [69] versus information extraction where masked-language modeling might be more beneficial due to the inclusion of context from the left and right side of an input [66]. When pre-training or even adapting foundation models, it might become necessary to parallelize the workload over multiple servers [145]. If structured resources such as knowledge graphs are available, it might increase performance to include them in the training process via models which can process text and structured information simultaneously and link them [165]. "Model Evaluation/Validation/Verification" contains characteristics which concern various aspects of evaluation. Appropriate criteria for evaluation should be selected [185]. They should address all model requirements [143] as well as business-driven

concerns [142, 143]. Even if the metrics indicate good performance, there could be a divergence between results in real-time application and test scenarios [205]. To understand problems with the model performance, debugging might be needed [142]. This can be supported via visualization [137]. Model explanations have also been used to debug models, in the context of deep learning models [216] and specifically when using foundation models [217]. Novel NLU applications might also call for novel evaluation methods, such as the "CheckList" approach, to test various general linguistic capabilities of models [218]. Models should not only be assessed with respect to requirements and performance, but also formally, i.e. whether they adhere to regulatory frameworks [143]. Finally, some characteristics address building model pipelines for different parts of the model lifecycle and to reuse code between training and production [141, 201].

Application Development: The subcategory "Application Development" [137] includes characteristics which describe the development process of the application in which the machine learning models are embedded. This includes software development characteristics, but even more so characteristics which highlight the difference between traditional software development and software development in a machine learning context. The characteristics in this subcategory further partition into "Application Design", "User Experience Design", "Implementation" and "Application Evaluation".

Application Design: One major subcategory of "Application Design" is "Architecture Design" [187]. Examples for existing architectures are the Lambda Architecture [200, 201], the Daisy Architecture [201] and the Kappa Architecture [201]. A worst practice example is given by [201] called the "Big Ass Script Architecture". Multiple studies emphasize the benefit of Microservice Architectures [144, 200, 201]. Apart from selecting an application architecture, other characteristics describe the need for designing database structures [208]. Data lakes are mentioned as possible forms in which data could be stored [200, 201]. In addition to storing data, the mode of data processing should be determined. Characteristics in this group are related to the various inputs which an application might need to handle [187], as well as whether data needs to be processed in batches, streams or in mixed forms [149, 150, 184]. A typical characteristic of machine learning based applications is component entanglement [137, 210], i.e. components are highly coupled [205] and errors in one component propagate to others [205]. This leads to the "changing anything changes everything" phenomenon [145], which implies that changes in one part of the software affect all other parts. Other characteristics describe the integration of machine learning models in the software [110]. These characteristics include the separation of concerns and modularization of different machine learning components [170, 185, 200, 201], encapsulating machine learning models withing rule-based safeguards [200] to mitigate model limits [185] and distinguishing business logic from machine learning models [200, 201]. On the contrary, NLU publications have shown that jointly using and even training models with logical layers might lead to performance improvements [23] and avoid propagation errors. Further important characteristics address the orchestration of multiple systems [145] and different ways of integration [136, 144, 155, 185] and compatibility [169, 170]. Lastly, applications might need to be customized [145] to address domain [181, 189] or user-specific requirements [189].

Implementation: The implementation of machine learning based applications shows characteristics which differ from traditional software development and increase the difficulty of creating application grade software [219]. Obviously, the code quality [48, 137, 170] is an important characteristic in programming [200]. However, a particular challenge in programming machine learning based systems is the tendency to use glue code [137, 201, 210, 220] and multiple programming languages [145, 201]. A characteristic particular to machine learning is the integration of machine learning with traditional software [145]. Amershi et al. describe machine learning components as more complex

than traditional software [155]. As in traditional software development, debugging plays an important part [187, 205], albeit with the extra challenge of debugging models which are not easily explainable. Lastly, some characteristics also describe the process of dependency management [137, 220] to handle third-party integration [145] and black-box packages [201].

User Experience Design: Another difference in developing traditional applications compared to machine learning based applications concerns the "User Experience Design" [140, 143]. When using machine learning based systems, user experience is concerned with characteristics such as accountability [140, 154, 221], i.e., whether a user is accountable for a mistake, user interaction, acceptance, transparency and understandability [139]. Regarding interaction, characteristics are the frequency, interactiveness and forcefulness of interactions [140], as well as the choices of intervention which are given to a user [139]. The characteristics describing "User Acceptance" are similar to traditional software, i.e. the software should be easily usable [145, 169, 170], accessible [154] and provide user-centric views [139, 145]. The "transparency and understandability" [139] of applications groups characteristics such as the explanation of failures [145, 187], providing transparency about how the application works [134, 137, 139, 154, 169, 170, 221] and visualization of predictions [140].

Application Evaluation The characteristics in the "application evaluation" category describe how a machine learning-based application is evaluated. This evaluation is in addition to the isolated evaluation of the machine learning models embedded in the application. The characteristics in the "Testing" subcategory [137, 145, 187, 205, 220] describe typical aspects of software testing such as traceability [145], integration tests [222] and user-feedback [137, 151]. However, they also add machine learning specific concerns such as the long duration of experiments [145], the dependency on uncertain models and data [145], the need to also test feature and data processing code [222] and practices such as allowing tests on single instances of data [222]. "Quality Assurance" [145, 185, 187, 205] should be established to measure the application quality [134, 145, 149, 150] with regards to reliability [154, 169, 170], latency [150], consistency [169] and runtime quality [145]. Further, the application should be validated by measuring requirements coverage [145]. The performance of the application should be assessed [146, 150, 169, 183, 184] with appropriate performance measures [137, 179] including metrics for predictive performance [154, 222], hardware-related metrics, such as processing time [152, 222], but also business-driven metrics [142, 143]. Lastly the cost of using an application should be assessed [145, 193] as well as the cost of wrong predictions [145].

Deployment and Operation: The subcategory "Deployment and Operation" is subdivided into "Model Deployment and Operation" [143] and "System Deployment and Operation" [110, 169, 185].

Model Deployment and Operation: The "Model Deployment and Operation" subcategory contains all characteristics regarding the deployment [106], integration [143] and monitoring [143, 149] of models in production. Characteristics concerning deployment address the need to specify a model execution environment [158], to test deployments internally [158] and the benefit of deploying canary models [200, 201], i.e., rolling out models to smaller user bases first to test their issues. Further, the transition of training to production workflows is considered a complex challenge [145]. The "Integration" of models [143] can lead to issues [158] which can for example be caused by software-engineering anti-patterns or mixed teams dynamics between IT and data scientists [143]. Best practices include the reuse of code and models from training [143]. Model "monitoring" [143, 149] contains typical characteristics in machine learning operation such as data, concept and model drifts [142, 143, 145, 158], shifts between the testing and real-world environment [158, 186] and feedback loops [137, 143, 220]. Multiple characteristics describe the need for logging [137, 219, 220]. Finally, characteristics concerning updating models [143] describe different ways of updates such as

continuous improvement and delivery [143], incremental learning [140] and retraining [145].

Application Deployment and Operation: The "Application Deployment and Operation" characteristics are closely related to the "Model Deployment and Operation" subcategory but extend the characteristics with application-specific concerns. Characteristics include the education of users [155] with measures such as onboarding [139, 185], specific user training and disclaiming [185] and monitoring the usage of the application by staff [139]. The application should also include monitoring and logging [139, 145]. Maintenance includes updates to the software and models [155, 185, 187]. However, using machine learning models in an application might lead to different update frequencies compared to regular applications [145, 155, 187]. Best practices in updating the models include validating the model before serving, allowing for quick rollbacks and running old and new models concurrently [222]. Lastly, applications should come with "Service and Support" [151] providing users the possibility to report errors and complaints [139].

Project Management: The "Project Management" subcategory encompasses characteristics concerning typical project management activities and further organize into "Planning", "Stakeholder Management" and "Agile Approach". A typical characteristic for planning data science projects is the necessity to cope with uncertainty [110] of results [145], technology [151] and tasks [209]. This makes it more difficult to establish timelines and deliverables [141] and to generally estimate time and effort [110, 137, 185, 194–196, 209, 220]. Further, this complicates project planning and scheduling [110, 193, 205]. A large subcategory of the "Project Management" characteristics is concerned with "Stakeholder Management" [193]. This includes challenges in coordination and collaboration [105, 141, 144, 195] and gaining a shared understanding [141, 145]. Transparent communication is regarded as a major challenge and key success factor [105, 110, 141, 158, 205]. Best practices include regular checkpoints for synchronization [110] and jointly exploring and discussing difficulties [105], data and initiatives [105] as well as results, especially with end-users [141]. A frequently mentioned characteristic is the difficulty of setting adequate expectations, as they are often too high [88, 141, 145, 158, 185, 193]. Other challenges regarding expectations are the expectation that results taken from big data are always truthful [186] and the problem that it might be hard to deliver results early in the project, as initial iterations might not yield benefits [194]. Lastly, many characteristics concern an agile approach to project management [110]. This includes, taking an iterative approach [110, 145, 185, 193, 205], adapting to changing requirements [31, 141, 193] and flexibility in the execution of tasks, depending on their scope, e.g. handling training and experimentation differently than deployment [144, 210].

3.3.3 Discussion

Our catalog of characteristics covers the categories "Artifact/Asset Management", "Management & Governance", "Process" and "Technology & Infrastructure". It is the most comprehensive overview of data science project characteristics, aggregating 1,396 characteristics, i.e. mentions of success factors, challenges and requirements, into 163 categories and subcategories. When comparing our catalog to existing meta-studies of such characteristics, we find that our catalog is an extended superset of these catalogs. In particular, our catalog is the only one to explicitly cover characteristics that are specific to foundation model projects. Further, we discuss how foundation models impact typical data science project characteristics.

Our categories cover various aspects of data science projects. While the "Process" category focuses on characteristics which are most relevant for the execution of data science projects, the other

categories take a higher-level view and include holistic characteristics such as the overall organization and project-overarching processes. Generally, the catalog can serve as a tool for data science teams that want to understand which characteristics they need to consider. Further, the catalog can help to decide which methodology is the best fit for a data science project, given the relevant characteristics of a project.

3.4 Grounding the Project Characteristics in NLU Projects

Our catalog is based on theoretical insights, drawn from two literature studies. This provides a solid theoretical foundation. However, the theoretical insights should be grounded in practical realities. Therefor we study the projects of an NLU team. As NLU was the first discipline to adopt foundation models on a large scale, this should provide valuable insights into the usefulness of the characteristics in general data science settings, but also settings concerning foundation models. Over the course of the last seven years, the team executed 26 projects in the NLU domain. We were directly involved in 19 projects in the field of natural language understanding. We analyze these projects with respect to our catalog of project characteristics. Further, we interview the project leads of seven other projects, in which we were not directly involved and enrich our analysis with their insights. These projects provide us with practical insights into the challenges, success factors and requirements of NLU projects. Presenting them with all their details would go beyond the scope of this thesis. However, we want to distill and present the characteristics which particularly influenced the outcome of the projects. In the description of our own projects, we restricted ourselves to naming the three most important characteristics. However, the interviewees found it hard to limit themselves to only three characteristics. Therefore, we take into account all of the characteristics which they mentioned.

We interviewed four project leads and asked them eight questions:

- ”What was the project setup?”
- ”What were the goals of the project?”
- ”How did you proceed, starting from the acquisition, ending with the project end?”
- ”Who was involved in the project and in which role?”
- ”What was the outcome of the project?”
- ”What were characteristics which were important for the success of the project?”
- ”Which challenges did you face?”
- ”Which requirements particularly affected the project process and outcome?”

After the interviews concluded, we tagged the transcripts with the appropriate characteristics from our catalog. In an additional step, we confirmed with the interviewees that the tagging corresponds to the actual meaning of their statements.

In the subsequent subsections, a detailed analysis of six out of the 26 case studies is provided. Three of the case studies represent our own projects, the remaining three represent interviews with other project leads. The detailed description of the remaining case studies can be found in Section A.2 in the appendix. At the end of each case study, we highlight the most important characteristics of each

project. For our own projects, we restrict this to the three most relevant characteristics. We did not restrict the interviewees to naming only three characteristics and will hence list all characteristics which they mention. The results of our analysis are presented in Subsection 3.4.7.

3.4.1 Medical Coding

In this project the goal was to build a machine learning model that complements a rule-based system in classifying medical documentation according to ICD (International Statistical Classification of Diseases and Related Health Problems) and OPS (the official German classification for the encoding of operations, procedures and general medical measures) codes. The client was a consultancy in the medical domain.

Project Setting and Problem Definition

The client had already built a software for medical coding and a large ontology as well as a rule-based matcher. They noticed that rule-based matching would not suffice to cover all types of ICD and OPS codes. Since there are thousands of ICD and OPS codes, the client decided to restrict the scope of the project to a subset of possible codes which was particularly difficult for rule-based matching. The data in the project was highly sensitive requiring precautions to preserve the privacy of the patients as well as data security. Further the data needed to be filtered to remove noise such as documents which did not contain relevant information regarding codes and parts of documents which contained no medical information, such as letter heads. Further, there was a need for combining documents to get the full information necessary to derive certain codes. Another challenge was obtaining enough labeled data for the codes. Some diseases and treatments are rare and should still be detected, thus it would be beneficial to train on data from not only one hospital. To tackle this challenge the team developed a federated learning framework, which sends a machine learning model around hospitals in a cyclic fashion to achieve better performance. Critical for this was establishing the right computational infrastructure in the hospitals. Modeling-wise the team started with a low-resource baseline which quickly provided the proof that a model could outperform rule-based engines for certain codes. Next, a pre-trained language model was used and fine-tuned on the task and clearly outperformed the baseline. Finally, a larger performance boost was reached by pre-training a language model on data from multiple clinics and then fine-tuning the model on this task. The pre-trained model was later used for various other tasks such as the detection of excluded diagnosis and procedures, i.e., when the rule-based engine recognized names of diseases and procedures, the model would detect whether they were negated or not. Finally, all of these models were deployed on dedicated servers in hospitals with a single GPU for running them.

Roles involved

On the side of the NLU team a project lead and multiple junior data scientists were involved as well as two MLOps engineers. The project lead coordinated the tasks in the team and communicated with the client. The junior data scientists explored the data and evaluated multiple models for their suitability. The MLOps engineers developed the inference service as well as the federated learning framework. On client side, the CEO was directly involved in the project, as well as a dedicated project lead, the lead of IT and a doctor who served as a domain expert and also overlooked the development of the rule-based engine.

Project Outcome

The project was a success from a machine learning but also from a business perspective. The final solution was deployed in multiple hospitals and led to a market leading performance in medical coding in Germany.

Most important characteristics

Artifact/Asset Management - Data - Data Management - Privacy Due to the sensitivity of the data and the legal regulation, privacy was a large issue, that led to many important decisions in the project: The development of the federated learning framework, the training on hardware that was explicitly bought for the hospitals and the overall project setup were influenced by this in a major way.

Process - Project Ideation - Concept Design Developing a "Minimal Viable Model" was a major driving factor for the project. Although the team had to make compromises with respect to accuracy, the model still performed better than a rule-based engine. It helped the client gain trust in the technology. After initially seeing the potential of the technology, the client was more understanding when it came to multiple iterations of modeling and evaluation without getting a new model.

Process - Modeling/Development/Experimentation - Training - Pre-Training vs Adaptation The decision of pre-training a distinct language model on the clinical data was a huge benefit for the project. This increased the performance over a regular domain language model more than 10% in the most relevant metrics. Further, the model was easily adaptable for different use cases such as the negation detection.

3.4.2 CV-Parsing

In this project the NLU team worked with a large German software company. A branch of the company was involved with matching job listings to appropriate candidates. They used a market-leading software for processing CVs with the goal of extracting relevant information from them to make them more easily searchable. However, the performance of the software was described as subpar. Although the performance was not systematically assessed by the company, the users stated that they spend most of their time correcting wrongly recognized details from the CVs. The goal of this branch was to replace the software with a tailored solution which provides better information extraction results to shorten the time spent on correcting the results.

Problem Definition

The problem in this project was posed as an information extraction task from semi-structured documents. The information extraction task could be further divided into entity recognition (e.g. employer, job title), relation linking (e.g. employer to job title) and classification (grading of language skills). Around 6,000 English and German CVs were available for training, stored as PDFs. These PDFs were not annotated, but a database with the already extracted values was available. Time for manual annotation was scarce, the end-users which were used to correcting wrongly parsed CVs could only annotate 100 CVs with the help of our own annotation tool. This meant that the remaining CVs would have to be annotated programmatically with weak supervision techniques. A couple of meetings were spent to explain how to annotate the CVs and to derive annotation guidelines. These CVs were used as a gold standard for evaluation. The 6,000 CVs were split into training, validation and test sets.

There were two different types of CVs, one type was described as regular CVs while the other type represented freelance CVs. Both types had a lot of variety in their structure, however the freelance CVs were more complicated. Freelancers often come up with their own CV styles, list projects rather than employers and often do not adhere to any regular order in listing information. Their CVs posed a particular challenge for the CV-parsing tool. However, it was not clearly communicated by the customer, that the focus of the project should be on improving the recognition of freelance CVs. The understanding of the NLU team was that regular CVs already pose a problem and freelance CVs are even more challenging.

There were two teams of end users. One end user was specialized on handling standard CVs, the other two end users were focused on freelance CVs. The latter only joined the project in its end phases when it was made clear that they would evaluate the project results mostly with freelance CVs.

The deliverable of the project was a service which took a CV PDF as input, parsed the relevant information and presented the output visually with the relevant information highlighted and linked. The project was set up as a proof-of-concept (PoC).

Roles involved

The project team consisted of four people on the side of the NLU team and five people on the side of the client. The NLU team consisted of a project lead with multiple years of experience, two senior data scientists, a junior data scientist and an MLOps engineer who joined at a later phase of the project. The project lead focused on managing tasks, i.e. the coordination of the project team and communication with the customer side. One of the senior data scientists was particularly involved in helping the client to create annotation guidelines and using an annotation tool. Both senior data scientists performed conceptualizing, data processing, modeling and evaluation tasks as well as application development. The junior data scientist mainly supported with modeling. He joined at the end of the project to help reach the proposed project deadline. He was also involved with the integration of the machine learning models in the final application. The MLOps engineer only joined in the last phases of the project to aid with the integration. The client team consisted of a project lead, a technical lead who was responsible for providing data and assessing the feasibility of integrating the resulting application in the existing system. One end-user joined the project from the beginning and was responsible for data annotation as well as providing feedback for the machine learning results and data understanding. However, this end-user was focused on standard CVs. Two further end-users were introduced in the last phases of the project. It became clear that these end-users were focused on freelance CVs and were responsible for assessing the final quality of the models to give the decision to continue after the PoC.

Project Outcome

From a machine learning perspective, the project was a success, without human-annotated training data, the F1-scores for CV parsing where in the area of 78-90% for the different tasks such as entity recognition, entity linking and information classification. All of this was achieved with only weakly annotated data and tested on gold data. The resulting application was able to successfully load a PDF, recognize information with a good performance and display the information.

However, from a business perspective the project failed. The client decided to not opt for a follow-up project as the quality of recognition degraded on freelance CVs. The end-users rated the quality of previously unseen CVs on a scale of 1 (very good) to 6 (unsatisfying) for standard and freelance CVs.

They came to the overall conclusion that standard CVs were parsed in good (2) quality and freelance CV in acceptable (4) quality. The main issue with freelance CVs was not the quality of the machine learning models but rather the parsing of the CV layout. The preprocessing pipeline often could not determine the right order of objects in the PDFs and would hence pipe information to the machine learning models in the wrong order.

Most important Project Characteristics

Various characteristics influenced the outcome in a positive and negative way. We will focus on three influencing characteristics which we deem as most important retrospectively.

Management/Governance - Stakeholder Involvement - Users: The involvement of the end users who were responsible for freelance CVs in the last phase of the PoC led to significant problems. It was not clear that these end-users would set the scope of the final evaluation on freelance CVs and that this was considered the most pressing issue. Further they had to be onboarded on the project in a late phase causing resources to be drawn from important modeling, evaluation and integration tasks.

Process - Project Ideation - Mutual Understanding: The goal understanding from a customer and business perspective was not sufficient. This is partially related to the former characteristic regarding the involvement of the end users. The scoping of the project from a technical perspective was done based on the assumption that the application should demonstrate CV parsing capabilities with respect to standard CVs and the assumption that freelance CVs would be tackled in a follow-up project.

Process - Data Processing - Data Augmentation: A characteristic that made a significant impact on the machine learning results was data augmentation. The team developed a framework for annotating CVs based on already extracted information. This alone made the project feasible as there were not sufficient resources available to annotate the CVs manually. Considering the effort it took the client team to manually annotate 100 CVs for evaluation, this weak supervision step saved months of effort.

3.4.3 Information Extraction from Cardiology Reports

In this project the client was a private hospital chain in Germany and in particular one of their hospitals, that was specialized on cardiology. The client wanted to automate the process of extracting information about patients and their diagnostic markers from cardiology reports.

Project Setting and Problem Description

Goal of this project was to develop a service which automatically extracts information from cardiology reports. The client had access to a vast number of cardiology reports, however none of them were annotated. The project was initiated by a professor of the hospital who later involved an IT professional as well as an external project lead in the project. However, these stakeholders were not part of the initial ideation phase of the project. It turned out that the IT professional had already been working on a rule-based solution to processing the information from cardiology reports, i.e. for many cardiology reports he had created lists of terms (including misspelled versions) to extract information from the documents. This led to an unusual project constellation, as the IT professional wanted to continue using the existing solution and avoided communication with the NLU team. Further, the project lead on the client side resorted to instructing the NLU team to manage communication and coordination of the activities on the client side. One goal of the project was to deploy the annotation tool of the NLU

team on the premises of the client, so that a domain expert on the client side could begin annotating documents and the NLU team could train named entity recognition and relation extraction models with that data.

Roles involved

On the NLU team side there was an MLOps engineer as well as a data scientist, who also served as project lead. The team organization of the client was unusual, i.e. the project lead was from outside of the hospital with which the NLU team worked, the other partners were IT staff from the hospital, a professor that was the business sponsor and a doctor which served as domain expert.

Project Outcome

The project was aborted prematurely, even before data was transferred to the NLU team. There was a communication gap of multiple months when nobody on the client side reacted to inquiries of the NLU team.

Most important Characteristics

Process - Project Management - Stakeholder Management The most critical characteristic of this project was the stakeholder management. The IT professional, who was supposed to deliver data and help with the setup and the integration of the software in the hospital, was opposed to the project because he was not involved in the ideation phase of the project. This problem could not easily be resolved by the customer as there was no more IT staff available for the project. Further, the external project lead did not engage in communication with the NLU team, which made the project even more difficult.

Process - Project Ideation - Acquisition Although the acquisition of the project was successful, not all relevant stakeholders were present during the acquisition and not all prerequisites for the project were made clear upfront. It seemed like the client was unaware that they need to actively participate in the project in the form of IT support, labeling and project management.

Process - Project Ideation - Data Tools The availability of an annotation tool on the side of the NLU team was an icebreaker for acquiring the project, as the medical doctor on the client side was able to see that the team would not start from scratch and already had experience with similar projects.

3.4.4 Interview: Automating Tendering Processes for Electrical Products

In this project the goal was to create a service which automatically recommends products for a tendering offer in the domain of electronic devices. The client was a large company in the field of electrical engineering.

Project Setting and Problem Description

The client already had made bad experiences with prior machine learning projects and had low expectations in the beginning of the project. This led the NLU team to starting with an analysis of what had been done in the prior projects and focusing on a better understanding of the problem domain, as this was identified as an issue in the former projects. A crucial decision was made to narrow the scope

and reduce the complexity of the task by starting with the most frequently used products. The decision was made to split the project into multiple phases to give the client the possibility to abort the project if the results are not promising. The project was split into a proof-of-concept with a termination point after an initial modeling iteration and an integration project with another review and termination point in the middle of the project. In total the project was divided into three modeling and improvement iterations.

Roles involved

In the beginning of the project the NLU team consisted of a project lead, a senior data scientist as well as a junior data scientist. In the second phase of the project a senior and a junior MLOps engineer joined the team to help with the integration of the software on the customer side. The client joined the project with two managing partners, a project lead and a domain expert.

Project Outcome

The project was a success in all regards, i.e., the machine learning results surpassed expectations and the final model was deployed on the client side for further evaluation. The customer reported that the application saved them time and that their users were satisfied.

Relevant Characteristics

Many characteristics influenced the project. The project benefited from: project management, involvement of stakeholders such as domain experts and IT, thorough requirements engineering, appropriate skills of the staff, team organization, a good mutual understanding, high maturity and readiness of the customer, good coordination, communication and collaboration, an agile project management approach, a good application architecture design with a clear separation of concerns, reusing a pre-trained model, frequent user testing and feedback as well as the fast generation of a baseline model with low complexity.

Characteristics which influenced the project negatively were legal aspects, in particular the contract creation, and the scheduling between the project phases.

3.4.5 Interview: Tenancy Law Assistant

This project was a proof of concept for building a tenancy law assistant for a large publisher in the legal domain. Goal of the proof of concept was to show that extracting information from court rulings in the tenancy domain could be automated to a high degree.

Project Setting and Problem Description

The client wanted to build a tenancy law assistant. In a previous attempt, they managed to build a similar assistant for a different legal domain, albeit with high manual efforts. The NLU team should explore how much of the process could be successfully automated. To this end, court rulings had to be annotated, to train a model to automatically extract relevant information. Given these information, the goal was to make court rulings more comparable. The client and the NLU team created annotation guidelines and jointly established annotation processes, utilizing the annotation tool by the NLU team.

The annotation guidelines were improved over the course of two annotation iterations. In the end, roughly 600 court documents were labeled. The NLU team had to invest significant efforts into data preprocessing as the documents were very long, with context sizes exceeding the memory capacity of standard models. However, the team managed to successfully reuse language models pre-trained on the legal domain. To model the extraction of arguments of the plaintiff and the accused parties, a public reading comprehension model was repurposed. Finally, a service was developed which processed a court ruling, extracted relevant information and presented them in a visual demonstrator.

Roles involved

On the side of the NLU team a project lead was involved together with a senior data scientist. The project lead also participated actively as data scientist. On the client side a project lead was involved as well as legal domain experts and data scientists.

Project Outcome

The project was successful, i.e. the team reached high scores on the relevant metrics and the service was successfully deployed on the client side for further testing. The client expressed a high satisfaction with the whole project.

Most important Characteristics

According to the interviewee, the most relevant characteristics were the reuse of code, software and models, a good involvement of stakeholders, mutual understanding of the goals, transparent communication and good planning and project management on both sides. The characteristics which influenced the project negatively were a complex data augmentation process, including the creation of annotation guidelines, a difficult team organization, as some skills were lacking, legal issues with the contract and the problem that no clear business case existed on the client side.

3.4.6 Interview: Medical Information Extraction for Telemedicine

The goal of this project was to build an NLU component which enables automated information extraction of diagnosis, medication and different diagnostic markers from patient documentation. The component should have been integrated in a telemedicine component. Doctors were supposed to be given the possibility to correct information extraction mistakes.

Project Setting and Problem Description

The machine learning problem in this case was a typical information extraction problem, i.e., it contained the named entity recognition task to extract 15 different information types, such as diagnosis or medication of a patient, and the relation extraction task to link entities such as the name of a medication to its dosage. A significant aspect of this project was the privacy of the data and the regulation in Germany. All data processing had to be done on hardware hosted on the customer side. Further, no annotated data was available but rather lists of diagnosis and medication. These could be used to automatically pre-annotate documents. Additionally, the annotation tool of the team should be

modified and integrated in the software of the client, such that data could be annotated and information corrected at any time by the medical staff.

Roles involved

On the NLU team side, several stakeholders were involved. The product owner of the NLU annotation tool was part of the project team as project lead as well as in the role of product owner. An MLOps engineer was supposed to guide the modification of the annotation tool and help with the development. On the client side, the project lead as well as an external software developer were involved. Further, there was a steering committee consisting of the CEO of the client, the team lead of the NLU team and the department head of the department the NLU team belonged to.

Project Outcome

The project was aborted right after the start of the project due to legal issues that could not be resolved. The main issue here was the liability in the case that a mistake in the information extraction led to a misdiagnosis or wrong treatment of a patient.

Most important Characteristics

According to the interviewee, multiple characteristics influenced the project outcome. The following characteristics had a positive influence on the project: The reusability of existing software, the availability of an annotation tool and the reusability of existing models and rule sets. The project was negatively affected by the lack of availability of IT experts on the client side, legal issues, privacy concerns and a difficult project ideation process. Further characteristics that influenced the project where the absence of labeled data, which resulted in a large focus on data augmentation and the availability of structured data to weakly annotate documents.

3.4.7 Analysis

We aggregate and analyze the most important characteristics of the projects. Our analysis leads to a condensed project-relevant view on our catalog. 45 different characteristics were mentioned, highlighting that a variety of characteristics are relevant for NLU projects. The most frequently mentioned characteristics are ”Mutual Understanding” (10 projects), ”Team Organization” and ”Data Augmentation” (9 projects), ”Model Reuse” and ”Software/Code - Reusability” (8 projects), ”Technology & Infrastructure - Data Tools” (7 projects), ”Stakeholder Involvement - Domain Experts”, ”Stakeholder Management - Communication” and ”Project Management” (6 projects) as well as ”Technology & Infrastructure - Visualization Tool” (5 projects). The 10 most important characteristics and their number of occurrences can be seen in Figure 3.18. All 45 characteristics and their ranking is attached in Section A.3 in the appendix.

It becomes evident that a mixture of general data science characteristics and foundation model specific characteristics were vital for the projects of the NLU team. This supports our approach of combining the results of the two literature studies. Concerning the foundation model specific characteristics, especially the possibility to reuse already pre-trained models played a vital role in projects. The team often resorted to using models from a public model hub and fine-tuning them

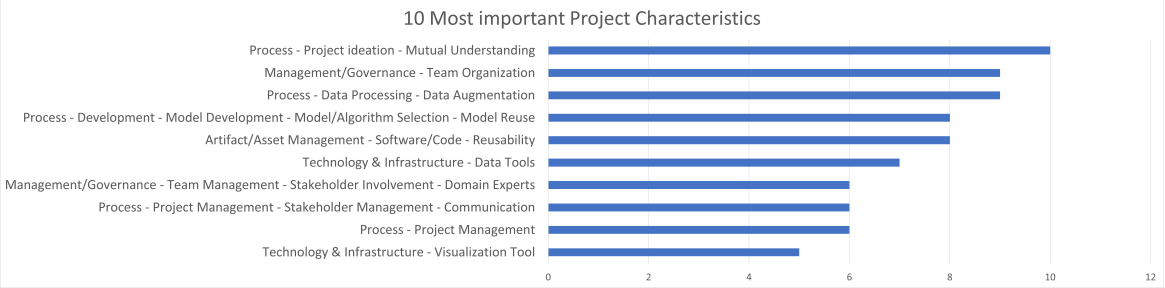

Figure 3.18: The number of projects in which characteristics were mentioned as being most important to the project process and outcome.

towards project-specific data. Further, pre-training their own model was crucial in at least one project, leading to significant performance improvements.

The fact that many of the characteristics are not foundation model specific indicates that a foundation model project methodology needs to account for these characteristics as well.

Limitations

There are multiple limitations to our NLU project study. The case studies we analyzed are all based on the experience of a single team. This leads to a bias based on the projects, experience, team members and clients of this team. We expect that the 45 most relevant characteristics that we distilled from the case studies, would differ for other teams. However, the team's projects were spread across multiple domains and customers, providing a broad view on the scale and style of NLU projects.

Another limitation is that our analysis only considers NLU case studies, i.e. it is restricted to certain types of problems and models. At most, some of the projects contained multi-modal data, there were no projects in which the modality of the data did not contain text. However, we believe that this is a minor issue as foundation models are fueling a consolidation of methods, technologies and processes across modalities.

3.5 Conclusion

In this chapter we systematically studied data science project and foundation model literature to understand the project characteristics, namely the challenges, success factors and requirements, of modern data science projects. We discovered 1,396 characteristics, which we aggregated in 163 categories and subcategories. The top-level categories cover the project "process", "artifact/asset management", "technology and infrastructure" and "management/governance". They provide a holistic view of the whole data science project process and project-overarching characteristics. Our catalog of characteristics is the most complete to date, summarizing the results of existing meta-studies and adding many new characteristics. Further, to the best of our knowledge, it is the only such catalog that considers the implications of using foundation models.

To take a more practical perspective on our characteristics, we analyzed 26 case studies in the NLU domain and found that 45 subcategories of our catalog were of particular relevance for the project outcome. They represent a project-relevant, condensed version of our catalog. The most frequently mentioned subcategories address the mutual understanding of a project, the team organization, data

augmentation processes, in particular labeling, reuse of models and software, data tools, involvement of domain experts, communication and project management as well as the use of visualization tools. This shows that both general data science characteristics as well as foundation model specific project characteristics play an important role in the execution of data science projects which make use of foundation models. A project methodology tailored towards foundation model usage should hence cover both.

The catalog provides the requirements that a foundation model methodology should meet. Naturally, the question arises if an existing methodology would already sufficiently cover the project-relevant characteristics or even the complete catalog of characteristics. Such a methodology could be deemed adequate for projects in which foundation models are used. In the next chapter we will examine existing methodologies with respect to how well they cover the characteristics from both, the general catalog and the project-relevant catalog.

CHAPTER 4

Metastudy and Assessment of Data Science Methodologies for Foundation Model Projects

In this chapter we present a systematic evaluation of the 27 data science project methodologies, we introduced in Section 3.1.2 and describe in detail in Section A.1 in the appendix. We evaluate them with respect to the catalog of characteristics, which we derived in Chapter 3. Our goal is to assess the suitability of these methodologies for use in foundation model projects. We start with an evaluation of the methodology groups and the individual methodologies which we assigned to them. Each methodology is introduced with a brief description, detailed descriptions can be found in the appendix.

The inspiration of our analysis is a similar study performed by Martinez et al. [88]. In their study, Martinez et al. focused on the 21 main challenges of data science projects, which they grouped into the categories ”team management”, ”project management” and ”data & information management”. They defined a set of criteria to solve the individual challenges and then assessed whether the criteria were not fulfilled, hardly fulfilled, partially fulfilled or fulfilled to a great extent. Based on the degree of fulfillment, they assigned a score between 0-3 points. The authors note that this way of rating the methodologies is highly subjective.

We propose a more objective score by only differentiating three possible states. If a characteristic is not mentioned we assign 0 points, if it is mentioned but not addressed with a recommended action we assign 1 point and if a characteristic is addressed with a recommended action, we assign 2 points. We refrain from assessing how good the recommended action is, to avoid subjectivity. Instead of using the 21 challenges, presented by Martinez et al., we consider our characteristic catalogs. We use the complete catalog with all 165 characteristic groups as well as the project-relevant characteristics which were deemed most important based on our analysis of NLU case studies. This way we can analyze both, the suitability for methodologies in a general data science context and in the context of NLU projects.

4.1 Assessment of Data Science Project Methodologies

In this section we present the assessment of data science project methodologies w.r.t. the project characteristics we derived in the last chapter. We group the methodologies according to the data science paradigm shifts we described in Section 3.1.2. We present a group summary of the results and then detail the results for each methodology within it.

4.1.1 KDD and Data Mining Group

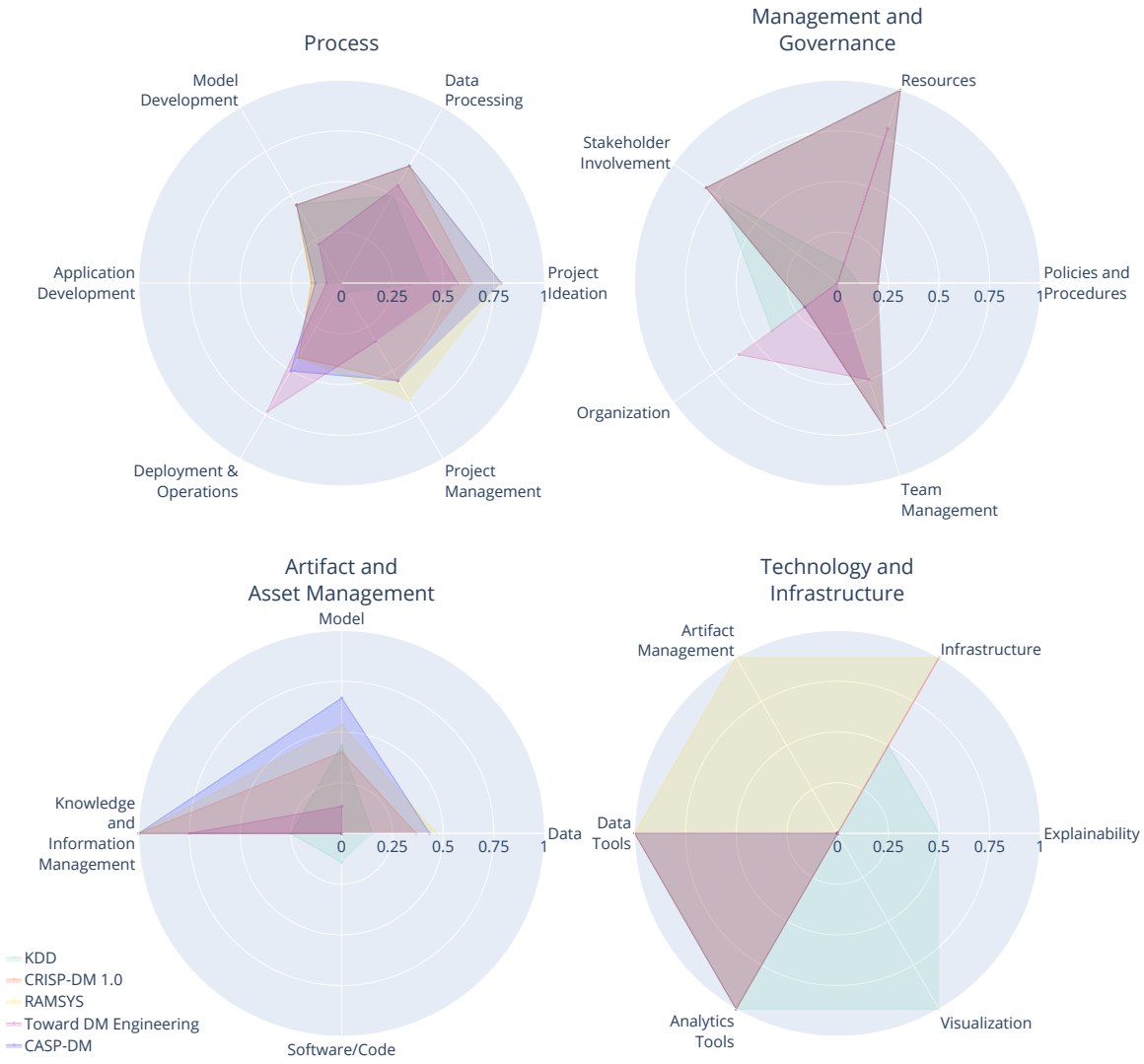

Figure 4.1: Radar diagram depicting the per-category scores of the KDD and Data Mining group.

This group contains early methodologies such as KDD [30] and CRISP-DM [2], which recognized the need for specialized data science methodologies, as well as extensions and adaptations of those methodologies to specific challenges and scenarios. The methodologies in this group, such as RAMSYS [111] and CASP-DM [112], which directly extend CRISP-DM are among the best methodologies in our assessment. They make useful additions to the already good CRISP-DM, by extending it to remote collaboration settings, introducing artifact stores and context-aware modeling. None of the methodologies in this group tackles the characteristics relevant to the development of applications and the management of software and code artifacts. The scores which the methodologies reach for each of the top-level, as well as second-level categories in our catalog can be seen in the radar plot in Figure 4.1.

Knowledge Discovery in Databases

The Knowledge Discovery in Databases (KDD) process was introduced in [29] by Matheus et al. and further elaborated on by Fayyad et al. [30]. The purpose of KDD was structuring the process of knowledge discovery projects. The process explicitly calls for a human-centered approach to knowledge discovery, meaning that the process revolves around a user which applies the process to extract knowledge from databases. It is structured into 9 phases, ranging from domain understanding to the application of project results. The core phase of the methodology is the data mining phase. The methodology provides brief descriptions of all phases and suggests an iterative approach to data science projects. Only three roles are mentioned in the methodology. The user, also referred to as data analyst or data archaeologist, the domain expert and the business sponsor. The methodology does not provide any detailed lists of artifacts. It uses data as the central artifact along with a data dictionary to explain the data. It further mentions only a few tools and describes typical machine learning methods used during the data mining phase.

Criticism: The KDD process laid the foundation for modern data science methodologies. However, at the time of writing it is almost 30 years old. Since its creation, the complexity and scale of data science projects have significantly increased. Although KDD mentions the integration of knowledge in another system as a result of a project, it neglects project characteristics concerning the development of ML applications, model and application deployment as well as their maintenance. Further, KDD mentions some roles which are involved in data science projects, but it mostly omits details concerning the management and governance of data science projects. It does not account for team management, policies and procedures and resource planning. KDD provides some insights into the Technology and Infrastructure category. In particular, it recommends some methods for data analysis and data visualization. KDD barely touches aspects of artifact and asset management. In total, KDD covers approximately 29% of all the project characteristics in our catalog. Considering only the 45 project-relevant characteristics, KDD achieves a score of 31%. Hence, it is not suitable for modern foundation model projects.

CRISP-DM

The Cross Industry Standard Process for Data Mining (CRISP-DM) succeeded the KDD methodology [2]. Despite being almost 25 years old it is still considered, the most popular data science project methodology to date [100]. CRISP-DM is structured into 6 phases, which are comprised of generic tasks. The CRISP-DM user guide provides detailed descriptions of these phases, the tasks as well as the artifacts which are used and produced throughout a data science project [102]. CRISP-DM does not provide any structured overview of roles which are involved in a project and how they are organized. However, it mentions 16 different roles.

Criticism: CRISP-DM provides a good overview of the processes in data science projects, however it fails to sufficiently address modern challenges e.g. in application development and deployment. It covers the characteristics of the Management & Governance category well, however omitting characteristics which address the organization in which a project team is embedded and the policies and procedures. CRISP-DM gives only marginal guidance with respect to the Technology and Infrastructure category, mostly mentioning tools for handling data and analyzing it. Concerning the Artifact and Asset Management category, CRISP-DM covers all characteristics related to knowledge and information management. However, it provides only spurious information related to other artifacts

such as data, models and software/code. In total, CRISP-DM achieves a score of 41% over all characteristics and 57% for the project-relevant characteristics. This indicates that CRISP-DM is better suited than KDD for models using foundation models. However, it still omits almost half of the project-relevant characteristics and two thirds of the characteristics in the complete catalog.

RAMSYS

The RApid collaborative data Mining SYStem (RAMSYS) methodology adapts CRISP-DM to remote collaboration settings [111]. It maintains the same phases as CRISP-DM but augments them with guiding principles, tools, roles and infrastructure suggestions which enable better remote collaboration. Central to the methodology is the so-called information vault, a tool for managing knowledge, data and experiments.

Criticism: RAMSYS strongly benefits from building upon CRISP-DM. It mainly contains everything that CRISP-DM introduced but augments it reasonably. In the Process category RAMSYS extends CRISP-DM with coordination and collaboration aspects, as well as researching previous projects for useful information. Considering artifact and asset management, RAMSYS introduces data management aspects. In the Technology and Infrastructure category, RAMSYS extends CRISP-DM by addressing characteristics concerning the project infrastructure and artifact management. Only in the Management & Governance category there are no changes compared to CRISP-DM. Since RAMSYS builds upon CRISP-DM it naturally outperforms it. Across all categories, RAMSYS obtains the best score of 54%. Considering the project relevant characteristics, RAMSYS obtains the second-best score of 61%.

A Process Model for Data Mining Engineering

Marbán et al. took a different approach from CRISP-DM with their Process Model for Data Mining Engineering (Toward DM Engineering) methodology [86]. They take an approach to data science which is similar to software engineering and name it data mining engineering. They call for more reuse, better integration and interchange of project results. Further, their methodology is concerned with aspects other than the development in the projects, such as organizational and managerial processes. Instead of structuring data science projects into phases, they propose four process groups: organizational processes, development processes, project management processes and integral processes.

Criticism: This methodology is the first to suggest using insights from software engineering processes to address the immaturity of data mining projects. This makes intuitive sense since both fields are closely related. The authors further address the differences between data mining and software engineering projects and address the strong focus of previous data mining methodologies on the development processes, i.e., data preparation and modeling processes. They add organizational, project management and integral processes. The organizational processes assure a holistic view of projects, enabling companies to learn from past experiences and carry their insights into future projects. Another interesting addition are acquisition and supply processes within the project management processes, something not addressed by previous methodologies as well as a focus on time and budget constraints. The methodology describes the processes on a high-level, leaving a lot of room for interpretation for the project members. They also do not suggest a project lifecycle and do not structure the project into phases, but rather say that this needs to be done according to the needs of each project. Further, they do not add roles, artifacts and tools and techniques. Martinez et al.

also criticize that the authors do not address team related challenges [88]. The methodology gives detailed descriptions of characteristics belonging to data processing and deployment and operation. However, it neglects modeling and application development characteristics, as well as most of the characteristics relevant to project management. Regarding the Management & Governance category, it omits most characteristics and mainly addresses resource related characteristics. In the Technology and Infrastructure category it only addresses characteristics related to the project infrastructure. Finally, in the Artifact & Asset Management category, it addresses most of the characteristics related to Knowledge and Information Management and only two further characteristics concerning the Model Artifact. Across all characteristics, the methodology achieves a score of 24% and a score of 38% for project relevant characteristics.

CASP-DM

CASP-DM is an extension of CRISP-DM and shares its phases, tasks and artifacts. It extends CRISP-DM by tackling the challenge of having context-changes in projects. It assumes that knowledge in the form of models can be reused by various methods such as reframing, retraining and revisioning and proposes this as a possible solution to changing contexts. Reframing describes the process of using a reframing method to adapt a model to a novel context, retraining simply describes the task of retraining a model with data from the novel context and revisioning means that parts of a model are patched or extended to accommodate a new context. A fundamental change to other methodologies is the assumption that models can be trained to be more versatile and adaptable. The authors argue that changes in the data science project process are needed to accommodate for the context-awareness. CASP-DM was released before the creation of foundation models and hence does not address them and their particular characteristics explicitly, however it introduces many concepts and processes which can be used in projects which utilize foundation models. It does not focus on model adaptation across projects, but rather in the context of a project with changing data.

Criticism: CASP-DM is the first methodology that explicitly models the process of reusing models, albeit in the scope of a single project. In the Process category, CASP-DM scores slightly better than CRISP-DM as it tackles more characteristics concerning the Model Development category, such as the reuse of models. However, it also omits most of the characteristics of the Application Development category. In the Artifact & Asset Management category it extends CRISP-DM with characteristics concerning model and data management. With regards to the Technology & Infrastructure category and the Management & Governance category it does not provide any changes compared to CRISP-DM. In total, this leads to a slightly better score than CRISP-DM. Across all categories CASP-DM scores 44% and considering only the characteristics most relevant for the projects, it scores 61%.

4.1.2 Big Data Group

The methodologies in this group specifically address big data related challenges. In contrast to the KDD and Data Mining Group, they place larger emphasis on characteristics which affect the organization in which a project is embedded. Surprisingly, most of the methodologies lack important details on the process, artifact and asset management, as well as technology and infrastructure characteristics. The radar plot of this group is depicted in Figure 4.1.

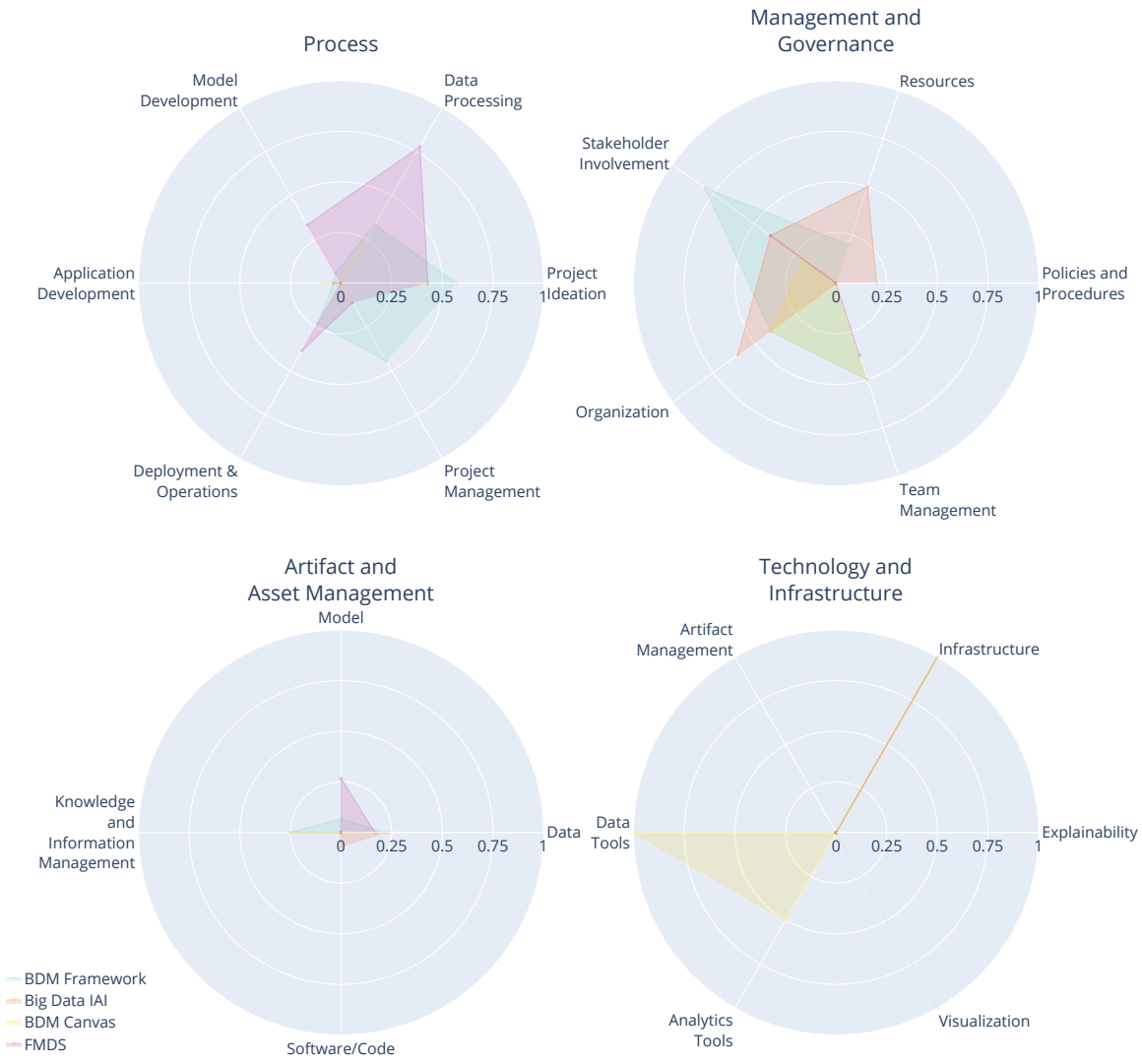

Figure 4.2: Radar diagram depicting the per-category scores of the Big Data group.

Big Data Managing Framework

The Big Data Managing Framework (BDM Framework) methodology explicitly tackles the challenges introduced by big data [113]. Their authors argue that at the time of the creation, there was a lack of suitable methodologies for big data projects. The methodology explicitly addresses organizational aspects of big data aspects, such as the need for a cultural shift. They also call big data projects shorter and more difficult to implement. They emphasize that the involvement of relevant stakeholders is a key success factor throughout the project lifecycle.

Criticism: The big data managing framework is the first data science methodology to address the challenges of big data projects. A key insight of Dutta and Bose is that big data requires organizational changes and the support of key stakeholders in a company. They emphasize the importance of the involvement of (senior) management and IT stakeholders as key success factors for big data

projects. Further, they advertise cross-functional teams which involve key business stakeholders, customer behaviour experts, management, data science and IT. The framework is rather high-level but accompanied with a case study. Aside of tools and techniques for business understanding, no general tools and techniques for the execution of the different phases are provided. However, the authors emphasize the importance of visualizations of results. Martinez et al. criticize that validation of data analytics and machine learning models is not addressed sufficiently by the methodology [88]. The Big Data Managing Framework Methodology does not address many characteristics of the Process category. It provides details about the ideation of a project and the preparatory work that has to be done before the development begins. However, it lacks details concerning data processing, modeling, application development, deployment and project management. In the Management & Governance category it addresses characteristics about the organization, stakeholder involvement and team management. However, it does not address policies and procedures and, somewhat surprisingly given its focus on big data, it also does not address characteristics regarding resources other than budget. The methodology discusses characteristics concerning the project infrastructure and tools for visualization and emphasize the importance of appropriate visualizations. Lastly, in the Artifact & Asset Management category the methodology provides insights concerning the data artifact and mentions some of the ”big data V” challenges without a lot of details. In total, the methodology scores 29% across all categories and 38% for the project relevant characteristics.

Big Data Ideation, Assessment and Implementation

The Big Data Ideation, Assessment and Implementation (Big Data IAI) methodology focuses on guiding the introduction of big data in organizations [114]. Because of its scope, it is not strictly a project methodology. However, we include it in our evaluation since other project methodology meta-studies considered it to be one [88]. The methodology puts a strong emphasis on the assessment of big data use cases and their requirements and is hence useful for the early phases of data science projects.

Criticism: While the big data management framework already established that the introduction of big data requires structural changes in a company, this methodology focuses on the processes required to transform a company. It distinguishes the case of companies which want to use data to optimize their own business and companies which would like to establish new business cases based on big data analysis. In contrast to the big data management framework, which bases its methodology mainly on a single case study, the authors claim that this methodology aims for a more scientific approach, combining insights from IT value theory, workgroup ideation processes and enterprise architecture management. While the methodology does not provide many insights on how to execute data science projects, it provides useful insights for companies which want to utilize big data. This translates to the customer role in most data science methodologies and provides valuable insights about business understanding and deployment of solutions. Introducing and using data science solutions can fundamentally transform how companies operate. Hence key stakeholders should be involved from the beginning. The methodology gives interesting insights about the people who should be consulted to assess data science use cases and tools, techniques and artifacts which should be generated during the process. Martinez et al. also criticize that the methodology misses issues related to team and data management [88]. The methodology does not cover the Process category in great detail and mainly addresses characteristics of the Project Ideation subcategory. It does not provide any details on how to process data, how to do modeling, application development, deployment or project management.

Considering the Technology and Infrastructure category it only addresses infrastructure aspects. In the Artifact & Asset Management category it only discusses a few characteristics of the data artifact. Finally, in the Management & Governance category it addresses some characteristics across the subcategories, with a particular focus on stakeholder involvement, organization and resources. Across all characteristics the methodology achieves a score of 22%. For the project relevant characteristics it achieves 16%.

Foundational Methodology for Data Science

The foundational methodology for data science (FMDS) [115] by IBM addresses the fact that emerging technologies automated steps in model and application building and made data science more accessible. The methodology has similarities to previous methodologies such as CRISP-DM, but enriches them to cope with emerging trends such as big data, text analytics in predictive modeling and the aforementioned automation of machine learning processes. The methodology consists of 10 phases which are highly iterative.

Criticism: The foundational methodology for data science does not introduce many novelties in comparison to former methodologies. A key emphasis of the authors is that novel trends such as the automation of parts of the data processing and model building should be incorporated in the data science process. The methodology does not provide explicit steps on how to automate them and which tools to use for the automation. The methodology only mentions three roles and defines no artifacts. According to Martinez et al. the methodology inherits the same problems that CRISP-DM has, i.e., lack of role definitions, reproducibility, knowledge sharing and gathering and data security [88]. In the Process category, the methodology addresses characteristics from the Project Ideation, Data Processing, Model Development and Model Deployment & Operation subcategories. It omits application development and deployment and does not provide any foundation model specific insights. Aside for the strong iterative nature of the methodology, it also does not provide any guidance concerning project management. Further, the methodology only addresses a few characteristics about the data and model artifact considering the Artifact and Asset Management category. In the Management & Governance category, the methodology only addresses the involvement of customers and domain experts, as well as team organization characteristics. Somewhat surprisingly, the methodology offers no insights into the Technology & Infrastructure category, despite motivating the need for a methodology that is adapted to novel methods. Across all characteristics, the methodology achieves a score of 14% and 17% for project relevant characteristics.

Big Data Management Canvas

The Big Data Management Canvas (BDM Canvas) methodology aims at tackling data science from a technical and a business side [34]. It combines aspects concerning the 5V of big data with aspects of value creation, market focus and visualization. All phases in the methodology contain a business view and a technical view and it is highlighted that business users and IT should be involved in all phases and collaborate.

Criticism: The unique feature of the big data management canvas is strictly separating business tasks from technical tasks. The authors argue that IT and business stakeholders should be involved in every phase of a project and provide a canvas to list the different tasks that both sides have in each phase. Martinez et al. argue that a strength of the methodology is its focus on value generation [88]. The

methodology does not provide information concerning roles, artifacts and tools and techniques. It also provides nearly no guidance regarding the process of data science projects. In the respective category only few characteristics are addressed, mainly from the Data Processing subcategory. In the Artifact & Asset Management category, the methodology mainly tackles characteristics belonging to the data artifact, in particular concerning the 5V of big data. Considering the Management & Governance category, the methodology does not include characteristics regarding resources and policies and procedures. Further, it only addresses few of the characteristics of the other subcategories. Regarding the Technology and Infrastructure category, the methodology addresses characteristics concerning infrastructure, data and analytics tools. It reaches a score of 19.3% across all characteristics and 14.4% for the project relevant characteristics.

4.1.3 Agile Group

The methodologies in the Agile group specifically tackle project management challenges due to growing sizes of project teams and the increasing complexity of projects. They introduce agile concepts in data science methodologies. The focus on team organization, stakeholder involvement and project management can be seen in the radar plot in Figure 4.3.

ASUM-DM

The Analytics Solutions Unified Method for Data Mining (ASUM-DM) methodology is considered to be an extension of the CRISP-DM methodology [116]. It extends and modifies CRISP-DM with respect to project management, agile development, prototyping, deployment, maintenance and infrastructure aspects. ASUM-DM offers detailed descriptions of phases, tasks, roles and artifacts.

Criticism: ASUM-DM provides a detailed definition of phases, activities, roles and artifacts. It further puts a lot of emphasis on deployment and is the first methodology to establish activities for operation and optimization of solutions even after a project ends. Another novelty of ASUM-DM is the recommendation to combine traditional and agile project management methods during data science projects. Following CRISP-DM, ASUM-DM is one of the most comprehensive methodologies for the execution of data science projects. A downside of ASUM-DM is that it is tailored towards IBMs SPSS software. Angee et al. argue that ASUM-DM in its plain form is not suitable for big data projects and multi-disciplinary, multi-organizational settings [104]. They propose changes to the project structure and additional activities to accommodate for big data requirements. ASUM-DM reaches the best coverage of characteristics in both the Management & Governance category and the Process category. In the Process category, it covers many characteristics in the Project Ideation, Deployment & Operations and Project Management subcategories, which CRISP-DM did not cover. Notably, it is the only methodology to cover all characteristics in the Project Management subcategory. Nevertheless, ASUM-DM omits most characteristics regarding application development. In the Management & Governance category covers almost all characteristics, except for the ones belonging to the Policies and Procedures subcategory. ASUM-DM pays less attention to artifact & asset management characteristics and omits almost all characteristics of the data and software/code artifacts. In contrast, it addresses all characteristics belonging to the Knowledge and Information Management subcategory. Finally, concerning technology and infrastructure, ASUM-DM only covers the project infrastructure and visualization tools. Across all characteristics ASUM-DM scores second-best with a score of 45%. For the project relevant characteristics ASUM-DM achieves the best score of 62%.

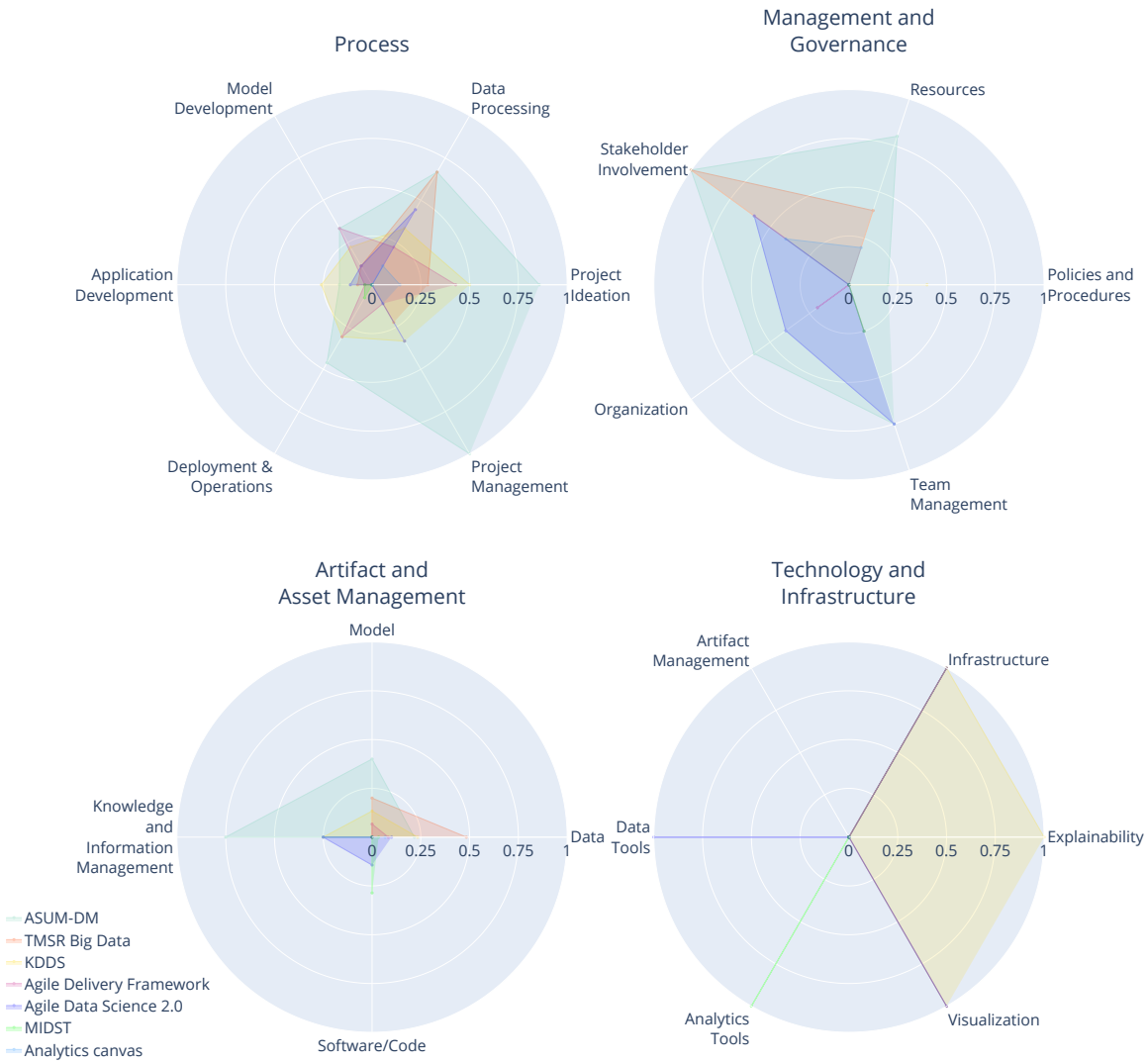

Figure 4.3: Radar diagram depicting the per-category scores of the Agile group.

Towards Methods for Systematic Research on Big Data

The TMSR Big Data methodology by Das et al. addresses the issue of uncertainty in data science projects. For their methodology they claim that data science projects should be regarded as special instances of research projects. Given the uncertain and ”researchy” nature of the projects, Das et al. argue that data science projects should be executed in an agile manner. More specifically they highlight the benefit of short iterations for iterative and incremental improvements based on user feedback. They call for automation of tasks whenever possible and state that project management should support the efficient execution of the project and not generate a restrictive structure.

Criticism: The research process which the authors describe for the projects lacks detail and they further do not provide any roles, tools and techniques nor artifacts. The lack of roles and interaction recommendations is also criticized by Martinez et al. [88]. The methodology does not address many

of the characteristics in the Process category and even omits most of the characteristics belonging to modeling aspects. It is mostly concerned with characteristics related to problem understanding and data processing. In the Management & Governance category, the methodology covers all of the characteristics belonging to the Stakeholder Involvement subcategory, however it omits almost all other characteristics. In the category Artifact and Asset Management it is mostly concerned with the data artifact. Finally, regarding the Technology and Infrastructure category, it only addresses Infrastructure. The methodology reaches an overall score of 27% across all characteristics and 22% for project relevant characteristics.

KDDS + Data Science Edge

The Knowledge Discovery in Data Science (KDDS) process along with the Data Science Edge Process Model by Grady et al. [31] is tailored towards challenges and requirements in big data development and machine learning application development and deployment. In particular, they argue that CRISP-DM does not sufficiently address big data aspects such as data storage, big data algorithms, analytics techniques, process automation and systems development. Lastly, they criticize CRISP-DM for degrading to waterfall-like projects. To that end they design their methodology according to agile principles.

Criticism: KDDS + DSE reiterates many ideas of previous methodologies which incorporate agile ideas. It also introduces the notion of data science products and introduces the role of a product owner and the concept of minimum viable products as intermediate results of data science projects. In addition, it is the first methodology to incorporate regulatory issues to be considered in the beginning of a project. Similarly to the big data management framework, the methodology puts a large emphasis on the visualization of results, as this is regarded as crucial for the communication of results to non-technical decision makers. The methodology does not provide many insights about roles and team management, other than to keep teams stable. This is also criticized by Martinez et al. [88]. No artifacts, tools and techniques, aside of NoSQL, are defined to support the methodology. The methodology addresses some characteristics across all subcategories in the Process category but lacks important details about model development. In the Artifact and Asset Management category, the methodology mainly discusses the data artifact. Except for policies and procedures, the methodology does not address any of the subcategories in the Management and Governance category. In the Technology and Infrastructure category, the methodology addresses explainability and visualization tools as well as the project infrastructure. Across all characteristics, the methodology achieves a score of 25% and a score of 27% for the project relevant characteristics.

Agile delivery framework for BI, fast analytics and data science

In 2016 Larson et al. analyzed how well the agile principles could be applied to data science in their Agile Delivery Framework [117]. They conclude that agile is a good fit, in particular for big data projects. They propose to deliver projects in increments to account for changing businesses and technologies and for flexible risk management. Defining deliverables should be part of creating a project roadmap. Their methodology suggests a project lifecycle with seven phases, which are very similar to CRISP-DM.

Criticism: The agile delivery framework is the first methodology to explicitly apply the agile manifesto to data science projects. The authors argue that agile methods are an excellent fit for data

science and argue for incremental and iterative delivery of deliverables within a project. The authors recommend timeboxed iterations within the Model/Design/Develop phase and close iterations with the Analyze/Visualize and Validate phases. They also argue for the benefit of experienced teams over inexperienced teams, which require more guidance, less documentation, rapid development and deliveries and for ongoing communication with customers. The methodology lacks any clear role definitions, artifacts and tools and techniques. Martinez et al. further criticize that the framework does not tackle data management and reproducibility issues [88]. The methodology covers only few of the characteristics in each category of our catalog, which leads to a score of 11% across all characteristics and 16% for project relevant characteristics.

Agile Data Science Methodology

The Agile Data Science 2.0 methodology by Jurney introduces the agile data science manifesto [6]. It specifically addresses the fundamental differences between data science and software engineering, in particular the challenge of coping with uncertainty. Jurney argues that data science projects have to be embedded in an organization and aligned with activities outside of the projects. He analyzes dependencies between different roles and phases of projects and concludes that many data science projects degrade to waterfall-like projects. He therefore advocates for smaller teams in which people have multiple roles. In addition, Jurney advises to build web applications for data science solutions to provide an easy way of delivering insights. Jurney does not structure projects in phases, but rather into a data-value pyramid in which data is transformed into states of different value.

Criticism: Jurney introduces new concepts with his methodology. Rather than applying the agile manifesto, like previous methodologies, he defines a new agile manifesto for data science. The methodology centers around data products and recommends the development of web applications. Rather than defining phases and a concrete life cycle, Jurney introduces the concept of a value-pyramid which aims at transferring data into value. This is an interesting approach but leaves the reader with the question of how to traverse this pyramid. Jurney provides extensive role descriptions and argues for the need of generalists which take on multiple roles. He argues that this reduces communication overhead and prevents projects from deteriorating to waterfall life cycles. He does not provide any clear communication and interaction guidelines for teams. Further, no artifacts are described and the tools and techniques described do not address all aspects of data science projects. The methodology does not yield detailed task descriptions and the focus on the data-value pyramid leads to a low score in the Process category. The methodology omits the characteristics of the Project Ideation subcategory and most of the characteristics belonging to the Modeling, Deployment and Application Development subcategories. In the Artifact & Asset Management category, only few characteristics across all subcategories are tackled. In the Management & Governance category, the methodology provides strong insights into team management and Stakeholder involvement characteristics. Finally, concerning the Technology and Infrastructure the methodology addresses characteristics of the project infrastructure, data tools and visualization. Across all categories, the methodology achieves a score of 28%. With respect to the project relevant characteristics, it achieves 27%.

MIDST

MIDST was introduced by Saltz et al. [118]. It focuses on improving the collaboration between team members in data science projects and introduces a tool to that end. Its novelty is the utilization of

stigmergic collaboration, a concept derived from the development of free/libre open-source software (FLOSS). A key concept of MIDST is the visibility of documents, processes and functions for every team member. The philosophy behind MIDST is that work by different data scientists should be combinable and improvable in modular increments. MIDST does not define any specific phases, tasks, roles or responsibilities.

Criticism: Martinez et al. included MIDST in their overview and rating of methodologies [88]. We would argue that it is more a tool designed to support methodologies. However, for the sake of completeness, we include it in our analysis. The idea of stigmergic collaboration is a great contribution of MIDST and it tackles common challenges of data science projects. MIDST does not provide many details regarding the process of data science projects or any of the other categories in our catalog. Hence, it reaches a score of only 9% across all characteristics and 2% for the project relevant characteristics.

Analytics Canvas

The Analytics Canvas by Kühn et al. was introduced in 2018 [119]. It provides a semi-formal specification method to describe analytics use cases, the necessary company-wide data infrastructure and requirements for interdisciplinary domains. The authors criticize CRISP-DM for not explicitly addressing the primary strategic goals of an analytics project, because they consider a goal-oriented development process to be very important. With their Analytics Canvas they follow the four types of analytics as described by Steenstrup et al. [223], which are Descriptive, Diagnostic, Predictive and Prescriptive Analytics. In the model, Steenstrup et al. specify for each analytic category how much human input is needed and how sophisticated the analytics program shall be. The authors aim to tackle the problem of communication and joint understanding in multi-stakeholder settings, where the stakeholders have different backgrounds, e.g. IT, business and analytics. They list communication and enterprise-wide digitization strategies as keys to success for companies, and lack of collaboration across disciplines and lack of consistent methods and processes as most important pain points in analytics projects.

Criticism: The analytics canvas does not provide many insights about how to execute data science projects, which roles should be present, which artifacts or tools and techniques. It does deliver an interesting way of modeling analytics use cases, similar to UMLS. At a glance it provides an overview of relevant information regarding data sources, data description and modeling choices to project stakeholders and can be used as a mean for discussion. Similar as with MIDST, we would not have considered the Analytics Canvas a project methodology, but rather a tool to support methodologies. However, again Martinez et al. classified it as a methodology [88]. The methodology does not provide many insights considering our catalog and only achieves a total score of 6% over all characteristics. It achieves a score of 7% for the project-relevant characteristics.

4.1.4 Integrated Group

The methodologies in the Integrated group are bundled with software frameworks or even infrastructure. This is strongly reflected by their focus on technology and infrastructure characteristics and can be seen in the radar plot in Figure 4.4.

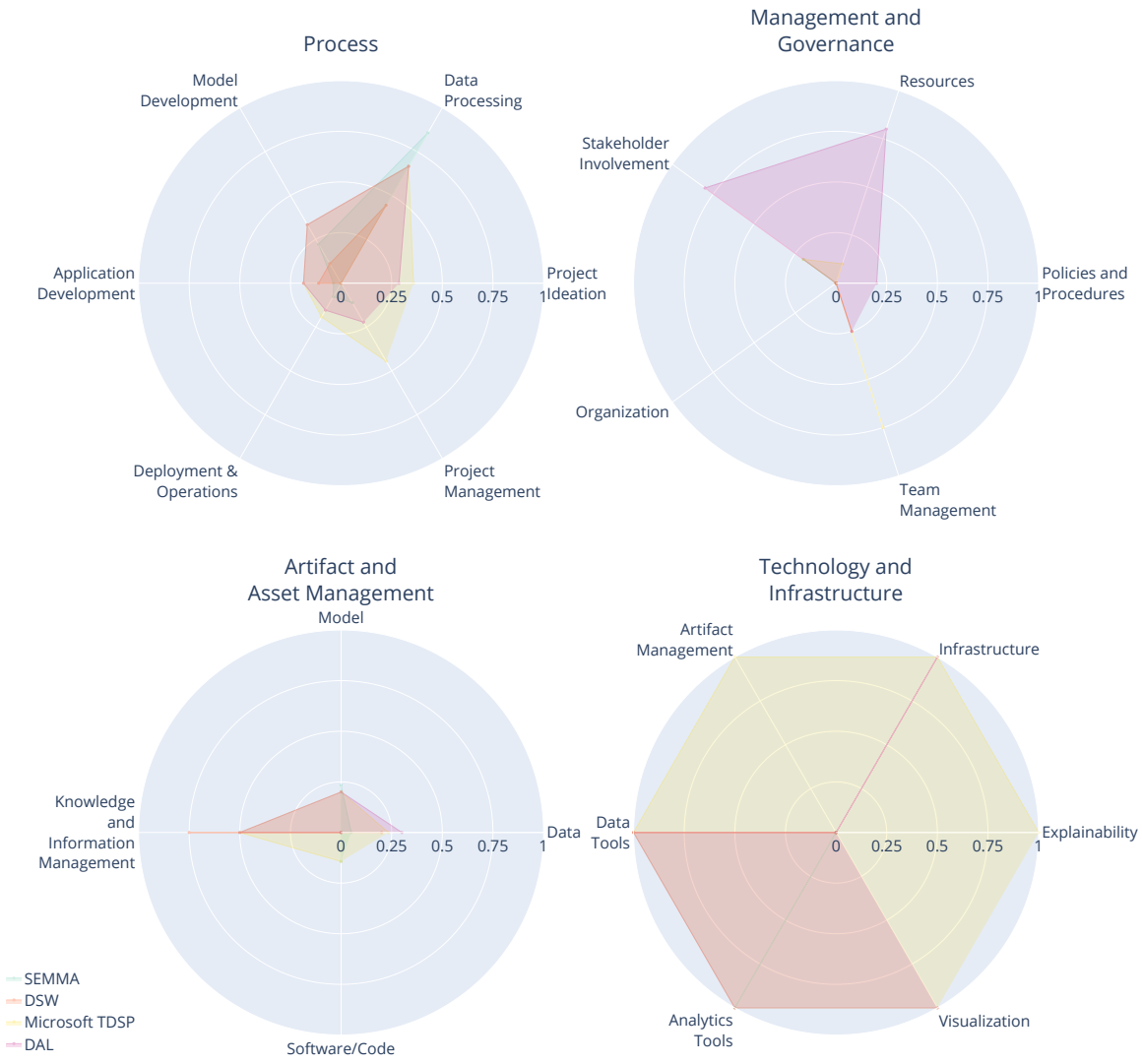

Figure 4.4: Radar diagram depicting the per-category scores of the Integrated group.

Data Science Workflow

In 2012 Guo introduced the research programming workflow [224], which he later renamed to data science workflow (DSW) [121]. The workflow puts a clear emphasis on the research-based work a data scientist has to perform and gives thorough details on how to tackle challenges that data scientists face during the phases of data acquisition, data preprocessing and modeling. It consists of only four phases and is centered around data and modeling issues. Along with the workflow, Guo developed tools to automate parts of it.

Criticism: Gao's data science workflow stems from his prior work, the research programming workflow. Its strengths lie in the incorporation of expert knowledge, which Gao obtained by interviewing and observing the work of data scientists, and the addition of tools to support this workflow. Martinez et al. add that having a reflection phase which is disjunct from the analysis phase

can also lead to benefits [88]. However, the workflow does not scale to the context of teams performing data science projects. This becomes immediately apparent through the lack of role definitions and project management. Concerning the Process category in our catalog, the methodology mostly addresses characteristics from the Data Processing subcategory and omits most of the others. In the Artifact and Asset Management category it addresses characteristics from the Knowledge and Information Management subcategory, as well as some from the Data subcategory, but none about software or models. In the Management and Governance category, the methodology only addresses characteristics about the communication, coordination and collaboration of team members. The tools which introduced with the methodology only concern the Data Tools subcategory. The methodology reaches a score across all characteristics of 11% and 7% for the project relevant characteristics.

SEMMA

SEMMA is an acronym for Sample, Explore, Modify, Model and Assess. It was created by the SAS Institute and is incorporated in the SAS Enterprise Miner product. The understanding whether SEMMA is a data mining methodology or not varies. However, today SAS refers to it as a data mining process and data mining methodology [225] and it is often referenced as data mining methodology in literature [94, 226–228]. SEMMA focuses on the data preprocessing and modeling aspects of data science projects. Its phases overlap with the data understanding, preparation and modeling phases of the CRISP-DM methodology. SEMMA does not provide any roles and artifacts and only a rudimentary life cycle description.

Criticism: The focus of SEMMA is on data analysis and model development. As an obvious shortcoming SEMMA does not include a business understanding phase and hence does not provide teams with a way of understanding the goal of a project [228]. This has also been proven to be a practical downside in projects, in particular when compared to using CRISP-DM [226]. SEMMA further does not explicitly define a deployment phase, i.e. it omits the phase which, in the words of the authors, is key to delivering value to a customer. However, SEMMA is deeply integrated with the SAS Enterprise Mining Software and can be regarded as the first of the methodologies that combines the phases of a project with appropriate tooling, especially designed to improve the project phases. Azevedo et al. even describe it as a practical implementation of the KDD process for the SAS Enterprise Mining Software [227]. Palacios et al. argue that this becomes a disadvantage in projects with data science problems for which the software is not specifically tailored to [226]. In our Process category, the Data Processing subcategory is the only subcategory which is thoroughly handled by SEMMA. SEMMA also only handles very few aspects of the other categories. This leads to a score of 11% across all categories and 4% for the project relevant characteristics.

Data Analytics Lifecycle

The Data Analytics Lifecycle (DAL) is described in a patent by Dietrich [122]. It describes a process for data analytics projects as well as the according tools for the automation of parts of the lifecycle, especially tailored towards cloud settings and infrastructure. The motivation behind DAL is that data analytics projects are becoming more and more complex due to increasing dataset sizes and more varying structure of datasets. According to the authors this leads to increasing uncertainty in project cost estimation, inflexible solutions and inadequate, costly solutions. They argue that this calls for improved data analytics techniques for business users and data scientists. DAL structures data science

projects into six typical phases which are interconnected in an iterative manner. It describes multiple artifacts which are generated in the lifecycle of an analytics project and describes the roles of a data analyst and business user.

Criticism: DAL does not deviate from most of the other methodologies in its phases and life cycle. However, it is the first methodology to explicitly tie the lifecycle towards cloud settings and to describe software to manage the data analytics lifecycle on a cloud infrastructure. The methodology aims at providing software modules for each of the project phases which support the business user or data scientists in all of its tasks and makes cost and time estimation more reliable. The addition of a data analytics plan as an artifact which is changed across projects is novel. Martinez et al. see this as a benefit, as it can serve as a quality control mechanism [88]. The idea of sandbox environments for testing and pilot programs for deployment are also novel in this methodology. In the Process category, the methodology addresses characteristics across subcategories, but mainly in the Data Processing subcategory and in the Application Evaluation subcategory. The methodology handles some of the characteristics from all artifact types except for software. It provides detailed information concerning the resources required in a project as well as the stakeholder involvement. Lastly, it covers a majority of the Technology and Infrastructure category, except for Explainability and Artifact Management Tools. Across all categories, it achieves a score of 41% and 33% for the project relevant characteristics.

Microsoft TDSP

In 2016 Microsoft introduced their own methodology for data science projects, called the Microsoft Team Data Science Process [7]. Their aim was to create a process which is agile, iterative and helps to deliver intelligent applications efficiently. The methodology emphasizes team collaboration and communication between the roles in a team. The data science lifecycle of TDSP is designed in a way that allows it to be combined or used complementarily with other methodologies such as CRISP-DM A.1.2 or KDD. Similar to these methodologies, TDSP has an iterative nature. While TDSP focuses on projects which aim for the deployment of models, it can also be applied to exploratory projects. Its goal is to move data-science projects toward a clear engagement end point. Each of the phases of the process contains a goal definition, steps for fulfilment and artifacts which should be created during the phase. The whole methodology is tailored towards the Microsoft Azure tool stack and infrastructure.

Criticism: The Microsoft TDSP methodology does not deviate far from the phases and life cycle of former methodologies. Yet, it introduces interesting new ideas which allow for better team coordination and knowledge sharing across projects. TDSP introduces standardized project structures and document templates across projects, which makes it easier to onboard new personnel to projects and to share information across teams and projects. Further interesting aspects of the methodology are the introduction of pipelines, e.g. for data processing, the introduction of a checkpoint to determine whether a project might not succeed and the incorporation of model interpretability and fairness assessments. TDSP provides very clear role descriptions and communication paths. A further strength of the TDSP is that it is closely tied to the tool stack of Microsoft Azure. At the same point this becomes a weakness, as the methodology is harder to implement outside of Azure. This is also criticized by Martinez et al. [88]. The methodology covers some characteristics in the Process category across different subcategories, but mostly in the Data Processing category. It does not cover many characteristics concerning the Management and Governance category, nor in the Artifact and Asset Management category. Due to its integration with the Microsoft Azure infrastructure and tool stack it reaches the maximum score in the Technologies and Infrastructure category. Overall, the

methodology reaches a score of 43% across all characteristics. However, it only reaches a score of 23% for the project relevant characteristics.

4.1.5 Software Development and Operations Group

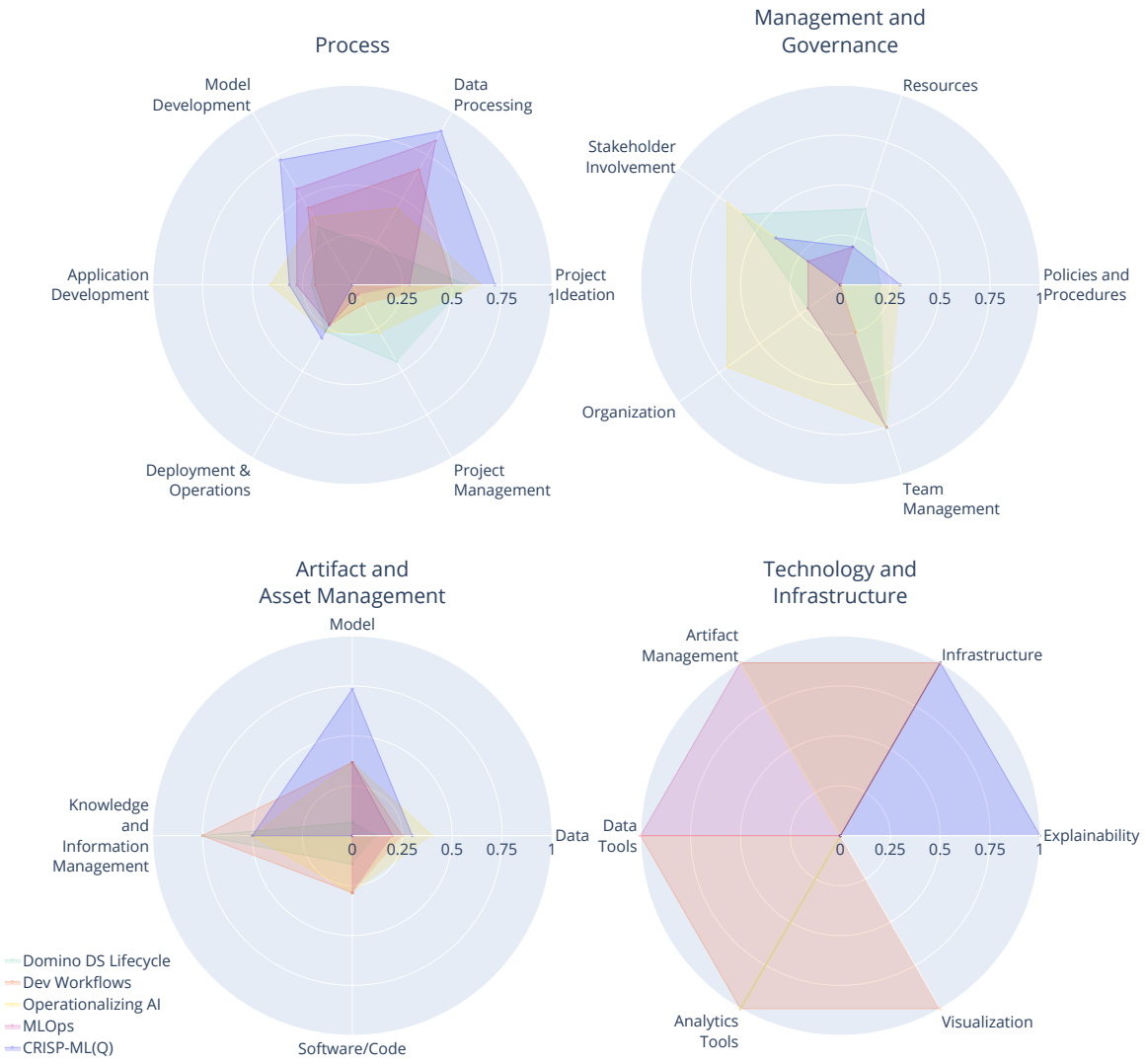

Figure 4.5: Radar diagram depicting the per-category scores of the Software Development and Operations group.

The methodologies in the Software Development and Operations group focus on the complete lifecycle of data science applications, i.e. they also address the operation and maintenance of applications. This is the first group which, at least to a certain degree, covers parts of the application development characteristics and the software/code artifacts, which is apparent in the radar plot in Figure 4.5.

Development Workflows for Data Scientists

In 2017 Ciara Byrne published the Development Workflows for Data Scientists (Dev Workflows) [124]. She argues that there was an explosion of data science interest in various fields but that data science as a field has not yet matured as software development did. In her understanding the core difference between data science and software development is that data science produces insight, while software development produces code. In the data science domain, code is a way to get insights about questions and not usually intended to end in productive systems. However, wrapping the generated insights into code makes them more easily accessible. She distinguishes projects in which it is already clear what to build from exploratory projects in which data analysis and experiments are performed to determine potential directions of a project. Byrne introduces guiding principles for her methodology. She encourages knowledge sharing, as done by Airbnb, via a knowledge repository. This knowledge repo should make data science work discoverable to the whole company, and ideally contains methods of making the knowledge shareable or even subscribable, e.g. via topic tags. Byrne discusses multiple processes, offers an overview of relevant team roles and a detailed list of tools and techniques to use in a data science project.

Criticism: Byrne describes three methodologies, a general data science process and instances of processes established at GitHub and BinaryEdge. She accumulates insights from companies which actively execute data science projects. While the processes she describes do not add many insights over other existing methodologies, she derives some best practices and guidelines which are novel and useful: In accordance with agile principles, Byrne recommends to produce and deliver results fast, even in early phases of a project. She further recommends recording all steps taken to achieve a result, even if the result is negative, to achieve reproducibility. She further encourages holistic thinking, i.e. sharing of code, models and knowledge within and across teams, and gives a knowledge repository as best practice for how to make knowledge discoverable for everyone. The methodologies that Byrne introduces address many of the characteristics in the Project Ideation, Data Processing and Modeling subcategories of the Process category, however only few of the other subcategories. In the Artifact and Asset Management category, she covers some characteristics of all subcategories. The Management and Governance category is almost completely omitted by Byrne. She does however cover Infrastructure, Data Tools and Visualization Characteristics in the Technology and Infrastructure category. Across all characteristics the methodology reaches a score of 31% and 30% for project relevant characteristics.

Domino DS lifecycle

The Domino DS lifecycle was published as a whitepaper by Domino Labs [123]. It uses insights from data science teams and claims to be based on more practical realities than CRISP-DM. The authors argue that data science project management is still very immature, but very much needed by the industry. The lifecycle is built around challenges which were observed in projects from data science teams and a root cause analysis as to what caused these challenges. Key root causes which were identified are insufficient stakeholder management, limited culture of introspection, staffing problems, no culture of iteration and delivery, disconnection from business problems, lack of reusability, sub-optimal organization and tooling. The lifecycle consists of the phases ideation, data acquisition and exploration, research and development, validation, delivery and monitoring. The methodology introduces some roles and technologies and infrastructure for data science projects.

Criticism: A key novelty of the Domino DS lifecycle is its focus on reusability. The authors suggest that not only software components but also artifacts such as datasets should be considered as reusable assets and explicitly define tasks in the beginning of a project to consider what to reuse. Aside of this, the methodology provides interesting insights into the different phases without providing a lot of detail on how to execute them. Martinez et al. praise the integration of data science, software engineering and agile approaches and criticize the more informative than prescriptive nature of the methodology [88]. The methodology introduces five roles, but does not provide many details on the tasks they are involved in. The authors argue that rather than using full stack data scientists, expertise should be split over multiple roles. While no artifacts are defined, the methodology contains a list of useful tools for every phase. The methodology mentions many of the characteristics of the Process category, but does not address them in depth. It does not cover many characteristics in the Artifact and Asset Management category and barely touches on the characteristics of the data artifact. In the Management and Governance category it primarily addresses the Team Management and Stakeholder Involvement subcategories and, in the Technology and Infrastructure category it addresses no characteristics except for infrastructure and analytics tools. Across all characteristics the methodology reaches 31% and 41% for project relevant characteristics.

Data Science & AI lifecycle

The Data Science and AI Lifecycle (DSAL) by John Thomas mostly aims at teams which serve as data science units within a company [47]. Thomas argues that many organizations build data science teams staffed with skilled people but still fail to bring data science projects into production and integrate the results in their processes. He calls for a systematic approach to ”Operationalizing AI”, which requires an end-to-end lifecycle. The lifecycle is split into 5 phases and highlights the involvement of 8 different project roles. Thomas also mentions tools and infrastructure to support the methodology.

Criticism: The Data Science & AI Lifecycle introduces processes and concepts already known to software engineers in a data science methodology. In particular a process of building, tagging for review, reviewing, tagging for deployment and deploying models is proposed, jointly with a model catalog to which a model is pushed. They further introduce the notion of data lineage, i.e. keeping track of where data was obtained from and how it was changed over time. It further introduces the use of continuous integration and deployment tools, model code reviews and model unit tests. The authors emphasize the importance of operationalizing the model, i.e. running, monitoring and managing it, even after deployment. The methodology gives a good overview of roles and responsibilities as well as a long list of tools and techniques. However, it omits the definition of artifacts. Martinez et al. criticize the lack of communication and collaboration guidelines and the missing elements for knowledge sharing and preservation [88]. The methodology addresses many of the characteristics in the Process category. Blank spots are the deployment and operation of applications as well as project management characteristics. In the Artifact and Asset Management category, the methodology provides details about many characteristics of the Model and Data subcategories but omits many characteristics concerning software as well as knowledge and information management, which is in line with the criticism by Martinez et al. Concerning the Management and Governance category, the methodology fails to address the Policies and Procedures subcategory as well as the Resources subcategory. Lastly, in the Technology and Infrastructure category, the methodology addresses most of the characteristics except for explainability tools, data tools and visualization tools. Across all characteristics, the methodology achieves the third best score of 45.4%. Considering only the project

relevant characteristics it achieves a score of 53.3%.

MLOps

Many of the newer methodologies introduce phases and tooling for the automation of phases of the data science lifecycle and the operation of machine learning solutions. Recently the term MLOps has been coined to describe a combination of machine learning and DevOps practices. Kreuzberger et al. have performed a systematic literature review and expert interviews to understand the principles, components and workflow of MLOps [125]. They introduce their own definition of it: "MLOps (Machine Learning Operations) is a paradigm including aspects like best practices, sets of concepts, as well as a development culture when it comes to the end-to-end conceptualization, implementation, monitoring, deployment, and scalability of machine learning products. Most of all, it is an engineering practice that leverages three contributing disciplines: machine learning, software engineering (especially DevOps), and data engineering. MLOps aims at productionizing machine learning systems by bridging the gap between development (Dev) and operations (Ops). Essentially, MLOps aims to facilitate the creation of machine learning products by leveraging these principles: CI/CD automation, workflow orchestration, reproducibility; versioning of data, model, and code; collaboration; continuous ML training and evaluation; ML metadata tracking and logging; continuous monitoring; and feedback loops." [125] Symeonidis et al. also screen current tools and techniques in MLOps and describe it as a collection of techniques and tools for deployment of ML in production with the aim to automate ML processes using practices and approaches derived from DevOps [8]. The ultimate goal of MLOps is to lift machine learning based solutions from a proof-of-concept state towards usage in production environments.

Criticism: The concepts of MLOps are a useful addition to existing methodologies. It addresses the needs of operating ML-driven applications with automation and continuous training, integration, deployment and monitoring. However, MLOps lacks some of the details of other methodologies, in particular with regards to project management topics. Therefore, it makes sense to integrate MLOps with existing data science methodologies. In the Process category, MLOps covers a large range of characteristics across data processing and model development, however it ignores application development and project management characteristics. It also does not address any of the artifacts related to project management and mainly team management characteristics in the Management and Governance category. In the Technology and Infrastructure category, the methodology addresses all characteristics but explainability, analytics and visualization tools. MLOps reaches a total score of 35% across all characteristics and 20% for project relevant characteristics.

CRISP-ML(Q)

CRISP-ML(Q) is a methodology that is centered around the development of machine learning applications [48]. It particularly addresses omissions of the CRISP-DM methodology: (1) changing environments in which models operate and phases for monitoring and maintenance of machine learning applications (2) and quality assurance methods. Quality assurance does not only concern the performance of the application that is to be developed but also the execution of the tasks in the process of creating it. To this end quality assurance methods are integrated in each phase and task of the process model. To the best of our knowledge, this is the only methodology out of the 27 methodologies we analyzed, which explicitly references pre-trained models. Moreover, it suggests utilizing unlabeled

data for pre-training can improve the project outcome. However, the methodology does not discuss further implications of pre-trained models other than that.

Criticism: CRISP-ML(Q) provides a modernized version of the CRISP-DM process and introduces useful additions, e.g. it provides great details about how to select appropriate models and it introduces MLOps aspects. However, it shares similar omissions with CRISP-DM such as the lack of project management activities. In contrast to CRISP-DM, CRISP-ML(Q) does not provide any detailed descriptions of project artifacts. In the Process category, CRISP-ML(Q) provides a lot of detail on the Project Ideation, Data Processing, Model Development and Model Deployment and Operation subcategories. However, it completely omits project management characteristics and lacks details for the application development and deployment characteristics. In the Artifact and Asset Management category, the methodology provides no details on software and only little details on the data artifact. In contrast to CRISP-DM, CRISP-ML(Q) addresses almost none of the characteristics of the Management and Governance category. Finally, considering the Technology and Infrastructure category, the methodology addresses explainability tools and the project infrastructure. Across all characteristics, the methodology reaches a score of 32% and for project relevant characteristics it achieves a score of 36%.

4.1.6 NLU Group

The methodologies in the NLU Group are specifically tailored towards projects which are in the textual domain. Despite their recency, both are from 2021 and newer, they do not address foundation model specific project characteristics. In general, they do not perform very well across categories, which can be seen in the radar plot in Figure 4.6.

Cross-Industry Process Standardization for Text Analytics

Skarpathioataki and Psannis [126] recognized the need for adapting CRISP-DM to projects which deal with unstructured data and proposed the Cross-Industry Process Standardization for Text Analytics (CRIS-TA) in 2021. They emphasize that text analytics brings a lot of potential to organizations but also comes with distinct challenges due to the nature of the data and the unique methods which are proposed to handle it. They identify challenges which are related to the three V's of big data: textual data comes in large volumes, it has a large variety in languages, formats and length and it often requires to cope with a high velocity.

Criticism: The Cross-Industry Process for Standardization of Text Analytics follows the CRISP-DM cycle closely but adapts its phases towards text analytics specific tools and requirements. It further introduces some interesting changes, such as the assessment of a model with regards to scalability, the interpretability-accuracy tradeoff of models and infrastructure and deployment specific requirements, such as deployment on edge devices. It further lists tools and techniques which are specific to text analytics tasks. However, it suffers from the same shortcomings as CRISP-DM, i.e. it does not provide any information on project management and team coordination. It also does not introduce any new roles or artifacts which are specific to text analytics projects. The tools and techniques that are specified in the methodology are rather outdated classical NLP tools. The methodology omits many recent developments in the field which are based on deep learning and does not even mention foundation models. In the Process category, the methodology mainly focuses on project ideation characteristics and data processing characteristics. It provides little to no details on application development, model

Figure 4.6: Radar diagram depicting the per-category scores of the NLU group.

and application deployment and operation and project management. The methodology barely addresses the Artifact and Asset Management and Management and Governance categories. In the Technology and Infrastructure category it only addresses analytics tools. Overall, the methodology reaches a score of only 14.5% across characteristics and 11% for the project relevant characteristics.

Data to Value

In 2022 Leidner introduced the Data to Value methodology [17]. The methodology is explicitly built around unstructured data, supervised learning, ethical questions, big data challenges and early evaluation. It introduces new phases and tasks to the data science lifecycle such as ethics reviews, gold data annotation, system architecture design, patenting and publication and model re-training.

Criticism: The D2V methodology does not provide any details on the execution of the phases

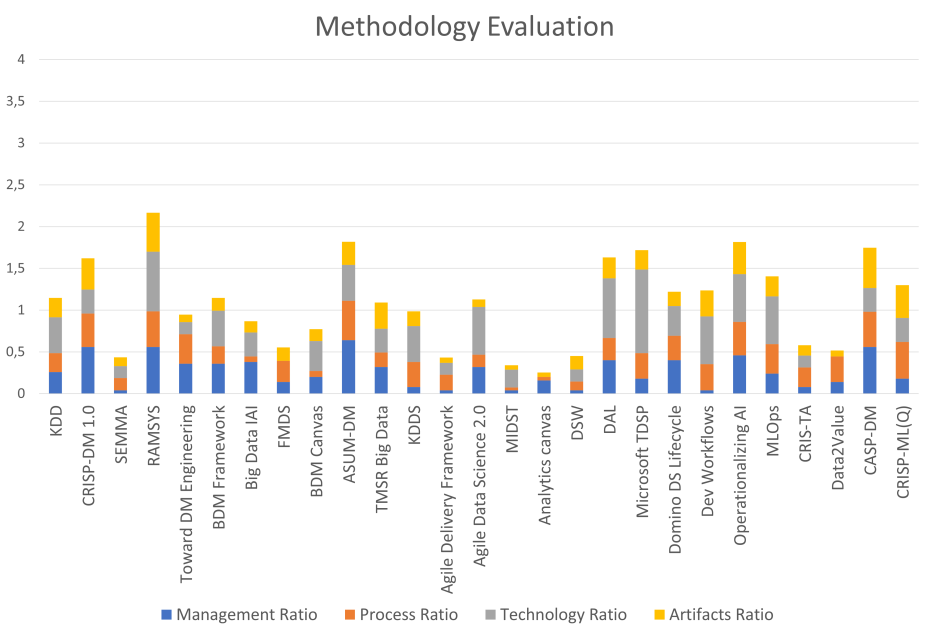

Figure 4.7: Evaluation of all methodologies according to the 163 characteristic categories we discovered with a relative score of up to 1 per category and up to 4 in total.

and lacks roles as well as suggestions for tools and techniques. The proposed choice of either a rule-based or machine learning based paradigm seems illogical, and important aspects such as data and model management as well as knowledge sharing within the organization are missing. Further, the methodology completely disregards recent development such as foundation models. Considering the Process category, the methodology addresses many characteristics but omits important subcategories such as the model development characteristics. The Artifact and Asset Management, Management and Governance and Technology and Infrastructure categories are almost completely disregarded by the methodology. In total, the methodology achieves a score of 13% across all characteristics. For the project relevant characteristics, it achieves a score of 28%.

4.1.7 Comparison with Respect to All Characteristics

Figure 4.7 and Figure 4.8 display the results of the evaluation, centered around the four top-level categories. Figure 4.7 assigns the ratio of points reached divided by possible points, which leads to an ideal score per category of 1 and an ideal aggregated score of 4. Figure 4.8 shows the absolute assigned scores, which can differ per category, as different amounts of characteristics belong to each category. The total amount of points reachable is 326, 50 for Management/Governance, 150 for Process, 14 for Technology and Infrastructure and 112 for Artifact/Asset Management.

These figures provide two different views on our evaluation. The ratio-based scores allow us to quickly see how well the methodologies represent each of our top-categories, while the absolute scores give us a better indication of how many of the characteristic categories were covered and tackled by each methodology. Both ratings have up- and downsides. While the ratio-based rating gives us a better sense of the overall variety of topics covered by a methodology it skews results based on

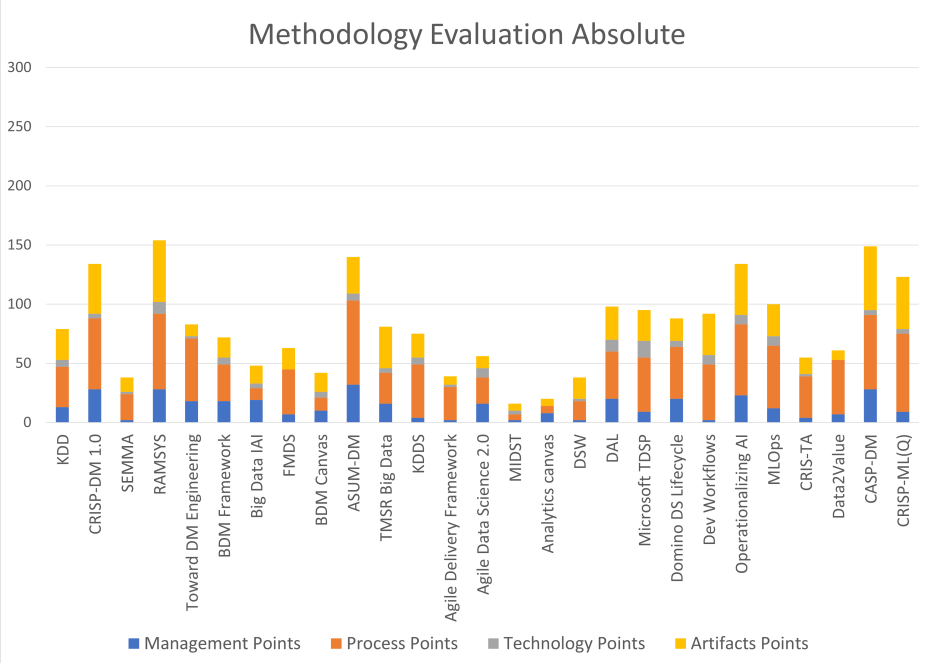

Figure 4.8: Evaluation of all methodologies according to the 163 characteristic categories we discovered with a total score of up to 326 points, out of which 50 can be assigned to Management/Governance, 150 to Process, 14 to Technology and Infrastructure and 112 to Artifact/Asset Management.

the Technology and Infrastructure category for which not many subcategories exist. On the other hand, in the absolute rating methodologies could score better which solemnly focus on the process and omit the other categories. Therefore, we deem it necessary to review both scores and ratings for an overall comparison. In both cases RAMSYS takes the first place with (ratio—total) scores of 2.17—154. This is rather surprising as the methodology is already 20 years old, however it builds upon the popular CRISP-DM and extends it with characteristics covering to Artifact and Asset Management and Technology and Infrastructure.

In the ratio-based rating ASUM-DM scores second (1,82). It is the only methodology to gain top scores in two categories, namely Management/Governance and Process. Operationalizing AI places third (1.81), CASP-DM fourth (1.75) and Microsoft TDSP fifth (1.72). Microsoft TDSP benefits from the complete coverage of all subcategories in the Technology and Infrastructure category, which is no surprise as it is an integrated methodology, i.e. it is tied to the Microsoft Azure infrastructure and tooling. Here, CRISP-DM comes in 7th place, especially suffering from the low score in the Technology and Infrastructure category. Again, this is no surprise, as CRISP-DM intended to be a rather general methodology not prescribing too much to its users.

The absolute scores paint a slightly different picture of the top 5. Here CASP-DM places second (149), benefiting from the strongest score in the Artifacts/Asset Management category and competitive scores in the Process and Management categories. Third is ASUM-DM (140). CRISP-DM and Operationalizing AI (both 134) take a shared fourth place.

It is noteworthy that with RAMSYS and CASP-DM two methodologies are among the highest rated methodologies which directly extend CRISP-DM and take over large parts of the methodology. As

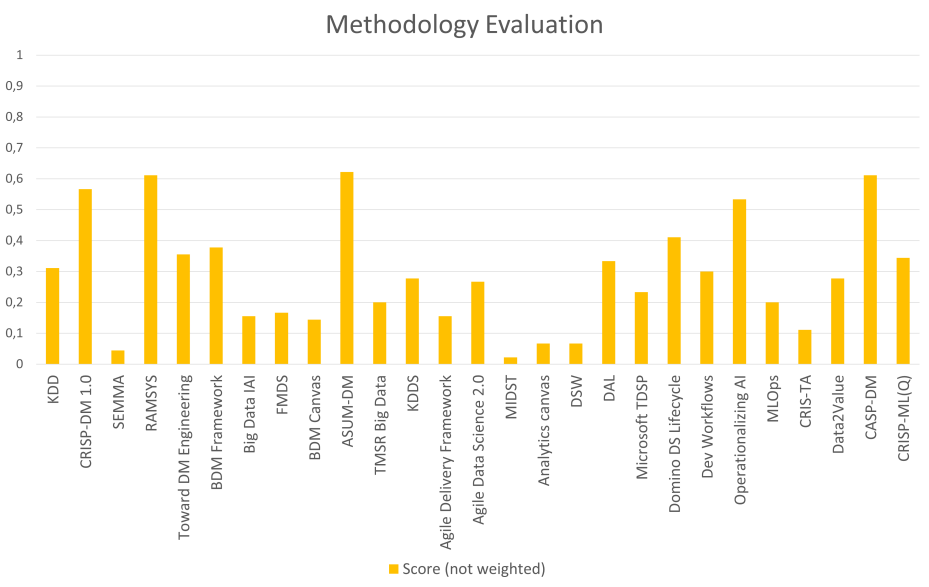

Figure 4.9: Evaluation of all methodologies according to the project-relevant characteristics.

both tailor CRISP-DM to specific scenarios, collaborative remote working and context-awareness, it is not surprising that the more general CRISP-DM is today still the de facto standard methodology for most data science projects. In short: other methodologies might better represent certain aspects of data science projects, but CRISP-DM still provides the overall best general methodology.

Overall, even the highest ratings are relatively low with 2,17/4 or 154/304 points. This leaves a large margin for improvement.

4.1.8 Comparison with Respect to Project-relevant Characteristics

In Subsection 4.1.7 we compared the score with respect to all characteristics. We can do the same considering only the project-relevant characteristics. This should give us a better indication of how well the methodologies would have supported projects in an NLU setting, utilizing foundation models. Now ASUM-DM performs best with a score of 62%, RAMSYS and CASP-DM are tied for second with 61% and CRISP-DM is third with 56%. Figure 4.9 and Figure 4.10 display the results of these project-relevant characteristics, centered around the four top-level categories.

It is noteworthy that the methodologies which score best here are also among the best methodologies when assessed across all characteristics.

Regarding foundation model specific characteristics, only CRISP-ML(Q) mentions the pre-training/adaption paradigm. The paradigm is discussed in terms of yielding potential benefits to increase modeling performance, but not discussed in detail or regarding any larger implications. CASP-DM is the only methodology which discusses the reuse of models in larger detail. However, in the context of adapting the model if the data in projects shows domain shifts.

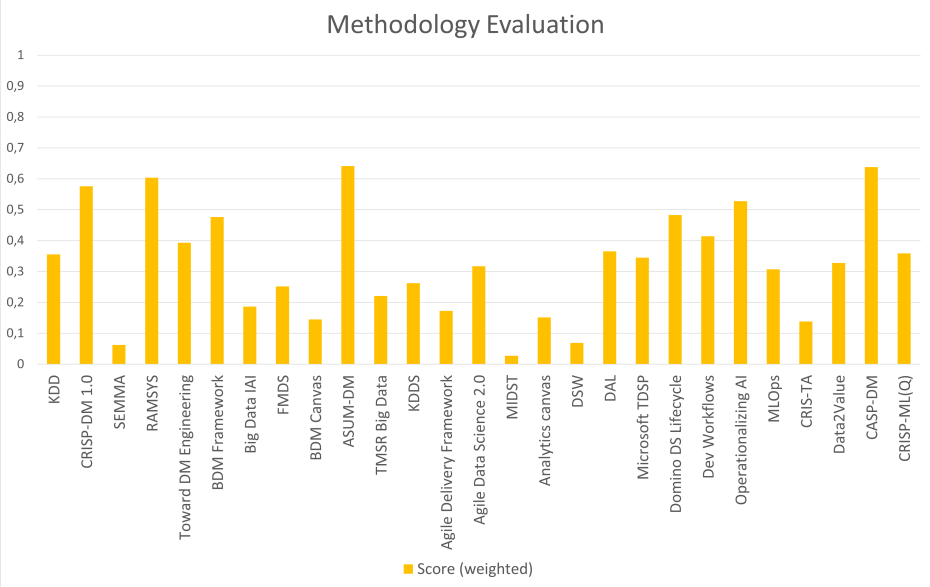

Figure 4.10: Evaluation of all methodologies according to the project-relevant characteristics (weighted by frequency of mention).

Weighted Scores

The scores above do not account for multiple mentions of characteristics. To put more weight on the characteristics which are more frequently mentioned, we multiple the score for each of these characteristics with the number of projects in which the characteristic was mentioned. This way, methodologies can score better if they tackle these specific characteristics, but it also hurts the score of the methodologies which do not tackle them. In the weighted rankings ASUM-DM achieves the best results with 64.1%, CASP-DM is second with 63.8% and RAMSYS is third with 60%. CRISP-DM stays fourth with a 57.6% score.

The scores show us that there is still a large margin for improvement concerning the characteristics which are most relevant to NLU projects.

4.2 Gap Analysis

In the previous section, we focused on evaluating the different methodology groups and the individual methodologies. In this section we look at how well each of the different categories is covered and which gaps there are in existing methodologies.

4.2.1 Artifact & Asset Management

When examining the Artifact & Asset Management category we note that CASP-DM gives the most comprehensive insights into how to handle the characteristics belonging to it. The methodology scores particularly well with regards to Knowledge and Information Management and the Model artifact. However, it omits many characteristics which are important with respect to Software/Code.

Management of Software/Code

In general, most methodologies omit the Software/Code related characteristics. This is an indication that the trend towards not only generating insights from data, but also building functional applications, is not yet covered by existing methodologies. Interestingly, not even methodologies like MLOps or CRISP-ML(Q), which are explicitly tailored towards producing solutions for productive use, cover these characteristics.

Model Management

Further, only few methodologies are concerned with Model Management and aspects such as model reusability, which is of particular interest to foundation model projects. It is noteworthy, that newer methodologies such as CASP-DM and CRISP-ML(Q) have better coverage of characteristics concerning the Model artifact. None of the methodologies discusses the serving of machine learning models.

4.2.2 Process

The ASUM-DM methodology scores highest in the Process category. It is particularly noteworthy that ASUM-DM covers every subcategory belonging to Project Management, something which is often regarded as a blind spot of methodologies such as CRISP-DM [104]. Almost all methodologies score high with regards to data processing and model development. Fewer handle characteristics of deployment & operations. We also find that many methodologies only consider the deployment of models and not the deployment and operation of applications.

Application Development

In general, application development is the subcategory, which is most scarcely represented, indicating that the methodologies do not put the data processing and modeling activities in the context of software/applications that embed them. Only the Data Science & AI lifecycle covers more than 10 of the 27 subcategories which belong to application development.

Expectation Management

It is further noteworthy that only the ASUM-DM methodology covers the subcategory of Stakeholder Expectation Management. This is rather unexpected as this challenge was frequently mentioned in the publications we discovered in our systematic literature review. It is frequently mentioned as a challenge and success factor of projects.

4.2.3 Management/Governance

ASUM-DM is the top methodology with respect to Management/Governance. It covers most of the subcategories. Almost all methodologies cover characteristics discussing the involvement of stakeholders and team management related characteristics. ASUM-DM is the only methodology to include the maturity and readiness of an organization in their considerations.

Policies and Procedures

Generally, most of the methodologies omit the Policies and Procedures subcategory. Only Data2Value and CRISP-ML(Q) discuss ethical issues. No methodology addresses the issue of sustainability which is particularly important when it comes to pre-training foundation models. Further interesting insights are that only CRISP-ML(Q) addresses societal impacts.

4.2.4 Technology & Infrastructure

In this category Microsoft TDSP reaches the maximum score as it provides suggestions for all types of tooling and infrastructure questions. Not surprisingly, Microsoft TDSP and the Data Analytics Lifecycle, which is the second-best methodology in this category, both belong to our Integrated Methodologies Group, which contains methodologies that are bundled with software or infrastructure. Some methodologies, such as the Foundational Methodology for Data Science and Data2Value, do not tackle any characteristic of the different subcategories. The most frequently addressed subcategories are the Infrastructure subcategory and the Visualization Tools subcategory.

4.3 Conclusion

Our assessment of project methodologies shows that most of them address only a fraction of all characteristics in our catalog. Even more importantly, none of the methodologies addresses more than 62% of the project-relevant characteristics. The methodologies all show gaps with respect to general characteristics of data science projects as well as foundation model specific characteristics.

Considering the foundation model specific characteristics, only the CRISP-ML(Q) methodology mentions the novel pre-training and adaptation paradigm. However, only with regards to a potential performance boost compared to non-pre-trained models. None of the methodologies comprehensively tackles the particular characteristics with regards to model and data management and only CASP-DM addresses the potential of model reuse.

This highlights that existing methodologies are not suitable for foundation model projects, which validates our hypothesis that a novel project methodology is required for foundation model usage.

The results further show that existing methodologies do not only leave a gap with respect to foundation models. Important aspects such as the development of data science applications, their deployment and operation are not sufficiently tackled. This shows that these methodologies fail to address the practical realities and expectations of customers. We expect that the development of applications which embed foundation models will play an even more important role in the future. The possibility to embed already available models or simply make calls to models via REST-API opens the possibility for companies to use foundation models, even without any modeling effort.

In the next chapter, we will present our own methodology which is tailored towards the characteristics in our catalog and addresses the shortcomings of existing methodologies.

CHAPTER 5

FM-DSM: A Data Science Methodology for Foundation Model Projects

In Chapter 2 we have shown that foundation models and their adaptability have led to a new paradigm in machine learning. Based on this observation we studied the requirements of a foundation model methodology in Chapter 3 and derived a catalog of relevant project characteristics. We examined existing methodologies and found that they do not account for these characteristics. In this chapter we propose our own methodology, which builds upon the strengths of existing methodologies and extends and modifies them to account for recent developments. In particular, our methodology is designed to account for the aforementioned catalog of characteristics and close the gaps of existing methodologies.

In alignment with other methodologies, we will structure our methodology into distinct phases, which are further broken down into subphases and activities. It does not deviate far from CRISP-DM on the level of phases. We introduce novel phases for the development of applications and their testing. We extend the Business Understanding phase to a Mutual Understanding phase and introduce a phase for the Maintenance and Operation of solutions. Another, larger change compared to CRISP-DM is the introduction of artifact hubs for models, data, knowledge and software. We embed projects into a stream of projects of a team and argue that artifact management is a project-overarching activity. Similarly, we introduce a continuous process of aligning projects with team strategy, governance and coordination activities.

While our intention is not to provide an exhaustive to-do list, we aim at guiding the reader through critical steps and highlight aspects that are uniquely different when utilizing foundation models, in particular LLMs. We will outline the phases of our methodology, accompanied by descriptions of the roles involved in each phase, potential tools and techniques that can be employed, and the artifacts that are utilized and generated throughout the phase. For each phase we provide a rationale, which highlights the project characteristics that are addressed. Generally, our methodology does not follow a linear structure. Many activities can be started simultaneously, therefore we highlight dependencies between activities.

Our methodology is centered around the usage of foundation models. On the level of phases and subphases our methodology presents a general framework for dealing with arbitrary foundation models, i.e. of any modality. However, we present task details, tools and techniques, etc., specific to NLU projects. We argue that this creates only minor adaption overhead and helps to provide a more concrete understanding of the implementation of our methodology.

Our methodology is based on guiding principles which we introduce in Section 5.1. The guiding principles embrace holistic thinking, end user involvement, visualization in all phases of a project and finally frequent deliveries of results. In Section 5.1, we provide a high-level overview of our methodology. In Subsection A.25 we provide an comparison of our methodology with other methodologies. The artifact management infrastructure is described in Subsection 5.2.2, followed by our proposed project lifecycle in Subsection 5.2.3. We describe the roles involved in a typical data science project in Subsection 5.2.4. Our selection of roles is informed by existing methodologies, but we also supplement them with additional roles. A detailed description of all our phases, subphases, their according tasks, the active roles and relevant artifacts is provided in Section 5.3. We conclude the chapter with a discussion of our methodology in Section 5.4.

5.1 Guiding principles

Our methodology is built around principles which guide its design. These principles affect every phase of our methodology and are specifically tailored towards the characteristics of modern data science projects.

5.1.1 Holistic Thinking

In their analysis of current challenges in data science projects, Martinez et al. call for a holistic approach to data science projects [88]. We argue that the use of foundation models amplifies the need for holistic approaches to project management. This is particularly evident when we focus on the artifact/asset management project characteristics such as model management, knowledge and information management, data management, software management and the reuse and adaptation of existing models, which are all not limited to a single project. They require thinking across projects. We will call this holistic thinking. This means that the goals of a project team should not only address the project they are currently working on, but also an overall team strategy.

Why Pre-Trained Models Call for Holistic Thinking

Due to their transfer learning capabilities, pre-trained models are inherently built for holistic project management. The ability to reuse and adapt models to different tasks and domains offers an incentive for cross-project model and data management. Foundation models benefit from pre-training them on large amounts of data, hence it makes sense to build a data hub with all data that is available and legally usable for pre-training. A benefit over only storing models is that with the availability of the data, models with new architectures can still be trained at a later stage. Building a model hub is also a clear benefit, as the reuse of models can provide a kickstart for new projects. Ideally, models are already suitable for a specific task and domain and can simply be reused without further efforts. If not, they can be adapted towards domains or tasks via fine-tuning. For generative models, it might even be sufficient to engineer prompts for novel tasks without specific fine-tuning. Research shows that domain adaptation benefits the performance of foundation models. A viable strategy could hence be to build general foundation models as well as domain-specific and task-specific derivatives of these models or appropriate adapters. A model hub should not only contain the model artifact but also related information, such as information about the dataset the model has been trained on, links to training scripts and information about important model metrics.

Reusing Software

Holistic thinking is not limited to data and models, it also promotes reusing software components. A possible way of designing such components is to focus on building solutions for similar tasks. For example, if a team works on multiple projects which require information extraction methods, it would help to reuse the same software for training and inference across these projects. It might even be possible to share the pre- and post-processing logic in task-specific components. Continuously developing such components over the course of projects with appropriate development processes will help to minimize hidden debt. With recent developments in the space of foundation models, designing such components becomes increasingly easy. Most transformer-based models are built using the Huggingface Transformers library, i.e. models of this type can easily be exchanged, while maintaining the same codebase. Another benefit of having such components is that it eases the integration of the components into other systems. At the start of a project a team could already deliver a preliminary version of such a component to a customer without the final models. The customer could use these components to integrate them in their systems or to gain experience in using and calling the component. Developing components across projects requires roles such as a product owner, which oversee the development and schedule and prioritize change and feature requests.

Sharing Knowledge

Knowledge should be documented and made easily accessible to the whole team across projects. Knowledge refers to any valuable insights that a team gains in projects and outside of them. Examples for such knowledge are insights about experiments, the approaches that a team has explored, general best practices, mistakes that the team made and documentation of the software they developed.

Infrastructure Implications

To enable holistic thinking, teams need to set up infrastructure, roles and processes for sharing their project artifacts. This can be done in the form of central hubs for knowledge, software, models and datasets. We build upon the ideas of the RAMSYS methodology [111], which proposes to have an information vault, containing information about the project goals, data and preprocessing decisions, evaluation criteria and results, modeling decisions and models. We extend this idea to a shared data hub, model hub, knowledge hub and software hub. All of these hubs should be used across projects. The content of the hubs should be made available to all team members across projects, as long as this does not break compliance rules. The benefits are obvious, as ramping up new projects and producing outputs will require less effort. However, additional overhead is introduced through the processes in which these storages are managed, as well as by establishing the roles which are responsible for them. Using such hubs implies cultural changes in teams, as they should embrace a culture of sharing, communicating, exploring and reusing. We give a more detailed description of the hubs in Section 5.2.2. In addition to sharing project artifacts we suggest to use a common project structure across projects. This eases the onboarding of new project members and enables team members to look for relevant information across projects. The Microsoft TDSP methodology shows how such a project structure could be set up [7].

Embedding Holistic Projects in a Team

Lastly, projects should be embedded into the strategy and vision of a team, i.e., projects should not only pursue individual goals but also the advancement of an overarching team strategy. A team lead should oversee the projects of the team and foster synergies in terms of model reuse, data and knowledge sharing and software reuse as well as collaboration on the development of software and models. To achieve this, project leads should discuss the strategic goals of a project with their respective team leads and the project team. Strategic goals can affect the selection of projects and the approaches taken by the team. Unifying infrastructure, artifact management and processes across teams is also beneficial in cases where projects need restaffing. Changes in personnel is a common challenge in projects and creates onboarding overhead. This is mitigated via our holistic principles.

5.1.2 Involve end users and domain experts in every phase

In Chapter 3 we have shown that end user and domain expert involvement is an important success factor of projects. Many methodologies suggest involving them during early stages of business and data understanding as well as during evaluation. However, we argue that, if possible, they should be involved throughout a project. There are multiple reasons why it makes sense to keep them close to the project during all phases. They can

1. answer questions about the domain and the problems which they encounter
2. help to improve the understanding of business and data
3. give valuable feedback during data exploration
4. provide input for modeling
5. support debugging models and give insights into problematic samples
6. help with the evaluation
7. provide valuable input regarding the acceptance of a solution.

Involving them and assuring transparency about the project progress, increases their acceptance of the solution and helps with expectation management. In our phase descriptions we will highlight the involvement of these roles.

5.1.3 Embrace Visualizations

In Chapter 3 we showed that visualizations and visualization tools are considered important project characteristics. They do not only support the discussion of results with a customer but can support various phases of data science projects. For example, during project acquisition, they can help customers to better understand the capabilities of foundation models. During development they can help data scientists to debug models, and during evaluation they ease the interpretation of results. We argue that whenever possible, visualizations should be used to support data science projects.

5.1.4 Deliver and verify frequently

An agile approach to project management is seen as a key success factor of data science projects. One of the principles that agile emphasizes, is taking an iterative approach to projects and an early and frequent delivery of results. Results can vary depending on the phase a project is in. In earlier phases, a result could be a problem understanding report or insights about the data. Later, a result could be a metric. Nonetheless it makes sense to discuss these results with the client and especially their domain experts. This can avoid misunderstandings and misconceptions.

Delivering Software

If possible, it is beneficial to deliver software to the client early on in projects. This has the benefit of accelerating the integration of components in their system. If no models are available, the client can test the integration based on a dummy model. If a model is already available in early stages, the client can also begin testing the component and gain initial insights into the acceptance of the users and the overall user experience. However, this requires a clear understanding on the client side that these models do not indicate the final prediction quality and are subject to change. If this is not communicated clearly, this can lead to disappointment early in the project. Overall, an early deployment of a solution, even if it does not contain the final models, helps identifying issues in a timely manner. Data science projects tend to take a waterfall-like structure [6] in which a deployment is not performed until the end of a project, when it is already too late to make bigger adjustments. Components which wrap models and can be reused across projects facilitate early deployments, as the team minimizes the implementation effort. Ideally, the client can deploy the component early on and receive updated models over the course of the project. This also has the benefit of enabling the client to test and implement integration with their system right from the start of a project.

Work Culture

Establishing a culture in which results are delivered frequently and transparently can significantly improve the communication and collaboration between the project team and the client. It helps to build trust and gives the client a sense of control over the clearly uncertain nature of data science projects. Further, it allows for rapid re-adjustments when challenges occur.

5.2 Methodology Overview

We will now present a high-level overview of the phases of our methodology and its main components. Further, we will contrast the phases to other methodologies and emphasize its differences to them. For each phase, we will briefly highlight how the phases are impacted by our focus on NLU and foundation models in general, but we will also explain general differences to other methodologies. The phases and main components of the proposed methodology are depicted in Figure 5.1.

A key property of our methodology is that projects interact with artifact stores which are managed across multiple projects. We call these artifact stores ”Model Hub”, ”Data Hub”, ”Knowledge Hub” and ”Software Hub”. We describe them in more detail in Subsection 5.2.2.

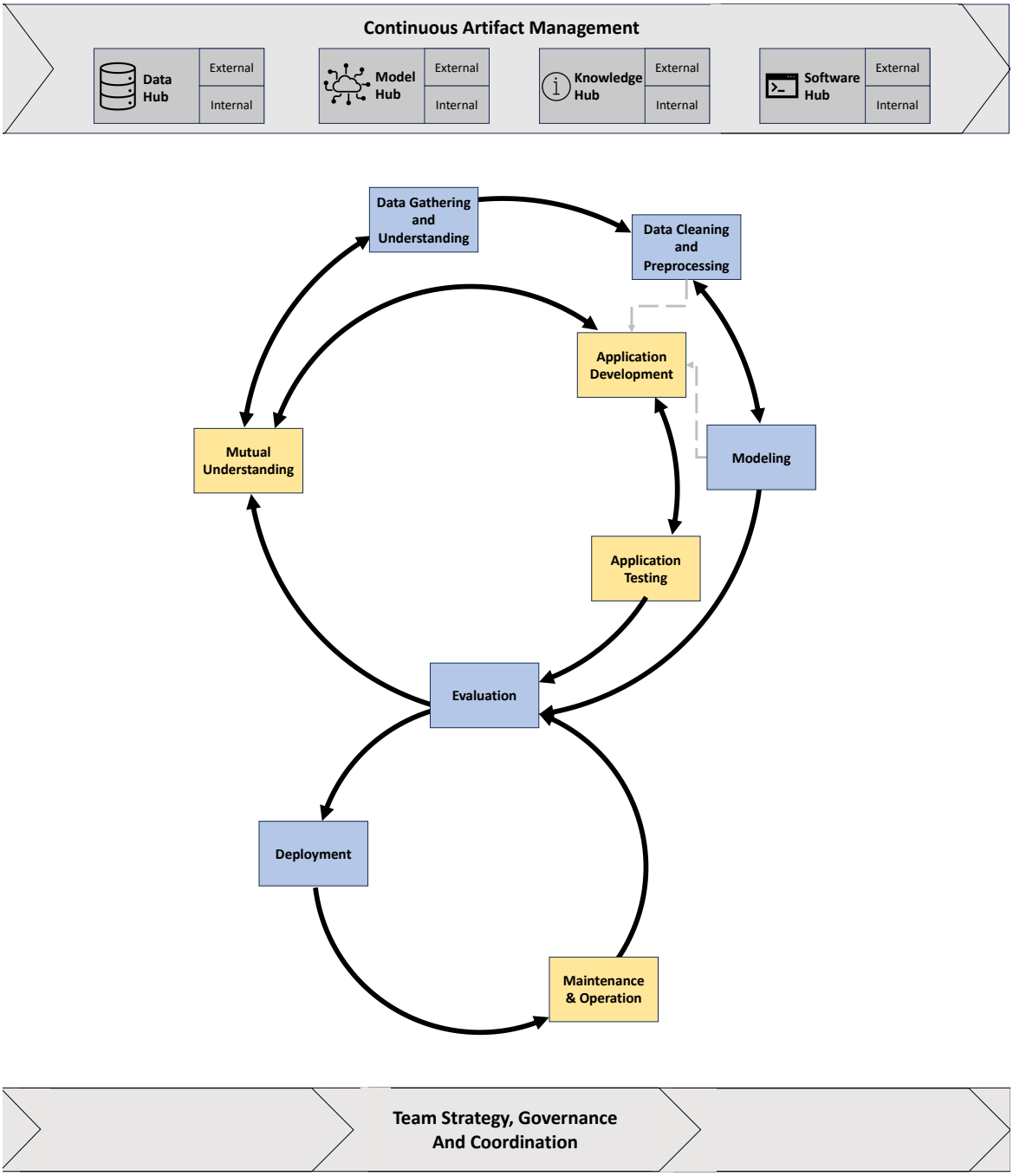

Figure 5.1: A high-level overview of FM-DSM. The methodology extends CRISP-DM with phases for application development, application testing as well as their maintenance and operation. It alters the business understanding phase into a mutual understanding phase. Integrating model pipelines, data pipelines and model artifacts in the application creates dependencies between Application Development and other phases. Further, it embeds projects in continuous, project-overarching processes for artifact and team management.

5.2.1 Phases of Our Methodology

We subdivide projects into 9 distinct phases: ”Mutual Understanding”, ”Data Gathering and Understanding”, ”Data Cleaning and Preprocessing”, ”Modeling”, ”Application Development”, ”Application Testing”, ”Evaluation”, ”Deployment” and ”Operation and Maintenance”. Each of these phases is further subdivided into subphases, which contain activities. For each of the phases we highlight which artifacts are used and produced in them, which roles are involved and which tools and techniques are helpful.

Comparing FM-DSM to CRISP-DM

Compared to CRISP-DM we extend our methodology on both ends of a project. We stretch our first phase ”Mutual Understanding” such that it also includes the initial acquisition of a project. Our reasoning here is that during the acquisition a team already obtains relevant information for project planning and further business development can set the expectation of customers right from the start. We name the phase ”Mutual Understanding” and not ”Business Understanding” as we emphasize the need to also create transparency of the capabilities of foundation models and solutions on the customer side. A successful project not only depends on the business understanding of the data science team but also on the understanding of the chances, limitations and implications of introducing data science solutions in a business. Further we extend our methodology beyond the deployment phase and introduce a ”Maintenance and Operation” phase. As we have discussed earlier, the goal of data science projects is often to bring applications into production, which raises novel challenges such as monitoring, data-drift, retraining, etc. This phase is heavily influenced by the recent development of MLOps. Another notable modification is that we introduce an ”Application Development” and an ”Application Testing” phase in our methodology, which add software development and testing tasks. Note that the Modeling and Application Development phases can be traversed in parallel, however there are dependencies between them. The development of the application and its testing is dependent on the availability of models that should be embedded. Foundation models can significantly accelerate development cycles, as teams can rely on already existing models before the new models are finalized. Software or application development was not tackled by CRISP-DM but plays a crucial role in today’s projects. Many characteristics, which we analyzed in the previous chapter, specifically address the software development processes in data science projects and the integration of machine learning based solutions within traditional software. A comparison with other methodologies can be seen in Figure A.25 in Subsection A.4.1 in the appendix.

5.2.2 Artifact Management Infrastructure

A core feature of our methodology is that artifact management is not limited to a single project but regarded as a project-overarching process, i.e. the life cycle of artifacts can span across multiple projects. Figure 5.1 sketches the flow of artifacts throughout projects. Artifact hubs are filled with artifacts from former projects. Current projects reuse those artifacts, potentially modify them and add new artifacts to the hubs. New projects then make use of the already existing artifacts. We are not the first to recommend central repositories for sharing and reusing. The RAMSYS methodology introduced the concept of an Information Vault [111], while [124] described the use of a Knowledge Repo at the company ”Airbnb”. The hubs in these methodologies focus mostly on sharing information and best practices, as well as providing platforms for discussions. We extend their concepts to all types

of artifacts and introduce hubs for data, software and models. Further, we follow the idea discussed by Byrne et al., to establish review cycles and processes for these Artifact Hubs in order to maintain a high quality. They suggest that this requires a short-term increase in workload but yields a long-term decrease. Thomas et al. [47] also suggest establishing model repositories.

Model Hub

The Model Hub contains models, their metadata and experiment results and makes them accessible to the team. Its purpose is to store models, make them reusable, searchable and easy to deploy. In short, the model hub combines the concepts of model stores, model registries and serving solutions. The adaptability and reusability of pre-trained models naturally promotes the use of model hubs. It should be a strategic decision of data science teams to treat models as artifacts which are used across projects. The benefits of having easy access to foundation models are evident:

1. If a suitable model is already available, the modeling phase can potentially be reduced to reusing a model or simply calling an API.
2. Adapting a pre-trained model significantly reduces modeling effort compared to training a model from scratch.
3. Models pre-trained on large amounts of data offer accuracy benefits over models without pre-training.
4. With more and more training data available models can be improved over time.
5. Appropriate models can be used for demonstration purposes.

Model cards offer a good way of describing and documenting models [190]. They provide important information about the model itself, its intended use, factors such as demographic groups, metrics, details on the evaluation and training data, quantitative analysis, ethical considerations and caveats and recommendations.

In addition to hosting their own model hubs, teams should use and potentially even contribute to public model hubs such as Huggingface [84]. As of the time writing, Huggingface contains over 250,000 pre-trained models. They are categorized by tasks, languages and the data they were trained on. Further, Huggingface also provides model card descriptions of all their models.

- Popular public hubs are: Huggingface, PyTorch Hub, TensorFlow Hub
- Model versioning solutions: Neptune.ai, MLflow, DVC, ModelDB
- Model serving solutions: Tensorflow Serving, TorchServe, BentoML, Cortex, Triton Inference Server, ForestFlow, Kserve
- Model registry solutions: MlFlow, AWS SageMaker Model Registry, Microsoft Azure ML Registry, Neptune.ai
- Experiment tracking solutions: Neptune.ai, Weights & Biases, Comet, Sacred, MLflow, TensorBoard
- ML Metadata Stores: Kubeflow Pipelines, AWS Sagemaker Pipelines, Azure ML, IBM Watson Studio, MLflow, Tensorflow ML Metadata

Data Hub

Data management is just as important as model management for data science projects, in particular when using foundation models. The performance of foundation models is correlated to the amount of training data used during pre-training [69]. It has even been shown that smaller to medium-size models can perform as good as very large models, if they are trained longer and on more data [161]. Further, adapting foundation models to domain-specific data leads to better models [73].

A data hub can serve as a central repository of data in which all of the datasets, available to a team, are stored. The datasets should be versioned and stored with additional information, such as the format, data domain, the source, data ownership, licenses and if applicable the type of labels the dataset contains. To assure reproducibility any scripts for obtaining and preprocessing the datasets should be associated with the datasets and data should also be kept available in its raw form whenever possible. If compliance allows it, all teams should have access to the datasets in the data hub.

A role such as a data architect or data master [111] could be responsible for administration of the hub and take on tasks such as the selection of new data sources, quality assurance and so on. If permissible, the individual projects of a team should contribute to the data hub.

Similar as in model management, teams can also use external data hubs. Prominent examples are Huggingface datasets, but also the datasets available on websites like ”paperswithcode”.

It is beneficial for teams to agree on a standardized data representation which can be consumed by all the solutions build by the team. This way the overhead of pre-processing is reduced and reusability increases.

When establishing a data hub, concepts and processes for data protection and storage should be established. Important datasets should be stored redundantly, to prevent data loss.

- Popular public data hubs are: Paperswithcode Datasets, Huggingface Datasets, LLMDataHub
- Data versioning solutions: DagsHub DDA, DVC, d0lt, LFS, lakeFS, neptune.ai, pachyderm, DeltaLake
- Data storage solutions: S3 Storage, MongoDB, Minio, HDFS
- Vector databases: ElasticSearch, FAISS, Milvus, Weaviate, Pinecone, Qdrant, Vespa, Vald, ScaNN, Pgvector

Software Hub

Managing software is as important in data science projects as in typical software development projects. Versioning systems should be used to assure code quality and, whenever possible, testing, integration and deployment should be automated.

Teams should share pipelines for data processing, model training and evaluation. Whenever possible software and code should be reused. For larger components it makes sense to define specific product owner roles that prioritize features and are in direct communication with the project leads from different projects and ideally the customers themselves.

The Software Hub should also establish roles and policies for committing code. Code reviews should be established as well as processes for code branches, builds and deployments. This all aims at establishing high software quality standard.

- Popular public hubs are: GitHub, Docker Hub
- Software versioning solutions: GitHub, GitLab, Bitbucket, Git, SVN, CVS, Mercurial, Gitea
- Container Registries: Amazon ECR, Azure Container Registry, Docker Hub Container Registry, GitHub Package Registry, GitLab Container Registry, Harbor Container Registry

Knowledge Hub

Sharing knowledge and information is a critical success factor of data science projects [88, 157]. Hence, knowledge and information management should also be embedded in team processes. The RAMSYS methodology suggests an Information Vault [111] which contains a problem definition, distilled knowledge from related problems as well as information about the evaluation, data hypothesis and model hypothesis for every project. Teams should not only document results but also processes to ease reproducibility. The Domino DS lifecycle emphasizes the importance of also preserving null results [123] as they also provide valuable insights.

Project related information should be made available to every project member, however, should not be limited to them as others can also benefit from the information. Harnessing knowledge obtained in previous projects for future work is an important success factor of projects [141]. A knowledge hub providing such insights and experiences from previous projects helps mitigating the reliance on experienced senior data scientists, which guide the team. Parts of the knowledge hub should also be open to the respective customers, for verification of the information concerning their project and if necessary, providing corrections.

- Popular knowldege sources: arXiv, medium, paperswithcode, Stack Overflow
- Knowledge sharing solutions: Confluence, Google Workplace, Microsoft SharePoint, GitBook, GitLab

5.2.3 Project Lifecycle

We follow the example of other methodologies and suggest an iterative project lifecycle. Our methodology contains multiple cyclic processes. The basic data science lifecycle is similar to those of other projects and contains the phases ”Mutual Understanding”, ”Data Gathering and Understanding”, ”Data Cleaning and Preprocessing”, ”Modeling”, ”Evaluation”, ”Deployment” and ”Maintenance & Operation”. There are transitions and backtracks between all these phases.

In addition to that cycle, there is a cycle for ”Application Development”, consisting of the ”Mutual Understanding”, ”Application Development”, ”Application Testing”, ”Evaluation”, ”Deployment” as well as ”Maintenance & Operation” phases.

Both Cycles share the ”Mutual Understanding”, ”Evaluation”, ”Deployment” and ”Maintenance & Operation” phases. The cycles are not independent. At multiple points during a project there are dependencies such as the integration of data and model pipelines and the models in the application.

In addition, there is the Artifact Management process, which is not limited to a single project. Artifact management is considered to be a project-overarching activity in our methodology.

Finally, there are project management and team strategy processes. The project management process accompanies the complete project, while the team strategy influences project management decisions across projects.

Data Science Trajectories

We follow the idea of so-called data science trajectories, presented by Martínez-Plumed [16] with regards to data science projects and their non-linearity. Not all projects have the same scope and require the same phases, subphases and activities, nor the same path and order in which they transition through these phases. Depending on the setup and type of a project, teams might omit certain phases, or not visit them in certain iterations of a project. On the other hand, some phases might be traversed in parallel, e.g. application development and modeling. Selecting a path through these phases should be something that is partially planned in the beginning of a project but also dependent on the progress of a project.

Our experience shows that a hybrid approach to planning is beneficial, as suggested in [87]. These approaches combine research phases or waterfall-like phases with agile approaches. Research phases could, e.g., be during data exploration, i.e., in phases where it is not easy to predict an end date due to the inherent uncertainty in the data. We further suggest following Jurney's suggestion to arrange even more waterfall-like project phases such as data exploration in sprints and to ship intermediate outputs, e.g., preliminary insights into data [6]. Further, we encourage the usage of a ticket board, even for more open-ended phases.

5.2.4 Role Definitions and Team Organization

The number of people who are involved in a project and the associated roles vary based on the project scope. We follow the proposal in [6] to have project teams in which team members occupy multiple roles. This is particularly relevant for projects with a smaller scope. Jurney argues that involving too many people in different roles in a project creates the "pull of the waterfall", i.e. due to the large number of dependencies across roles, waterfall-like project dependencies occur. Jurney further prioritizes more experienced people over juniors. Staffing projects with seniors is not always a viable option. Hence, we recommend pairing senior team members with junior team members whenever possible.

Teams should establish regular exchanges between their members to enable knowledge and experience sharing and to establish guiding for more inexperienced team members. Meetings can be held with various durations and frequencies. Daily stand ups and weekly jour fixes are common practices in projects. Further, planning and review meetings for sprints can be held every 2-4 weeks. These should also include relevant stakeholders on the customer side. Additionally, project teams should have regular meetings with other organization members to exchange ideas, update them on available artifacts and share their knowledge.

Examining the methodologies we analyzed in Chapter 4, we discover 61 different project role titles. We analyze these roles and cluster them according to their definitions and common activities. As a result, we define 18 distinct roles, which we will present in the following subsections.

Project Lead

There are project leads on the side of the data science team as well as the client side. Their activities include planning and coordination, tracking of deliverables, resources, time and progress [116]. Further activities include project steering, supporting the project team, organizing the project team and communication with the other parties [116]. Depending on the project setup, further roles such as scrum masters could be counted towards the project lead role.

Business Stakeholder

Business stakeholders are involved in various activities. They often champion the project and operate as financial sponsors [2, 116]. They are responsible for setting the scope, defining the objectives, KPIs and the expected ROI [30, 116, 123, 125, 229]. Further, they provide valuable input about the business domain to support the modeling and development processes [2, 113, 115, 123]. They are also included to make strategic decisions and should be involved in assessing the impact of the application on the enterprise and its processes and implementing appropriate changes to the enterprise architecture [113, 114].

Business Developer

Business developers are responsible for gathering market information to guide the team strategy and to find and create sales opportunities [2, 6, 114].

Data Scientist

The role of the data scientist is mentioned in most of the methodologies. Data scientists are responsible for explorative tasks and modeling tasks. The term they have been referred to has changed over time, In earlier, more explorative projects they were also referred to as data analyst [30] or data miner [2]. Their tasks vary from discovering insights from data [2, 6, 30, 123], defining and designing analytical approaches and predictive models [113, 115, 125] and evaluating approaches and visualizing as well as communicating results [6, 115, 116, 123, 229]. In foundation model projects, data scientists have specific knowledge about foundation model related methods, models and technologies. In NLU for example, they should be familiar with the training of large language models and the various NLU related tasks for fine-tuning.

Solution Architect

Solution architects design the solution application and select the appropriate technologies to meet the requirements [7, 119, 125]. Generally, they should have a good understanding of software architecture, the project requirements and NLU technologies.

Software Engineer

The work of software engineers is often complementary to the work of data scientists, i.e. they write the software which embeds the machine learning components [229]. They turn the concepts, which the solution architect envisions, into well-engineered applications, by using design patterns, coding guidelines and best practices for developing ML-based solutions [125]. Often, their solutions are web-based applications [6], requiring appropriate development skills. Further, software engineers are responsible for designing and performing tests of their systems [6].

Requirements Engineer

Requirements engineers interview relevant stakeholders on the client side and obtain a good understanding of the business problems. They translate the business problems into functional and non-functional requirements.

ML Engineer

The role of ML engineers only came up recently [125]. They are responsible for building and operating ML infrastructure, managing automated ML workflow pipelines, CI/CD processes and model and infrastructure monitoring. They ensure the performance and scalability of ML-based solutions [124].

Product Owner

Product owners are responsible for translating business needs into requirements and software engineering tasks [6, 229]. They serve as intermediate layer between customers, users and software engineers. Further, they ensure the quality of their products [123, 124].

Data Engineer

Data engineers are responsible for handling data processing, feature engineering and building respective pipelines for both [125]. They also ensure the connectivity with the respective data stores.

Data Master

The RAMSYS methodology first mentioned the role of a data master in the context of data science projects [111]. According to the authors, the data master is responsible for managing the stored data. [229] defined a data steward similarly, as a role that manages access to data and enforces privacy policies. RAMSYS also gives the data master the responsibility of authorizing and executing data transformations, however we omit this for our methodology as it would introduce a bottleneck, since datasets are used across projects. Nevertheless, it is recommendable to have a person in and across projects who is responsible for overseeing data transformations.

User Experience Designer

The user experience designer is responsible for creating the design and the interaction possibilities of applications, such that the users can interact with the data and the models and interpret the results despite their uncertainty [6].

Domain Expert

Domain experts provide valuable insights for the project team. They give background knowledge about the domain in which the project is set in [30]. Further, they provide relevant information about data and potential data sources [119] as well as processes and requirements [116]. Domain experts also help with building a domain terminology [2].

End Users

The end users are the people on the client side who are using the application [113]. They test the application and provide valuable feedback for improvement and acceptance [116, 117]. The end users are often, but not always, also domain experts.

IT Experts

IT experts support the client and the NLU team. On the team side, they ensure that the infrastructure for training models and managing artifacts is up and running to enable the work of the team members [6]. IT experts on the client side should already be involved early on in projects to support requirement definition, installation and maintenance of software and hardware and administer systems, databases, networks and IT security [2, 116, 117]. Generally they support all technical processes [2, 113].

Legal Experts

Legal experts are not mentioned in the methodologies that we analyzed, however they should be included in different phases of the project to assure compliance to laws and regulation, to oversee contracts and licensing topics.

Auditors

Auditors have various responsibilities in the projects. They assess trustworthiness aspects of the project and the resulting application, such as ethical questions, fairness aspects, security and compliance [124, 229].

Team Lead

Team leads support the project teams with strategic decisions and a vision. They monitor project-overarching processes and provide guidance and direction. They are also responsible for the team organization and staffing of projects.

5.3 Project Phases

We will now introduce the 9 project phases of our methodology. Our methodology structures projects into phases, subphases and activities. All of them are presented with a brief description and should serve as a guidance for important steps and decisions in a project. The activities should not be regarded as a linear task plan. They should rather help project managers to verify which actions to take and whether they thought of everything important. For each phase, we also describe which roles are active, which artifacts are used and produced and which tools and techniques could be helpful. In our phase descriptions we focus on NLU related implementation details, i.e., certain tasks, tools, techniques and artifacts are specialized to the NLU context. However, most of this is easily adaptable to other types of data science fields which require different types of foundation models. We end each phase with a rationale, explaining the importance of a phase, what distinguishes it from other methodologies, especially with regards to NLU and foundation models and finally we show which project characteristics are covered by a certain phase.

5.3.1 Mutual Understanding Phase

NLU projects usually start with knowledge exchange. In contrast to most methodologies, we refer to the first phase not as a business understanding phase [2, 111, 112, 115, 126], but rather a mutual understanding phase. Gaining domain understanding on the development side is a critical success

factor for NLU projects. However, we also deem creating an understanding of NLU at the customer side an integral part of NLU projects. Introducing the customer's business stakeholders, IT experts, domain experts and end users to the capabilities and limitations of NLU and the team's portfolio can help to set expectations right from the start. Customers which have no experience with NLU based solutions might have a severely misinformed view on NLU and overblown expectations. On the other hand, some customers might not be aware of the chances and possibilities current NLU methods offer. Hence, providing an understanding of its capabilities early on might lead to the discovery of unexpected use cases. The early involvement of end users and domain experts can help to set the focus right and prevents misalignment of project ideas and customer needs. Gaining a mutual understanding starts as early as in the acquisition of the project and often lasts until the end of the project. This phase focuses on achieving a good understanding of the business goals, requirements, the data at hand and the data necessary to reach the business goals. It concludes with project planning. In the following we present the subphases of the Mutual Understanding phase. A compact summary of the phase is depicted in Figure 5.2. Detailed descriptions of the roles, artifacts and tools are in Subsection A.5.1 in the appendix.

Acquisition

In the Acquisition subphase a contract between the team, a client and potentially other parties is created. This requires a good understanding of the problem, good cost, time and resource estimates, clarification of legal aspects as well as showing the client the potential of NLU solutions and defining clear measurable goals, outcomes and deliverables. The Acquisition subphase can be regarded as a minor iteration of the whole Mutual Understanding phase. The challenge in this subphase is to create a good estimate of resource requirements and risks without investing too much effort. Following, we list the activities of the Acquisition subphase.

- **Initiate Contact**
  Gaining a mutual understanding about NLU, the business problem and its requirements starts during the project acquisition. Customers might have a business need which leads them to their in-house NLU teams, NLU business offers or the customer might be approached by an NLU sales team.

- **Introduce customer to NLU potentials and gain initial business problem understanding**
  Although NLU is becoming increasingly popular due to models such as ChatGPT, many customers have little to no knowledge about its capabilities and limitations. Ideally, business developers can provide a first idea about the solutions at hand and the technology in general. Realistic business goals, which can potentially be reached with NLU solutions, are identified in a dialog with the customer. The job of business development is not only to generate interest on the customer side but also to collect first information for the team. They should provide the team with an initial understanding of the business goal, and the customer with an initial assessment whether existing solutions are suitable for the business goal. Further, business development should get an overview of general requirements on the customer side, such as the available data, infrastructure and so on.

- **Verify project alignment with team strategy**
  Given the information from business development, technical team members such as solution

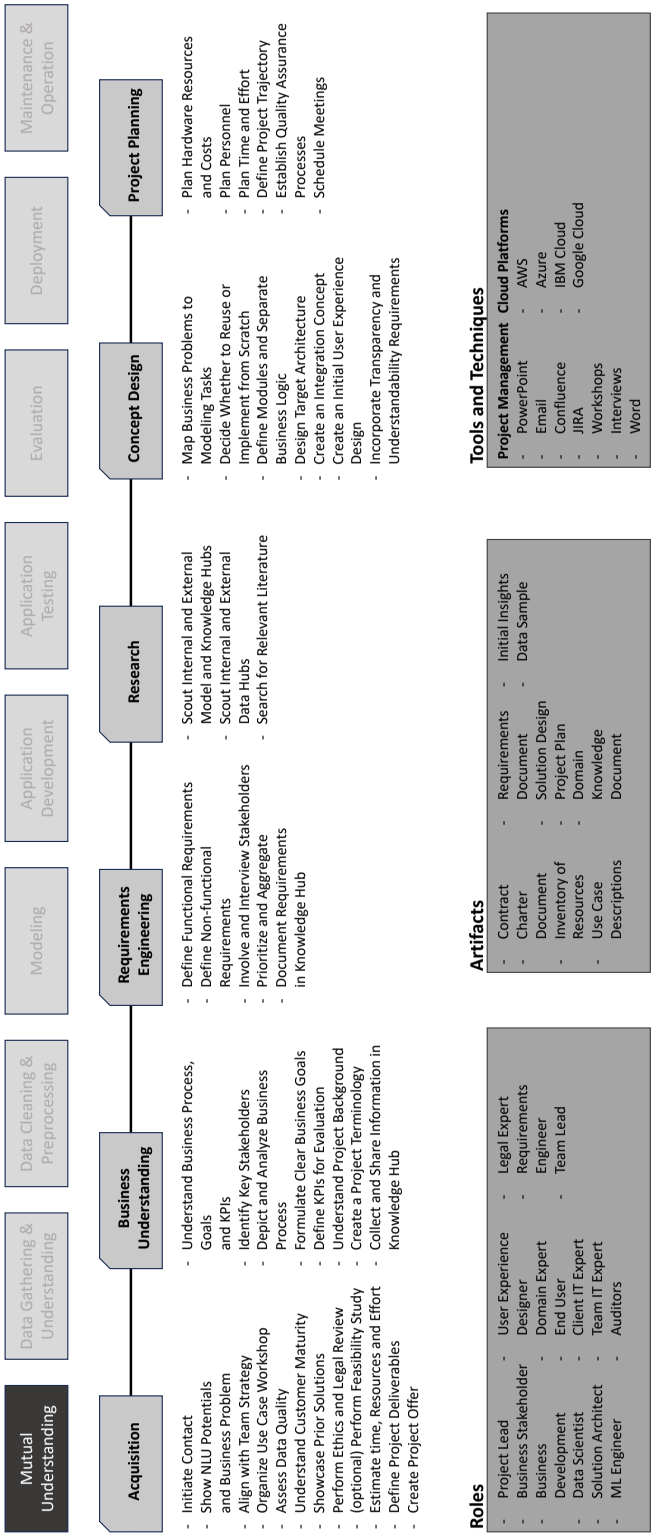

Figure 5.2: Overview of the Mutual Understanding phase.

architects and data scientists can perform an initial assessment, relate the potential projects to past projects, assess the fit to existing solutions, the experience of the team members and the fit to the own team and organization strategy.

- **Organize use case workshops with relevant stakeholders from both sides**
  If the team decides that a project might be suited, they should aim at building a more complete understanding of the business goals and requirements. To this end workshops between the customer side and the technical side should be held, in which potential use cases are discussed on a more technical level. Ideally, managers, domain experts, end users and IT experts from the client side are involved as well as data scientists and ML engineers. Prior to the workshop a questionnaire should be prepared and handed to the customer to make sure that all important questions are addressed. If the goal of a potential project is to integrate an NLU solution into an existing system or process, a live demo by the customer is recommendable. The demo should serve as a basis for discussions and for clarifying requirements. Use cases which are identified in this step should be documented for further analysis regarding costs and benefits. Other important factors to consider for prioritizing use cases are the environment in which the customer operates, their competitors and competing solutions.

- **Make an initial assessment of the data and its quality**
  The NLU team should get an introduction to the available data, i.e., the documents or texts which will be analyzed in the projects and other data that is available, such as structured data from database exports. This is an important step in order to understand the complexity of the business problem and especially the quality of the data, arguably one of the most important characteristics of NLU projects. Generally, the team should gather as much information as possible about the data. Information should cover the volume of the data, whether labels are present, the label distribution, data types and the sources.

- **Understand customer maturity and readiness**
  Introducing AI based solutions in businesses can have a large impact on existing processes and the employees and often requires a change in company culture and large implementation efforts. The team should hence spend some time assessing the maturity and readiness of the customer with regards to integrating AI based solutions in their work and capturing data in their daily processes.

- **Showcase prior solutions, ideally with customer data**
  If earlier work of the NLU team has tackled the same or a similar business problem, e.g., information extraction from medical documents, a live-demo of the capabilities of such a solution should be given to demonstrate its functionality and capabilities. Ideally, these demos are visual and easy to understand. This helps the customers to understand the functionality and the quality of results they can expect. The demonstration can also be done with the data of the customer. This increases the trust and facilitates the understanding on the customer side. It makes sense to involve the IT experts of the customer in these activities to discuss output formats and interfaces or any related technical information relevant for integration, deployment and operation.

- **Perform review of ethical and legal aspects**
  At this stage of the project, we recommend following the suggestion by Leidner to perform

an ethics review [17]. Ideally, companies establish guidelines to decide whether a project is acceptable from an ethical standpoint. Various ethical questions should be raised, including whether a solution could be misused, the societal impacts of a solution as well as sustainability concerns. Pre-trained language models make this more important as they offer a lot of potential for misuse, have high-energy consumption during pre-training and can be biased based on the training data. It might be beneficial to have a dedicated role for such questions, [139] recommend establishing the role of an ombudsman, who can serve as ethics auditor. In addition to the ethical questions, legal questions might be of concern. Data used in the project might be particularly sensitive and protected by law. It is commendable to involve legal experts, information security auditors and IT experts in this stage of a project, to understand the legal requirements and to make sure that the necessary measurements and infrastructure for compliance can be provided. In certain scenarios, e.g., in hospitals, it might also be necessary to conduct ethics reviews on the customer side. This can be a showstopper and should ideally be done before the project begins, as the result of the review could imply that the client is not allowed to use data for project relevant use cases.

- **(optional) Perform feasibility study**
  One goal of the acquisition subphase is to obtain as much information about the business problem and the requirements as possible before the actual project commences, in order to estimate the effort, time and resources needed to run the project as accurately as possible. If there is still a lot of uncertainty, a possible solution might be to reduce the scope and start with a proof-of-concept or feasibility study. Similar to [17], we view a proof-of-concept as a trimmed down version of a project, comparable to a single iteration through the project lifecycle. However, proof-of-concepts typically do not end with a deployment but rather with an evaluation of models to proof that a problem is solvable with a sufficient quality. To reduce effort during this feasibility study, the team could resort to limiting the analysis to a subset of the data or a subset of problems that need to be solved. The team should focus on producing reusable results for the follow-up projects, potentially in form of a usable minimal viable product (MVP). Since this requires significant development effort, reusing existing components for the MVP drastically reduces overhead. This could include data processing and modeling pipelines. Another clear benefit is that customers already get something more than a presentation of results and might be more likely to commission the team for a project.

- **Estimate time, resources and effort**
  When estimating the effort for a project, all relevant stakeholders on the NLU team side should be consulted. If the team decides to reuse an existing component which might need to be extended for the project, software engineers who developed the component or, if available, the product owner of this component should be included. The team lead should verify the alignment of the project goals with the team strategy and vision. Potential risks which will influence the true effort should be documented along with respective contingency plans.

- **Define project deliverables**
  Before a project offer is made, the client and the team should jointly discuss the scope of the project and the desired outcome as well as the deliverables of the project. This can serve as validation point to see if the goals of the project represent the true goals of the customer. Since it is not always easy for customers to understand whether the developed solution aligns with

their needs, it is important to give a precise and detailed description of the project deliverables including the definition of measurable project success criteria. Again, here it helps to already be able to show customers results from similar projects.

- **Create project offer**
  This subphase concludes with a project offer. The project offer contains a summary of the problem that is to be solved, the KPIs which should be measured and achieved with the project outcome, the project deliverables and time estimates. It is important to emphasize requirements which the customer needs to meet, i.e., making data available in appropriate volumes and quality as well as including relevant stakeholders in the project.

Business Understanding

One of the most crucial parts of NLU projects is getting a good business understanding. This involves understanding the project background, business objectives and success criteria, the problem to be solved as well as the status quo. Achieving this requires the help of the business stakeholders as well as the people who deal with the problem, preferably domain experts and end users. We now list the activities of the Business Understanding subphase.

- **Thoroughly understand the business process, goals and KPIs**
  To ensure an alignment with the client's needs, it is crucial to understand the actual business problem that should be solved. The team should focus on understanding the business processes thoroughly as well as business goals and the KPIs which are used to measure them.

- **Identify key stakeholders**
  The team should identify the key stakeholders on the client side, such as decision makers, domain experts, IT and end users. They should also identify strategies as to how to manage those stakeholders.

- **Depict the business process and identify potential for optimization**
  NLU projects often begin with vague ideas of how NLU can support a company. [123] suggests explicitly depicting the business process of a customer. The depiction can be used to identify the parts of the process in which NLU solutions could be most beneficial.

- **Formulate clear business goals**
  Based on the assessment of the business processes, clear business goals should be formulated. The business goals should be assessed with respect to their potential return of investment (ROI), i.e., the ratio of cost and benefit.

- **Define KPIs for evaluation**
  To understand whether a project is successful, key performance indicators (KPIs) should be defined. It is a common mistake in data science projects to only evaluate solutions from a technical standpoint and omitting the evaluation of the actual business KPIs as they are not always correlated. For example, there could be the case that a machine learning model reaches high accuracies but does not solve the actual business problem.

- **Understand the project background**
  The data scientists which work on the project should get an introduction to the domain, relevant data and if applicable the current solution and previous attempts to solve the problem.
- **Create a project terminology**
  A project terminology should be created which contains the relevant business terms as well as the relevant terms on the NLU side. This will ease understanding and communication.
- **Collect and share all information via Knowledge Hub**
  All of the knowledge about the domain and the business problems that the team acquires should be documented in the Knowledge Hub and be made accessible to the customer for verification. Further, the business KPIs should be documented and explained, ideally together with a way of measuring them.

Requirements Engineering

For proper requirements engineering it is important to understand the purpose of the project, the environment in which a solution should be deployed and operated, as well as the properties and functionality a solution should have. Requirements can be divided into functional and non-functional requirements. The activities of the Requirements Engineering subphase are listed below.

- **Define functional requirements**
  Functional requirements define functionality of an application, e.g., the solution should extract amounts from invoices. Such requirements help translate the business problems to appropriate machine learning models.
- **Define Non-functional requirements**
  Non-functional requirements cover various properties of solutions such as the performance, inference speed, security, cost of operation and much more. Defining non-functional requirements is particularly important when using foundation models, as this has many implications for the project setup. Choices such as whether to use a cloud-based solution, whether to use a smaller or a larger model, whether to use encoder or decoder models and so on, depend on the non-functional requirements.
- **Involve and interview the right stakeholders**
  In order to get a full understanding of project requirements, various stakeholders should be involved. On the side of the client, managers, end-users and IT experts should provide input regarding requirements. On the development side, data scientists, ML engineers and user experience designers should be involved. End-users can provide valuable insights on functional and non-functional requirements of a solution. Domain experts can provide valuable input for data requirements, data sources and model properties. The client IT experts can provide insights into requirements with respect to the target environment, e.g., on-premise vs in the cloud, and the integration with other systems. Finally, business stakeholders can discuss requirements such as the cost of the maintenance and operation of a solution.
- **Prioritize and aggregate**
  Different stakeholders might have different priorities for requirements or even conflicting

requirements. It is the task of the requirements engineers to prioritize, aggregate and filter the various requirements. The final list of requirements should be discussed with and approved by the client.

- **Document requirements in the Knowledge Hub**
  Requirements should be documented and verified with the customer. Since requirements in data science projects are often more ambiguous than in traditional software development [153, 195, 205] it is recommendable to clarify and discuss them thoroughly.

Research

This subphase typically involves screening past projects, competing solutions and literature, but also screening the hubs for useful artifacts such as models or datasets which are appropriate for the project. The goal is to get a mostly complete picture of existing solutions and available internal and external assets. Below, we list the activities of the Research subphase.

- **Scout internal and external model and knowledge hubs**
  In the most trivial case, teams might already possess software components and models which solve the business problem. Teams should explore their model and knowledge hub for related projects and the associated artifacts. If nothing suitable is found, it might be possible that open-source solutions and models exist which can be reused. For example, the Huggingface model hub contains more than 250,000 pre-trained models available for reuse [84]. However, the usage license of these models needs to be checked, as not all of them can be used for commercial purposes.

- **Scout internal and external data hubs**
  At this point the team could also scout internal and external data hubs for datasets which are related to the project. The team might be able to retrieve datasets which contain domain-specific knowledge and language, such that new language models can be pre-trained or existing models can be adapted to the domain- or task-specific data, leading to potential performance improvements [108].

- **Search for relevant literature**
  In addition, the team should research for scientific papers, case studies and business offers by other companies which tackle similar problems. This way the team can gather ideas on how to solve problems. All of this helps the team to understand the complexity of the problem, estimate effort for the project realization more precisely and help with the solution concept design.

Concept design

An important task in early stages of a project is the translation of the business goals into NLU problems [185, 207] and systems specifications [145]. In Chapter 2, we introduced various NLU tasks indicating what type of problems NLU models can tackle. Next, we describe the activities of the Concept Design subphase.

- **Map business problems to modeling tasks**
  It is important to note that the solution design and modeling phases of most methodologies

assume that a single model always suffices to solve the business problems of customers. Our own project experience shows us that this is not the case and that many solutions require multiple ML components and potentially rule-based components which are integrated in an application.

- **Decide what to reuse and what to implement from scratch**
  Scout the Model and Software Hubs for reusable components and models and research for public alternatives.

- **Define modules and separate them from business logic**
  In many scenarios, modularizing ML and rule components is a success factor [200, 201]. Further, any kind of business logic should be separated from the ML models [200, 201] and used as a safeguard to mitigate model limitations [185, 200].

- **Design the target architecture**
  Solution architects should create a target architecture design which maps the business problems to different NLU tasks and shows the interplay of the components. Often machine learning models are only small components in larger systems [99]. Creating separate components or services per NLU task can ease the implementation and ensures the ability to exchange models more easily. However, a common problem which the team needs to account for is component entanglement [137, 210]. Here, changes in one component can lead to changes in the rest of a solution [145] due to high coupling. Another problem when chaining multiple components could be error propagation [205], where prediction errors in one component lead to follow-up errors in successive components which rely on the predictions.

- **Create an integration concept**
  If the result of a project should be integrated in existing systems or processes, it is beneficial that software engineers and user experience designers get an in-depth demonstration of them. This demonstration can serve as a starting point for a discussion on the integration of the NLU application as well as on the user experience design. Further, the solution concept should include descriptions of the interfaces. Integration in itself can pose a significant challenge [155, 183] and questions such as the compatibility and the orchestration of the systems should be discussed early on in the project.

- **Create an initial user experience design**
  User experience design in NLU projects is different from traditional software engineering. Most users are not accustomed to using software which produces uncertain or wrong outputs and many applications are not built with the uncertainty of machine learning models in mind. User experience design plays a crucial aspect in developing NLU solutions. Questions regarding accountability have to be considered, i.e., the question who is accountable in the event of an error caused by the application. It is important to explain the possibility of wrong or uncertain outputs and the need for users to correct the results of machine learning models. The interaction between users and the system is another important aspect. Does the user have to verify model outputs, if so all of them or only some, e.g., based on a model uncertainty threshold? How can the user correct wrong outputs? It might be useful to discuss the possibility of modifying a system such that it can support collecting user annotations for model training and monitoring. A user experience designer should interview the end users and domain experts and establish feedback loops with them.

- **Incorporate transparency and understandability requirements**
  Transparency and understandability of the inner workings of the system can lead to improved user acceptance. Both are not easy to achieve when using foundation models. As for all deep neural networks, they are black boxes by design. However, recent research aims to tackle this limitation by providing useful explanations for model behavior and model outputs [202]. It is important to note that such model explanations might not correlate with a human understanding of a correct explanation.

Project Planning

The Mutual Understanding phase ends with a concrete project plan that should be accepted by all parties involved. Planning the project accumulates all of the information acquired during acquisition, business understanding, requirements engineering, research and concept design. Planning should account for all required resources, i.e., the budget, time, personnel and hardware, as well as risks and contingency plans. Budget and time are hard constraints for the planning process and should be considered at every step. We now list the activities of the Project Planning subphase.

- **Plan hardware resources and costs**
  Given the substantial hardware requirements necessary for pre-training foundation models, it might be necessary to account for the reservation of existing or even the purchase of new computing resources. Reserving resources for pre-training can become a hurdle as these resources are potentially required over a course of multiple weeks and cannot be used for other projects. Foundation model training benefits mainly from the usage of graphics processing units (GPUs) or tensor processing units (TPUs). If not enough hardware is available, it might be necessary to purchase more hardware. The decision to invest in new hardware has to be agreed on with the respective management and to be accounted for in the project offer. Aside of the purchase cost, the energy cost of large language models is significant and need to be considered. If the hardware is not available or not affordable, teams might have to resort to cloud-solutions, although this can be a significant cost factor. The cost of training models like GPT-3 on cloud hardware is estimated to be above $5,000,000 [107]. Considerations regarding hardware have to be made for the deployment infrastructure as well as the inference infrastructure. Inference with foundation models is also resource-heavy and might require multiple servers with GPUs, depending on the desired response times of a system. If the customer hosts the solution, they have to be made aware of the cost of purchasing, maintaining and operating the inference hardware. If the software should be run on cloud infrastructure, a cost estimate should be calculated based on the predicted hardware needs. Another solution is to resort to the APIs of large language models by vendors such as OpenAI [230]. Here, the cost calculation needs to factor in which models will be used and how many tokens will be processed with the API. Other considerations, such as data privacy, of course play an important role in this decision.

- **Plan personnel**
  Regarding personnel planning, particular attention has to be paid to the availability of staff with the right skills. A project lead should define clear roles [88, 181, 191] and strive for balanced, multi-disciplinary teams [105, 141, 144, 192, 193]. A key success factor in many projects is the involvement of stakeholders on the customer side. Domain experts can be helpful in various phases of a project and their availability should be considered for the project plan.

This is particularly crucial in tasks such as annotation or evaluation, where domain expertise is indispensable. Ideally, stakeholders are being told early on when they are needed in the project. If possible, it makes sense to have certain stakeholders such as domain experts and end users available throughout the whole project.

- **Plan for time and effort**
  Scheduling and effort estimation is a notoriously difficult task in NLU and data science in general. This is mainly due to the many uncertainties in such projects. Project leads should account for large enough buffers in their planning. Further, a good way of mitigating risk and to react to uncertain outcomes is to resort to an agile approach and plan the project in iterations and with milestones. This allows project leads to redirect the project focus if certain decisions or outcomes lead to dead-ends. Especially in the beginning of NLU projects, during data exploration, there are not many demonstrable results, making it hard for customers to see the progress and the benefit of a project. Hence, it is crucial to make this transparent early on and deliver results in the form of progress reports and insights. To mitigate this delay, it helps when teams can build upon existing software and models, as this skews the time needed to reach a state in which the customer sees a benefit towards earlier phases of a project. Another good practice is the definition of milestones and deliverables and their timings. To decrease the risk a customer takes, milestones can be tied to breakpoints of a project, i.e., points at which a customer might decide to not continue the project.

- **Define Project Trajectory**
  The project lead should sketch which phases and subphases will be relevant in a project and in which order they will be traversed. This plan should be revised multiple times during the project.

- **Establish Quality Assurance Processes**
  Quality assurance is an important factor for all different parts of a project. This includes assuring the quality of results but also the processes and project management itself. The project lead should assure the quality of the project processes, while the team lead could be responsible for assuring quality of the project management. Quality review meetings should be established together with relevant stakeholders from the team and the customer side. Steering committee meetings are typically used to that end.

- **Schedule meetings**
  Finally, regular meetings inside of teams but also across teams should be established. Depending on the project management style, the frequency and types of meetings can vary. A possible mode could involve Scrum-like sprints, where there are team stand ups and 2- or 4-weekly sprint planning and review meetings. The sprints should not only be tied to feature development but could also account for intermediate results such as a ”data exploration report” [6].

Rationale

The Mutual Understanding phase sets the groundwork for the whole project. Both customers and the NLU team need to get a good understanding of the use case, possible solutions and the project process. The acquisition subphase is omitted by most other methodologies. We include it in our methodology

as it can play an important role for expectation management and planning. Important business and domain information are shared during the acquisition subphase and the use cases are defined. Further, a lot of planning and estimating has to be done in this phase. Time, budget and resource estimations in data science projects are often very difficult. Hence, it is important to get as much information as possible prior to the project start.

Good artifact management can improve estimation and reduce overall effort. Artifacts from the Knowledge Hub can be used to relate a new project to former projects and prior experiences. Reusable models, datasets and software can lower the resource demand for executing projects and can further serve as demonstrable assets to increase customer trust. Using foundation models has a large impact on the overall cost of projects. If the pre-trained models are already usable for the customer task, they reduce modeling and data processing efforts to a minimum. If models need to be pre-trained from scratch, the costs and resource demand significantly increase. Foundation models allow to reuse the same artifacts across projects and hence increase synergies. Building and expanding the Model and Data Hub of a team requires strategic thinking across projects and should be an influential factor for the marketing and acquisition efforts of business development.

The Mutual Understanding phase also serves the goal to set up the project organization and infrastructure. This includes planning, staffing, establishing communication channels, ethics and legal reviews as well as early solution concepts.

This phase covers all of the characteristics from the Management & Governance category of our characteristics catalog, which intuitively makes sense, as these things need to be set up and defined at the start of a project or even before it starts. From the Process category, it covers all of the characteristics belonging to the Project Ideation and Project Management subcategories, as well as some characteristics from the Application and User Experience Design subcategories. From the Technology and Infrastructure category, only Visualization tools are mentioned, as they can be used to ease the understanding of the customer with regards to NLU solution capabilities. Finally, from the Artifact and Asset Management category, mostly characteristics regarding the reuse of models, data and software are covered. In total, this phase covers 64 different characteristics.

5.3.2 Data Gathering and Understanding Phase

The second phase of the methodology tackles the aspects of understanding the available data and gathering and assembling a dataset which helps to achieve the modeling goals. Depending on the project goal, the data might already be on the customer side and only needs to be made accessible to the NLU team. However, in the case of working with foundation models it can often be beneficial to also search for additional datasets which contain domain knowledge. This can be publicly available data or additional data on the customer side, which is not necessarily directly related to the machine learning problem but still contains valuable domain knowledge. In the case that a team wants to pre-train their own foundation model, data crawling is an essential part. While teams can resort to publicly curated datasets, it is common practice to also crawl data. The data that is obtained should be explored and analyzed. Finally, a data assessment should be completed and hypothesis about the data and potential models should be generated. Next, we provide a detailed description of the subphases of the Data Gathering and Understanding phase. A compact summary of the phase is depicted in Figure 5.3. Detailed descriptions of the roles, artifacts and tools are in Subsection A.5.2 in the appendix.

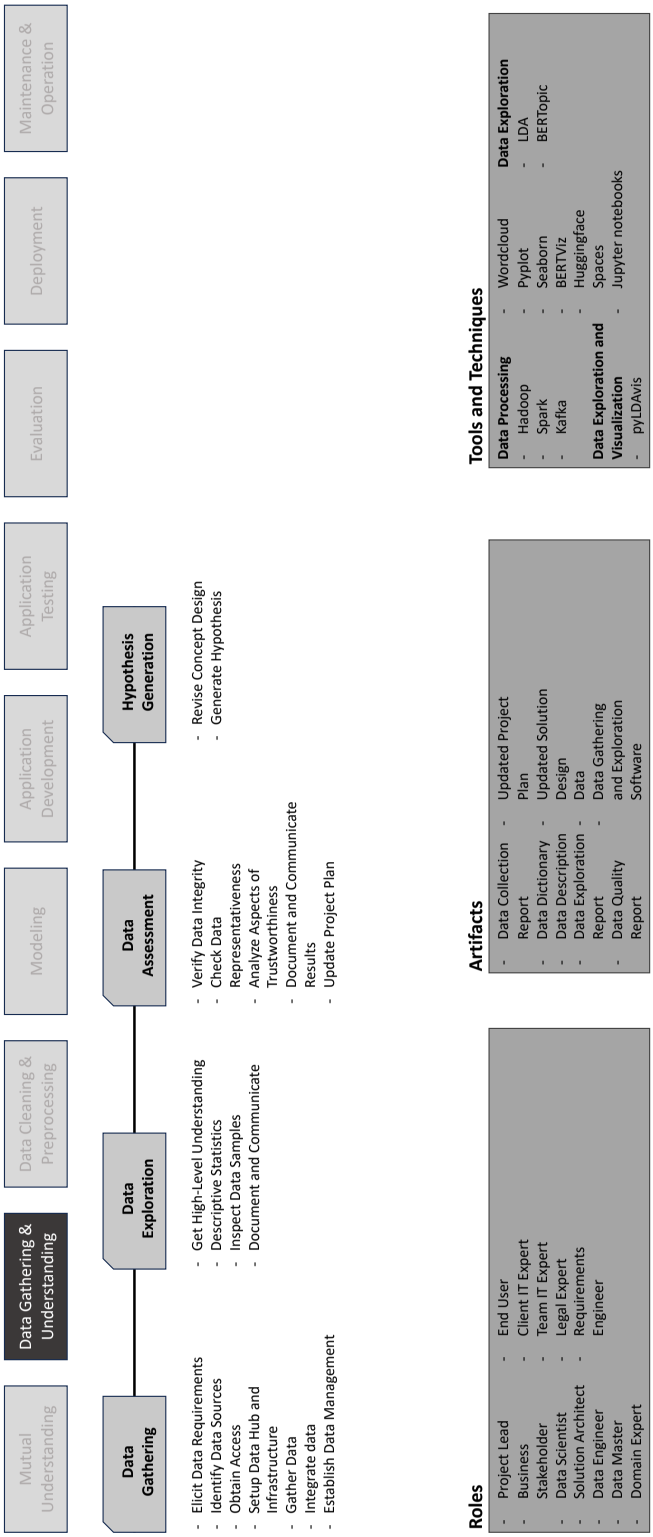

Figure 5.3: Overview of the Data Gathering and Understanding phase.

Data Gathering

Gathering and collecting data is vital for the success of NLU projects. LLMs are pre-trained on vast amounts of text and model performance is directly correlated to the amount of training data used. Hoffmann et al. define a scaling law, which roughly suggests to double the amount of training data for every doubling of model size [71]. Further, domain-specific data improves the language understanding capabilities of LLMs and can lead to better performance on downstream tasks [108]. We now list the activities of the Data Gathering subphase.

1. **Elicit Data Requirements**
   In this phase of the project the team should expand on the requirements elicited in the Mutual Understanding phase and focus on data requirements. Data requirements are mainly dependent on the business problem to be solved and the analytics approach chosen by the team. They cover aspects such as the required volume of data, the content of the data, its format and type as well as information such as if labels are present. Since foundation models are trained in a self-supervised fashion, projects in which foundation models are pre-trained or fine-tuned to a domain could benefit from high volumes of unlabelled data. All requirements should be elicited together by data engineers as well as domain experts.

2. **Identify Data Sources**
   Based on the data requirements, appropriate data sources are identified. This happens in collaboration with domain experts and the IT experts of the client. The sources can be internal, containing historically captured data by the client, as well external, e.g., from public data hubs such as Huggingface [84] or repositories such as the LLMDataHub [231]. Clients might not always have a clear overview of their data sources and the data they contain. Hence, it makes sense to create a high-level overview of the available data sources and assets on the client side. Further, the teams Data Hub should be scouted for relevant datasets from previous projects which could be useful. Relevant data does not necessarily have to be in the form of text, it might make sense to look for structured resources such as ontologies to improve modeling and ease annotation, e.g., via distant supervision, or to look for multimodal data such as document images, which can easily be processed and understood by recent multimodal foundation models. [115] suggest to hold out from expensive dataset purchases in this phases until further information about data quality and model quality are obtained. However, if purchased data is used [123] recommend documenting its use and value to justify future purchases. [48] recommend selecting data sources in a way that mitigates the risk of bias.

3. **Obtain Access**
   Depending on the sensitivity of the data it might be necessary to define usage and access policies. If possible, authorization and access to the previously identified data sources should be obtained. If it is not possible, e.g. due to privacy issues, it might be possible to get access to an anonymized subset of the datasets.

4. **Setup Data Hub and Infrastructure**
   Before starting to gather the data, the infrastructure for storing the data has to be set up appropriately. Important characteristics which need to be considered are the volume of the data as well as security, privacy and safety concerns. IT should set up the Data Hub and the

infrastructure according to these characteristics. If legally possible, the data should always be stored in a manner that makes it accessible for the whole team, but at the very least to all stakeholders in the project who need access. The project lead has to ensure proper handling of the data according to project, team and regulatory specifications. This includes data protection as well as redundant storage to prevent data loss.

5. **Gather Data**
   In this step the data is collected and transferred to the training environment and is also stored in the Data Hub. If the data contains privacy relevant information, it could be necessary to anonymize it before transfer. Data connectors and crawlers are written or reused and set up to accommodate for the chosen data sources. If streaming-based sources are used, the period for which they are crawled should be determined. The database design for storing the data on the training environment is based on Big Data V characteristics such as volume, velocity and variety. All connectors and crawlers should be committed to the Software Hub.

6. **Integrate Data**
   It might be necessary to integrate data from multiple sources and databases. [232] argue that this can lead to privacy issues when critical connections between data can be made.

7. **Establish Data Management**
   The obtained datasets are versioned in the Data Hub and enriched with available metadata, time and source of retrieval, information about data ownership and licensing as well all other domain- and data-specific information. Further, the history and lineage of data is tracked in the Data Hub and the datasets are named according to the teams naming conventions. All information about the collection and modifications of the datasets are stored in the Knowledge Hub to assure reproducibility, to enable tracking downstream errors and for quality control.

Data Exploration

Data understanding helps data scientists in the process of data cleaning and modeling. This way important properties of the data can be discovered early in the project. Data exploration might already yield important insights for businesses and can in itself already be a valuable project deliverable. Next, we describe the activities of the Data Exploration subphase.

1. **Get High-Level Understanding**
   In the beginning it is important to get a high-level understanding of the available data, its structure, formats and content. Domain experts and business users can provide valuable insights into the data and point the team towards interesting subparts of the data, expected problems in the data quality or their hypothesis on the importance of different data samples.

2. **Descriptive Statistics**
   To understand the data better teams should perform descriptive analysis of the data and visualize their results. This analysis focuses on the complete dataset rather than single samples. Examples for interesting statistics are the volume of data, the frequency of labels, the distribution of languages across documents, its volatility, variability and so on. Interesting or surprising insights from the analysis should be discussed with domain experts and business stakeholders.

3. **Inspect Data Samples**
   After gaining insights about the dataset, the team should inspect a subsample of the data individually. This helps to understand the data and how to solve the problem better. The inspected subsample should give a concise picture and should be representative for the dataset. It also makes sense to look at conspicuous data samples, e.g., data samples which significantly differ in size or in the amount of annotations. Visualization methods can help foster a discussion about the samples with the domain experts.

4. **Document and Communicate**
   All insights about the data should be documented in the Knowledge Hub and communicated to the client. This also holds for all the questions and answers in the discussions about the dataset with the client. The knowledge about the data should be enriched with domain knowledge by the client. Finally, the team should document how the insights were generated to assure correctness and reproducibility.

Data Assessment

Data quality is central to the success of data science projects. In this subphase the quality is assessed with regards to integrity, representativeness for the tasks and trustworthiness aspects. Assessing and analyzing large volumes of data is not trivial and might require specialized tools for big data analysis. Biases in the data are particularly important, when using foundation models, as they can lead to discrimination, which can lead to further damages. Following, we list the activities of the Data Assessment subphase.

1. **Verify Data Integrity**
   The team should verify the data integrity by checking the data for completeness, missing values, correctness and consistency. Particularly in projects with large data volumes it is not possible to check all data points, instead the results of the descriptive analysis and the sample screening are used. For labelled data it is also useful to check the correctness of the labels and whether the dataset contains noise. Problems can be caused, e.g., by wrong labeling or ambiguity of labels.

2. **Check Data Representativeness**
   In this step the team should check whether the data is representative for the business problems and suitable for the analytical approaches. Important aspects to verify are the timeliness of the data, the distribution of the data and labels and whether the data contains the right labels to solve a given task with analytical approaches.

3. **Analyze Aspects of Trustworthiness**
   Trustworthiness aspects of data have become even more important given the use of foundation models and the vast amounts of data they are trained on. Together with domain experts, different aspects of the data set should be analyzed. Sample questions at this stage are: Does the data contain biases? Is it discriminatory?

4. **Document and Communicate Assessment Results**
   Based on the data assessment decisions are made. It might be necessary to gather more data, to obtain additional annotations or to discard certain parts of the dataset, e.g., due quality or bias

issues. All of the insights are documented in the Knowledge Hub and shared with team and the client.

5. **Update Project Plan**
   It might be necessary to adjust the project plan at this stage. Scouting and obtaining further data or labeling more data are resource- and time-expensive tasks. This could imply an adjustment of the schedule, and a revised estimate of necessary resources such as hardware, time and budget. The renewed project plan has to be communicated to the client and accepted by them.

Hypothesis Generation

Based on the insights about the data, the original solution architecture can be revised. It might for example be necessary to use other modeling tasks or model classes. If further exploratory analysis of the data is wished, hypothesis about the data can be generated in this subphase. We will now list the activities of the Hypothesis Generation subphase.

1. **Revise Concept Design**
   The assessment of the available data might lead to a revision of the concept design. Since the model selection depends heavily on the available data, it could be that initial modeling concepts have to be changed.

2. **Generate Hypothesis**
   In projects with exploratory settings, hypothesis about insights in the data can be formulated together with relevant stakeholders from the client. These hypothesis should be documented in the Knowledge Hub and shared and discussed with the rest of the team and the client, as suggested by [111].

Rationale

Data is the central aspect of data science projects and NLU projects alike. A key component of our methodology is the introduction of a Data Hub and the expansion of the data lifecycle across projects. Gathering, maintaining and managing datasets has become increasingly important with the emergence of foundation models. Pre-training foundation models requires large amounts of data, with the largest datasets containing over one trillion tokens [233]. Handling such volumes of data poses challenges already known from big data projects, however further challenges arise. Datasets need to be investigated with respect to bias and fairness, legal aspects need to be considered, safety and security aspects as well as the ownership of data. Teams within an organization should establish strategies for data sharing to benefit from the work of others.

Another problem that is typical for NLU projects is the difficulty of exploring large amounts of unstructured data. It can be beneficial to use NLU models to cluster document collections into topics, to provide an easier to digest overview. Further, NLU projects often combine structured and unstructured data, requiring to handle and process both types of data.

This phase covers many characteristics of the Artifact and Asset Management category and especially almost all characteristics which belong to the Data subcategory. Intuitively this makes sense, as the phase revolves around collecting data and understanding its properties. Further, five out of seven characteristics in the Technology and Infrastructure category are covered, namely those

for infrastructure and data analysis, processing and visualization. From the Process category only seven characteristics are handled. This includes the Requirements Engineering, Data Understanding, Collection and Dataset Assembly characteristics. Further, the choice of pre-training vs adaptation of models influences the dataset collection. The only three characteristics in the Management and Governance category which are important in this phase are the stakeholder involvement, including business stakeholders, i.e. the customers, and domain experts.

5.3.3 Data Cleaning and Preprocessing Phase

Recently, the data science research community has called for a data-centric data science approach [234]. The reasoning behind this is, that the research community solely focuses on improving algorithms. Cleaning and optimizing a dataset is considered a vital aspect of success for data science projects and NLU projects in particular. While classical NLU approaches often required vast preprocessing, such as stemming, lemmatization, phrase-detection and more, the subword tokenizers of pre-trained language models reduce the preprocessing steps to a minimum. Nevertheless, it is important to clean a dataset, in particular during pre-training. Methods to score the quality of data samples for filtering and to deduplicate texts significantly improve the models [235]. An aspect that is often neglected by existing methodologies is data annotation. Data annotation takes a significant amount of time and benefits from clear processes and guidelines. In the previous chapter we have seen that data augmentation, in particular data annotation, is among the most important characteristics of NLU projects. Overall, this phase is considered one of the most crucial and time-consuming activities in NLU projects [116]. Following, we describe the subphases of the Data Cleaning and Preprocessing phase. A compact summary of the phase is depicted in Figure 5.4. Detailed descriptions of the roles, artifacts and tools are in Subsection A.5.3 in the appendix.

Data Transformation

In order to make data from different sources, formats and types usable, it has to be transformed in a way that makes it consumable for the software components and models that should be used. These transformations should be standardized via pipelines and made accessible to everyone in the team. The Data Transformation subphase contains the following activities.

1. **Determine Standard Formats**
   Data is available in many formats and types. NLU projects do not only rely on textual data but can be enriched with structured data, images and other data types. It is beneficial for teams to build upon standard formats for their data and transform data into these formats. This makes it easier for others to work with the same data and simplifies communication between the different software components. Some frameworks such as spaCy, Apache UIMA, PyTorch or TensorFlow offer standard formats for datasets. Another important factor, when dealing with text, is the encoding used for the texts. The team should agree on a standard encoding as well.

2. **Use Data Transformation Pipeline**
   Pipelines for data transformation should be reused from the Software Hub or implemented and committed to the Software Hub and shared with the rest of the team. Different data formats require different parsers and contain various meta-information, such as layout information,

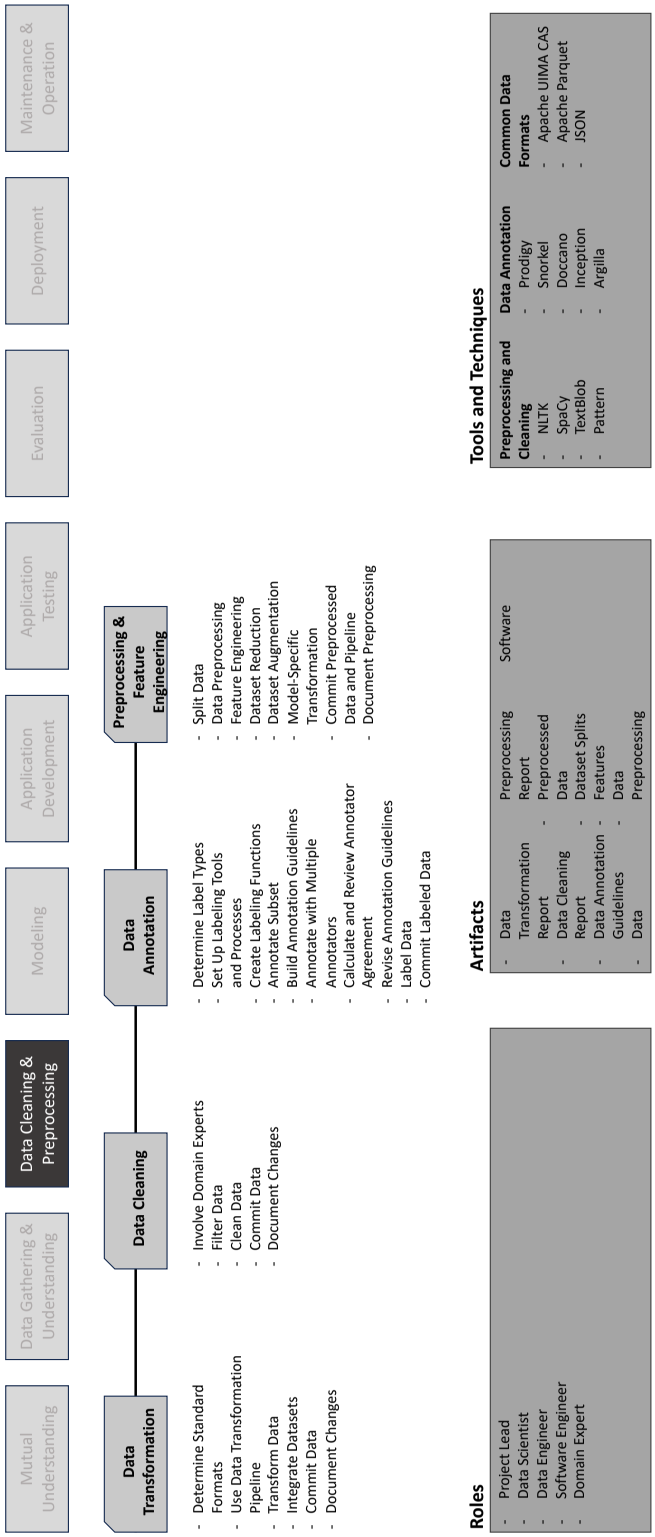

Figure 5.4: Overview of the Data Cleaning and Preprocessing phase.

which should ideally be preserved. Standard parsers for many data formats exist as open-source solutions.

3. **Transform Data**
   Once standard formats are agreed upon, the data should be transformed into the respective formats. Information regarding the format and metadata should be preserved in the transformation. If the data is multimodal, the new format should preserve the content of the modalities other than text, e.g., images or positional information.

4. **Integrate Datasets**
   If multiple datasets, perhaps from multiple sources, have been gathered, they need to be joined in this step. This step is a lot easier if the datasets have been transformed into a joint format.

5. **Commit Data**
   The transformed data should be stored in the Data Hub and be linked to the raw, original data.

6. **Document Changes**
   All changes to the data should be documented in the Knowledge Hub alongside the reasoning behind the changes.

Data Cleaning

Cleaning data assures the quality of the data and can play an important role for modeling. Many pre-trained models can only digest limited context lengths in the input. Hence, removing unnecessary parts of the input helps to center the attention of the models on the relevant parts. Further, cleaning prevents the models from learning from noisy, wrong or biased texts. Next, we describe the activities of the Data Cleaning subphase.

1. **Involve Domain Experts**
   Based on the insights of the Business and Data Understanding phase, the team discusses how to clean the data with domain experts. This can lead to inclusion and exclusion criteria for samples, e.g., based on quality criteria, relevance, duplication, outlier or bias. Further, the domain experts should help identifying relevant and irrelevant parts of the data.

2. **Filter Data**
   The data is filtered based on the exclusion and inclusion criteria and the filter pipeline is committed to the Software Hub. The filtering step can include data quality scores which have to be met. Such scores can for example be determined by using existing language models.

3. **Clean Data**
   Cleaning the data includes tasks such as removing noise, imputing missing values or NANs and handling special values. Especially noise removal can play an important role in NLU projects. Many pre-trained language models have limitations regarding the context sizes which they can process. If it is possible to reduce documents to relevant parts, this can directly lead to better modeling quality. The data cleaning pipeline should be committed to the Software Hub.

4. **Commit Data**
   Once cleaned, the data should be committed to the Data Hub and linked to the original data.

5. **Document Changes**
   All changes to the dataset regarding filtering and cleaning should be documented in the Knowledge Hub and be shared and discussed with the team and the domain experts. It is important to not only document how the data was cleaned but also why the particular cleaning steps were chosen

Data Annotation

Data annotation is overlooked by many methodologies, but it is an essential part of NLU projects. Especially when it comes to supervised learning and assessing the performance of models and applications, annotated data is indispensable. Annotating data is a difficult and tedious task. First of all, it requires domain expertise, which is often not available and costly. Further, the domain experts need to be educated on using the annotation tools and on making proper and coherent annotations. The Data2Value methodology is the only methodology in our analysis which describes an annotation process. We follow this process but modify it, since it ignores vital aspects of modern projects, such as using tools or models for automated annotations. In particular, foundation models can be used to annotate data (semi-)automatically. Following, we list the activities of the Data Annotation subphase.

1. **Determine Label Types**
   Depending on the problem, different types of labels are needed. In classification problems, e.g., a document and an according label are required, while for entity extraction tasks annotated spans are most useful. However, the type of label is also dependent on the model type, as mentioned named entity annotations can be represented as text spans, however, the problem can also be posed as a question-answering problem for which it would suffice to have the document and the respective value, without annotating the exact span.

2. **Set Up Labeling Tools and Processes**
   Data Annotation Tools should be used to ease the labeling process. There are various tools for text annotations, spanning from tools specialized in automated annotations to tools that help manually annotating data. The annotators, often domain experts, should be instructed on how to use the labeling tools. Potentially different roles are given to annotators, e.g. some annotators get the role of reviewers, who review the annotations of others. Recently, LLMs have shown great potential to create meaningful weak annotations [236], which can be used to train fine-tuned models.

3. **Create Labeling Functions**
   If possible, domain knowledge should be used to create labeling functions. Labeling functions can vary from list-lookups to ontology matching or rule based annotations.

4. **Annotate Subset**
   Initially only a subset of data should be annotated by the annotators. This subset will be used to guide the annotation process and defining guidelines.

5. **Build Annotation Guidelines**
   Based on the initially annotated subset of data, annotation guidelines are created, shared with the team and the client and documented in the Knowledge Hub. These annotation guidelines should contain the annotation scheme, rules for annotation and sample annotations per annotation type.

6. **Annotate with Multiple Annotators**
   Once the annotation guidelines are established, a team of annotators should annotate the data. Initially a small subset of the data should be annotated by all annotators. If no special knowledge or domain expertise is required for annotating, it is common practice in NLU to use crowd-workers for annotation tasks [237]. In this case it also makes sense to let them annotate a subset of the data first, to verify that they produce labels of high quality.

7. **Calculate and Review Annotator Agreement**
   Once the first subset of data was annotated by all annotators, the inter-annotator agreement between them should be calculated. Based on the results new insights about the annotation process can be derived, such as whether the annotation scheme is complete and correct and whether the annotation guideline is sufficiently specified.

8. **Revise Annotation Guidelines**
   Based on the results of the annotation review, the guidelines are overhauled and again posted into the Knowledge Hub. If necessary, the new annotation guideline is again verified by a new iteration of annotations and reviewing.

9. **Label Data**
   Once the annotation guidelines are finalized the labeling process starts. It can make sense to train models before the whole training set is annotated. If the quality of these models suffices, e.g., if they have a high precision, they can be used to help in the annotation process. Another option is to use active learning methods, to reduce the annotation effort.

10. **Commit Labeled Data**
    Once the labeling of the dataset is finalized, it is committed to the Data Hub and linked to the original dataset.

Preprocessing and Feature Engineering

The Preprocessing and Feature Engineering subphase aims at preparing a dataset for modeling. Ultimately, the datasets are transformed such that they can be consumed by the models and yield the best possible results. For many tasks, transformer-based language models reduce the complexity of preprocessing to tokenization and sentence splitting. While classical models required source- and language-specific preprocessing methods, pre-trained language models are often trained on various sources and even languages, which further significantly eases the preprocessing steps. We will now list the activities of the Preprocessing and Feature Engineering subphase.

1. **Split Data** If the dataset contains labeled data, the dataset should be partitioned to a training, validation and test split. This split should be broadcasted to the whole team to assure reproducibility and comparability of results.

2. **Data Preprocessing**
   Common preprocessing methods for NLU include removing punctuation, tokenization, sentence splitting, stopword removal and many more. For traditional learning methods, these methods were crucial for successful modeling, however their importance has declined with the use of pre-trained language modeling. When using pre-trained language models, they frequently come

with a subword tokenizer. Preprocessing is than often reduced to sentence splitting and applying the tokenizer. Some models require lower-cased text, however for certain languages the casing of words is important as it shifts their meaning.

3. **Feature Engineering**
   Classical NLU feature reduction and engineering methods include stemming, lemmatization and phrase/ngram detection. Other feature engineering methods include applying existing models, e.g., for clustering, topic modeling or sentiment detection and enriching the data with the meta-information. These methods are especially useful when using BoW-like feature representations. Distributed embedding methods such as Word2Vec also benefit from them. With pre-trained language models, feature engineering is mostly reduced to obtaining vector representations by inputting the text to the models.

4. **Dataset Reduction**
   There can be multiple reasons for reducing the dataset. One example could be a skewed distribution of labels. In this case, it might make sense to under-sample data from frequent classes. Another example could be that only certain classes, clusters or topics should be examined in the modeling phase, for example to focus an analysis on documents which contain a certain topic. Decreasing modeling complexity could also be a good reason to reduce the dataset, e.g., by focusing only on particularly relevant labels. This could also imply that certain document types become irrelevant.

5. **Dataset Augmentation**
   Augmenting a dataset can be helpful to tackle the imbalance of labels or to increase the robustness of a model. Augmenting textual data is not trivial as changes to the texts might alter their meaning or introduce grammatical errors. There are various methods such as synonym replacement, sentence shuffling, etc. which can be used for augmentation and libraries such as NLPAug [238].

6. **Model-Specific Transformation**
   At this point the data should be transformed in a way that makes it suitable for consumption by the model types that will be used in the next phase. Different model types require different input formats, often dependent on the task they should learn.

7. **Commit Preprocessed Data and Pipeline**
   The preprocessed data should be committed to the Data Hub, the data preprocessing pipeline should be committed to the Software Hub and all information should be linked. The code for the data pipeline should be reviewed by a software engineer or data engineers and committed.

8. **Document Preprocessing**
   The preprocessing should be documented in the Knowledge Hub together with a rationale of the preprocessing choices.

Rationale

Data preprocessing is a phase, which almost all methodologies share. It is known as the most time-intense phase of data science projects [102]. In addition, data annotation or labeling, is required

in many projects and often requires the time of domain experts. Finding a suitable annotation scheme is a difficulty and often requires multiple iterations. Many of the case studies we presented in the last chapter, had long data annotation phases with multiple revisions of the annotation guidelines. Apart of our methodology, only the D2V methodology covers an annotation phase which we took as a starting point for our subphase [17]. Data annotation in texts can also be performed with the help of structured resources, rules or even existing models. Foundation models are a great source for weak annotations due to their significant zero-shot capabilities. We therefor incorporated explicit steps to create automated annotation functions.

We emphasize the importance of managing and versioning data after the different preprocessing steps. Further, we put an emphasis on data cleaning methods which are particularly useful when pre-training foundation models, such as data deduplication, filtering for bias and filtering by using quality gates. Pre-training of foundation models requires large volumes of unstructured text, which need the respective compute power and storage to handle the preprocessing. Further, we suggest building standard data processing pipelines, to automate preprocessing over time.

Preprocessing of textual data has changed with the advent of pre-trained language models. Most of those models have dedicated tokenizers and only require minimal preprocessing once the data is in a format which is digestible for the models. Feature engineering is often reduced to applying the language models and using their embeddings.

This phase covers 24 characteristics. Most of them concern the Data Preprocessing subcategory as well as the Data subcategory in the Artifact Asset Management category. Regarding the Data category, particularly the characteristics are tackled are which concerned with data types and formats. This phase further involves the Infrastructure and Data Tools characteristics from Technology and Infrastructure. Finally, the involvement of domain experts is mentioned from the Management and Governance category.

5.3.4 Modeling Phase

In this phase appropriate modeling techniques for the business problems are selected, implemented, evaluated and then the resulting models are submitted to the Model Hub. In contrast to other methodologies, we emphasize that especially larger projects often require combinations of multiple models and heuristics/rules. Modeling is a highly iterative and experimental process. The results of modeling depend on the correct understanding of the business problem, correct data understanding, appropriate preprocessing and the choice of model. Projects often loop back to data preparation and domain understanding activities after initial modeling efforts.

With the availability of foundation models, the modeling phase in data science projects has significantly changed. Modeling iterations can be sped up by utilizing already existing models and their transfer learning capabilities. If a team decides to use foundation models in a project, they should start by researching existing models. A large emphasis in the modeling phase should be put on researching internal and external model hubs for models which are suited for the problem types in the project. Public model hubs such as Huggingface, contain over 250,000 pre-trained models, many of which are already specialized for certain tasks such as sentiment detection or named entity recognition [84]. Further, so called Adapter Hubs exist, which provide adapters for existing language models, to adapt them to certain domains or tasks [239]. Additionally, public models are available via APIs and can either be queried for text embeddings [240] or for performing tasks such as sentiment detection and named entity recognition [241]. More recently, APIs for large generative language models have

become more popular [230]. As shown in Chapter 2 these models show the ability of being adapted to a task via natural language prompts, often excelling in tasks even in zero-shot settings, i.e., without requiring training data. This might reduce the need for annotated data and reduce modeling activities to prompt design.

If a team decides to train their own model, the choice of training method is important. The team should decide whether it is sufficient to adapt an existing model to a task, whether a model needs to be adapted to a certain domain or whether pre-training from scratch is most beneficial. Generally, we recommend following the rule to start with the least complexity in modeling and then increase the complexity. Typically projects require multiple modeling iterations until they reach the desired quality or run out of budget. Shorter iteration cycles are helpful, because domain experts and users can provide their feedback quicker, which can in turn be incorporated in new modeling iterations. We now describe the subsections of the Modeling phase. A compact summary of the phase is depicted in Figure 5.5. Detailed descriptions of the roles, artifacts and tools are in Subsection A.5.4 in the appendix.

Model Selection

The choice of an appropriate model depends on the business goals, the data and the problem types the respective goals were mapped to. There are often multiple ways of mapping a business problem to problem types. Information extraction from documents could for example be posed as a reading comprehension problem or a named entity recognition problem. Further, there are multiple choices of models or rules/heuristics for these problem types. Hence, the selection of models depends on the specific requirements of a project. Following, we list the activities of the Model Selection subphase.

Studer et al. list various characteristics which influence the selection of an appropriate model: Performance, Robustness, Scalability, Explainability, Model Complexity and Resource Demand [48]. According to them, model selection requires finding the right trade-off between these characteristics, this is depicted in Figure 5.6. They emphasize that model fairness plays an important role as well. We further want to add the dimension of Reusability or Adaptability, which has become more important with foundation models. Building and using reusable models can have a large strategic value for a team, as they can build upon them in future projects. However, they usually require tradeoffs in terms of explainability and resource demand. Lastly, the availability of annotated training data and the type of annotations can impact the selection of the right model types, i.e., data must be available in the form required by the model and the model needs to produce the right types of outputs.

1. **Map Problem Types to Modeling Techniques**
   In this activity, each problem type is mapped to modeling techniques. These techniques can be rule-based or machine-learning based or in some cases combinations of both. Whenever possible it might be beneficial to build end-to-end models instead of chaining models and rules, to avoid error propagation and dependencies. At this point it makes sense to research for past solutions in the Knowledge Hub or in external sources, e.g., by studying literature. The page "paperswithcode" offers a good overview of state-of-the-art models based on tasks and datasets [206]. If multiple modeling techniques need to be combined, the dependencies between them should be clearly depicted, as they have to be accounted for during application development, evaluation and application testing.

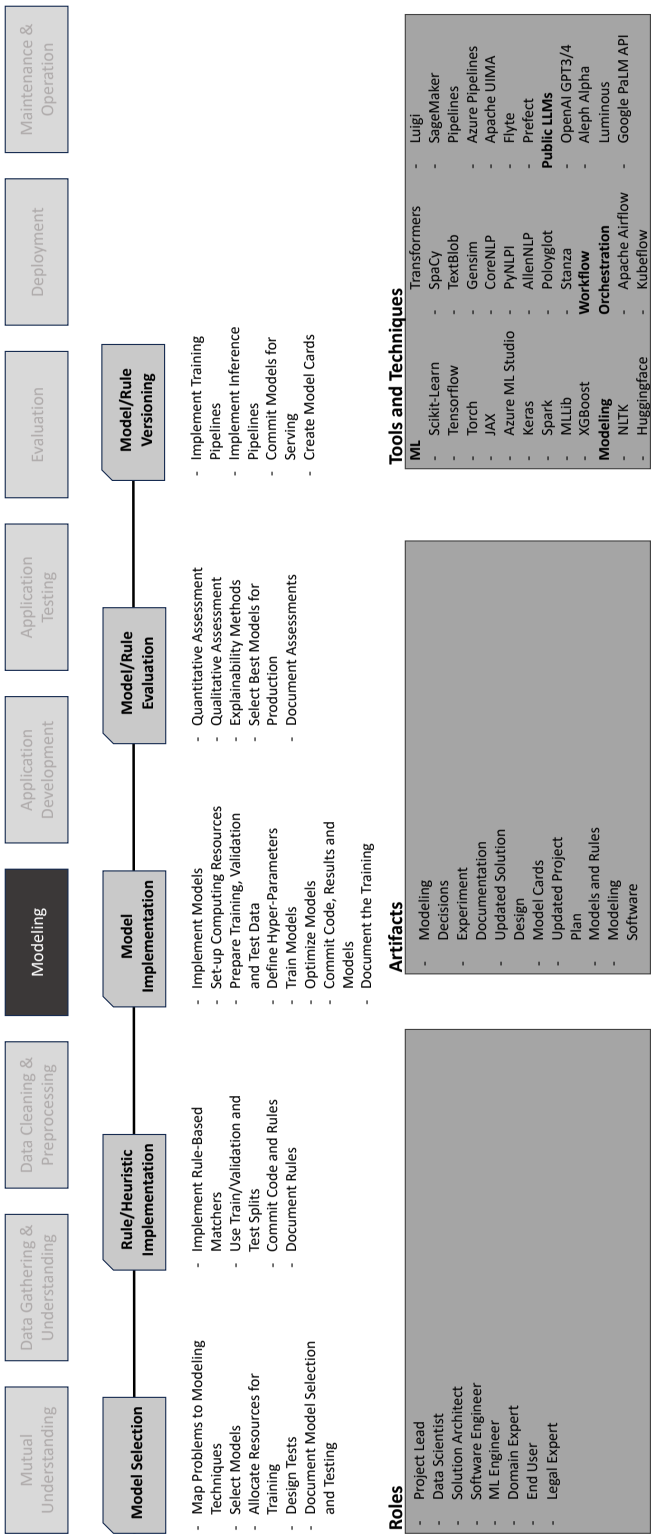

Figure 5.5: Overview of the Modeling phase.

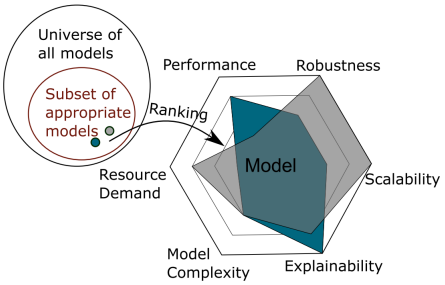

Figure 5.6: The trade-off between various model characteristics when selecting a model, as depicted in [48].

2. **Select Models**
   Based on the aforementioned characteristics, appropriate models are selected. It is a good practice to start with a low complexity baseline in a first modeling iteration and then increase the complexity, if the model quality does not yet suffice [31, 48, 123]. Generally the order of selecting models should be the following: If task-specific models are available, try these models first. If not, try adapting general models via prompting or fine-tuning with annotated data. If the quality does not suffice, try enriching the general language model with more domain- or task-specific data, i.e., fine-tuning it to the domain. Then, try fine-tuning on the task again. If this still does not suffice it is worth considering pre-training a model from scratch. However, this requires resources such as hardware, budget and time. External and internal Model Hubs should be used to scout for the models. It is important to pay attention to the licensing terms and validate that the models can be used for the project. To this end, legal experts can advise the team.

3. **Allocate Resources for Training**
   Based on the selected models and the amount of data the model training is scheduled and the appropriate resources in terms of hardware and personnel are allocated. The training of large language models takes a significant amount of time, especially when pre-training and the allocation of computing resources for this task should be communicated to others to avoid scheduling conflicts with other projects.

4. **Design Tests**
   Based on the selected models and the requirements, appropriate tests are designed. These tests have to account for multiple factors such as error propagation if multiple models are combined, they should assess the model quality with appropriate metrics and should consider requirements such as inference speed, robustness and fairness. The choice of an appropriate metric depends on the task models are benchmarked on. Aside of typical machine learning metrics, there are also NLP-specific metrics such as BERTScore [178]. Behavioral testing of NLU models with CheckList provides a good way of testing models for their ability to generalize and their general robustness [218]. Lastly, tests for the business KPI evaluation should be defined which will be performed in the Evaluation phase. This step is crucial, as these KPIs determine the success of a project.

5. **Document Model Selection and Testing**
   The overall modeling concept, the model selection choices and the test design are documented

in the Knowledge Hub, shared with the team and communicated to the customer.

Rule/Heuristic Implementation

Many problems in NLU can be tackled with rule-based methods such as regular expressions, list or ontology look-ups and so on. For simpler problems it might completely suffice to base a solution on rules, e.g., for simple pattern matching, such as detection of date-patterns. Other times, rule-based matches can be used as input for machine learning models [242] or as post-processing of model results [243]. Regular expressions, ontologies, etc. should also be viewed as artifacts which can be used and maintained across projects. Next, we list the activities of the Rule/Heuristic Implementation subphase.

1. **Implement Rule-Based Matchers**
   In this step the knowledge of domain experts can help the team with modeling. They can point the team to valuable resources, help them build ontologies or terminologies and formulate domain-specific rules for matching. Compared to modeling, these steps are often more time-consuming and only useful if it is clear that there is a benefit of using rules over machine learning models. Software and Model Hubs should be scouted for existing implementations.
2. **Use Train/Validation and Test Splits**
   Just like in machine learning, rules can be overengineered. To understand the quality of the rule-based methods, they should also be developed against the same training and validation split as the models and later evaluated on the test split.
3. **Commit Code and Rules**
   The final rules should be committed to the model hub and the respective code for implementation to the Software Hub.
4. **Document Rules**
   The rules and the rationale behind them should be documented in the Knowledge Hub and shared with the team and the client.

Model Implementation

Implementing and training models is a core activity of NLU projects. However, the process has become significantly easier since many models are available pre-trained and fine-tuned for specific tasks. This way modeling can ideally be reduced to reusing existing models. The availability of models enables project teams to quickly produce baselines and first results. However, training can be more difficult when larger models are pre-trained from scratch. In these cases training is often an effort which requires the interplay of multiple people and also distribution across hardware. We will now list the activities of the Model Implementation subphase.

1. **Implement Models**
   In this step, the training pipelines for the designated machine learning models are implemented. Ideally, existing code from the external or internal Software Hub can be reused for training. In the case of resorting to already pre-trained models, appropriate scripts are most likely available from the Huggingface Transformers library [84]. Depending on the size of the model and whether the model is pre-trained or fine-tuned it might make sense to use dedicated frameworks for distributed training such as DeepSpeed [244].

2. **Set-up Computing Resources**
   Appropriate computing resources for the training need to be set up, which includes allocating the computing hardware and setting up the environments for training, this includes installing appropriate software and tooling.

3. **Prepare Training, Validation and Test Data**
   Potentially, the features used in machine learning training need to be revised, if, e.g., the results of rule-based models or other machine learning models should be used for a model training. This would require an iteration back to the Data Cleaning and Preprocessing phase, concluding with a newly committed and documented dataset.

4. **Define Hyper-Parameters**
   Many machine learning models have large sets of hyper-parameters which can be optimized via methods such as Random Search [245], Bayesian Optimization [246] or others, by validating results of the training on the validation split. For neural networks, methods such as Neural Architecture Search can be used to optimize the architecture of a neural network towards a task [247]. When training foundation models, the amount of hyper-parameters that are tuned are often limited due to the cost of training. Especially during fine-tuning only a small set of hyper-parameters such as the learning rate and weight decay are optimized. Hence, it is common practice to resort to standard architectures and avoid costly architecture searches. Before the training commences, a suitable set of hyper-parameter ranges should be selected, that should be explored. Especially on smaller datasets, training via cross-validation should be considered for hyper-parameter tuning.

5. **Train Models**
   Once everything is set up the training of the models is started. Appropriate monitoring of the model loss and important metrics should be implemented. Further, problems in training should be logged. It is good practice to save intermediate checkpoints of models to avoid full re-training should errors in training occur.

6. **Optimize Models**
   Models can be further optimized with regards to the business requirements, e.g., by using calibration methods to get more representative confidence scores [248]. Further optimizations include model compression [249] or ensembling methods, which have been shown to even enhance prompt based methods [250]. If training data can be generated in regular work processes, active learning might be a viable option to reduce annotation effort and sample unannoted data efficiently.

7. **Commit Code, Results and Models**
   The training code should be committed to the Software Hub. Before committing, software engineers or other ML engineers and data scientists should review the code. All results should be stored alongside the models and their metadata in the Model Hub. Further, the programming environment is committed to ensure reusability.

8. **Document the Training**
   The training process should be documented in the Knowledge Hub for reproducibility. Choices of parameter ranges should be explained.

Model/Rule Evaluation

The evaluation of individual models and rules is a key checkpoint of each project. Evaluation determines whether the project proceeds into the subsequent phases or whether further iterations are necessary. It can also serve as a breaking point for the whole project if the modeling problem turns out to be too difficult to achieve. The goal of the evaluation is to assess the performance of models and rules and to pick the most promising candidates for deployment. Later, in the Evaluation phase, the complete application is assessed, i.e., also the interplay of different models and rules. Next, we describe the activities of the Model/Rule Evaluation subphase.

1. **Quantitative Assessment**
   The performance of each model is assessed and reported to the client. To this end appropriate statistics are created and visualized for better understanding of the customer. Visualizations such as confusion matrices can be used to better understand the type of errors the model makes.

2. **Qualitative Assessment**
   For the qualitative assessment, domain experts and end users should inspect the results of the models together with the data scientists. This is especially helpful to understand the strengths and weaknesses of models. Further, a qualitative assessment is particularly helpful for unsupervised tasks such as topic modeling or clustering.

3. **Explainability Methods**
   Explainability methods can be used to debug model behavior [217] and to provide better understanding which features a model relies on for predictions. In addition, these models can help users better understand model behavior in the application.

4. **Select Best Models for Production**
   Based on the predefined metrics the best models are chosen. The best model is not necessarily the model with the highest performance value. What is best, is determined based on the requirements and the aforementioned characteristics. Another influential factor is the user experience when using the model. It could be that users prefer models with high precision and lower recall over models with lower precision and higher recall. If the models prove to be good enough for production, they are selected for deployment, otherwise another modeling iteration starts.

5. **Document Assessments**
   The results from the assessments are documented in the Knowledge Hub and shared with the team and the client. Experiments are stored alongside the models in the Model Hub.

Model/Rule Versioning

In this subphase the models and rules are stored in the Model Hub and the respective pipelines are committed to the Software Hub. Next, we describe the activities of the Model/Rule Versioning subphase.

1. **Implement Training Pipelines**
   Training pipelines are developed and committed to the Software Hub to enable automated retraining in the future.

2. **Implement Inference Pipelines**
   Inference pipelines are developed for each model and committed to the Software Hub.

3. **Commit Models for Serving**
   Model Serving components are developed for the models. This is typically done with containers and REST APIs [125]. The serving component is committed to the Model Hub and the according code to the Software Hub.

4. **Create Model Cards**
   Model Cards [190] are used to describe the models in the Model Hub. They contain information on the task the model was trained on, the dataset, the performance, its intended use and so on.

Rationale

Compared to projects in which traditional machine learning models are used, the Modeling Phase has significantly changed when using foundation models. A key factor to consider is that foundation models are reusable and adaptable artifacts, which reduce the need for annotated training data. This holds even for tasks which are typically modeled as supervised learning problems. If appropriate models are available, modeling is either reduced to simply using those models or to prompt engineering. We generally suggest approaching modeling in a way that the least complex solutions are used first and then gradually more complex solutions are explored such as fine-tuning models or even pre-training models from scratch.

We place an emphasis on model management, i.e., tracking models and experiments in a dedicated Model Hub which is made available for the rest of the team and organization if permissible. This way, models can be reused in future projects and be built upon with further training data.

In contrast to all other methodologies, we design this phase such that it does not only concern a single machine learning model. Many projects require the interplay of multiple models and rule-based components, which interact with each other.

The Modeling Phase addresses 48 characteristics. It places a special emphasis on characteristics belonging to the Model subcategory in the Artifact and Asset Management category. This makes sense as this phase mainly deals with reusing, producing and, more generally, managing models. From the Process category all of the characteristics belonging to the Model Development subcategory are addressed, as well as some characteristics regarding User Experience Design and Monitoring. The phase involves characteristics from the Management and Governance category such as resource considerations, e.g., when pre-training large models, and the involvement of various stakeholders such as users, domain experts and legal experts.

5.3.5 Application Development Phase

The Application Development phase is concerned with the development of applications which embed models. Often, a project deliverable is not just a trained model, but an application which wraps models and provides further functionality such as a business logic that verifies and transforms model results and a user interface to display the results and let users interact with them. Software engineering and application development in data science is different from traditional software engineering [6]. However, most methodologies omit this phase. In fact, out of the methodologies we presented in Chapter 4 only the Data2Value methodology considered a phase for system development [17]. This phase can mostly

be executed in parallel to the Data Gathering and Understanding, Data Cleaning and Preprocessing and Modeling phases. However, there are certain checkpoints in which dependencies from these phases are needed, such as data pipelines, model pipelines and the model artifacts themselves. We will now describe the subphases of the Application Development phase. We now describe the subsections of the Modeling phase. A compact summary of the phase is depicted in Figure 5.7. Detailed descriptions of the roles, artifacts and tools are in Subsection A.5.5 in the appendix.

Application Design

The initial design from the Mutual Understanding phase is revisited and refined based on new insights. Defining the interfaces for the application early-on can ease the integration process for the customer, as they can already develop against the interface specification. Reusing existing software components and models can enable an early delivery of a prototypical application which enables integration testing on the customer side even before project-specific model artifacts have been produced. Next, we list the activities of the Application Design subphase.

1. **Revisit Requirements**
   The Application Development phase starts with revisiting the requirements of the project. In particular, requirements with respect to available hardware in the productive environment, expected throughput, safety, integration into other systems as well as interaction with other systems should be considered.

2. **Architecture Design**
   Based on the requirements and the insights from previous phases and activities, the architecture design is revised. Current best practice in data science is to develop modular architectures using microservices, which allow for a good separation of concerns and an easy exchange of modules and machine learning components. Whenever possible, existing software components of the team should be reused. To this end, the relevant stakeholders such as product owners should be included in the development process, to understand how the components need to be modified and customized, how they can be integrated and to ease planning. In addition, the team should research publicly available software that can be used. The architecture design should account for the data processing mode, e.g., stream vs. batch, interfaces between different components, possible inputs and outputs, database designs, integration with other systems as well as the integration of the machine learning based components and their workflow. Software complexity should be kept low, to increase maintainability. Further, the design should ensure scalability and reusability of code.

3. **Interface Definition**
   Defining the interfaces between the various components early in the development helps parallelizing development efforts on the team and the client side. The interface definition should be created jointly with the developers of the client.

4. **User Experience Design**
   User experience design for applications with machine learning components has to consider other characteristics than traditional software development. One characteristic is the interaction of users with the system. Should a user interact with the system, e.g., by intervening, and if

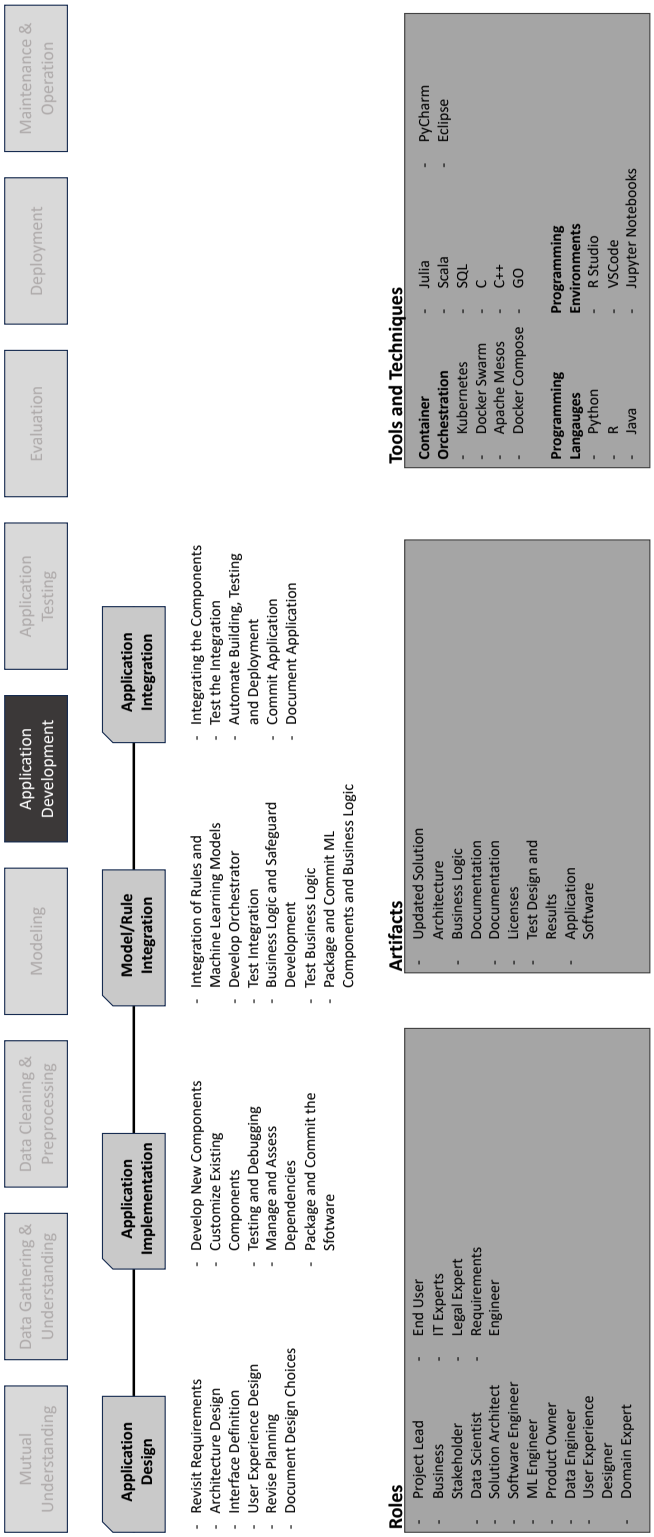

Figure 5.7: Overview of the Application Development phase.

so, how often? Accountability is also an important characteristic. Who is accountable for mistakes by the system? Does a user need to verify the output of the system? Another important characteristic is the understandability and transparency of the system. Making decisions or predictions by a system understandable and transparent can increase the trust of users in the system. To increase the acceptance of the end users, they should be involved in this phase and decisions should be discussed with them.

5. **Revise planning**
   Based on the new concepts, the planning should be revised. The plan also should account for the availability of end users and the development schedule of the machine learning based components. The integration of the machine learning components should be tackled early on. Projects can benefit from reusing already existing models. However, as long as they are not available, the software developers can use dummy services instead. Development sprints should be planned and the respective development tickets should be created.

6. **Document Design Choices**
   All design choices should be communicated and discussed with the relevant stakeholders on the client side. They should be documented in the Knowledge Hub.

Application Implementation

The development of ML-based applications should adhere to best software development practices, including peer-reviewing code, commenting, testing and debugging. In early iterations the application will not contain the final models, yet. Hence, model stubs or drivers should be used to simulate model behavior. Ideally, existing pre-trained models can be used before project-specific models are created, such that the input and output behavior of them can be simulated before they are readily trained and evaluated. We will now list the activities of the Application Implementation subphase.

1. **Development of new Components**
   Based on the requirements, API specifications and the application design, new components are developed. To ensure high software quality, quality assurance and code versioning processes should be established. If possible, code should always be peer-reviewed, preferably by experienced team members, commented and tests should be designed during development. To ease implementation software versioning tools should be used.

2. **Customization of existing Components**
   Any customization of existing components should be implemented and tests for them should be developed. If these components are used over the course of multiple projects, the customization needs proper scheduling and planning to avoid resource conflicts and conflicting changes of the software. To this end the respective product owners should oversee and coordinate the process.

3. **Testing and Debugging**
   The individual components should be tested. Whenever needed, stubs should be used to test communication with components which are not yet ready. An automated testing pipeline should be developed as part of the continuous integration process. Debugging will take up a significant amount of time.

4. **Manage and Assess Dependencies**
   If the software relies on third-party software, it is important to track the dependencies and analyze their licenses. As part of the development and build process automated updating of license files should be implemented. If necessary, the licenses should be evaluated by a legal expert. Dependency files should be stored alongside the code.

5. **Package and Commit the Software**
   The individual components should be committed to the Software Hub. This includes the code as well as the packaged component containers. Using containers helps assure easier dependency management.

6. **Creating a Documentation**
   A documentation for each component is created and shared in the Knowledge Hub.

Model/Rule Integration

Once first models and rules are available, they are integrated into the software components of the application. The trend of using foundation models, built upon the same transformer architecture, helps easing the integration process, as most models share the same code base and only the exchange of some model files is required when replacing models. Following, we describe the activities of the Model/Rule Integration subphase.

1. **Integration of Rules and Machine Learning Models**
   Rules and models are integrated in the components with their respective data and inference pipelines. If the models are built compatible to standard libraries such as spaCy or the Huggingface Transformers library, exchanging models over time becomes very simple, as long as the problem type of the model and the required data does not change. This way models can easily be exchange over the course of the project iterations, whenever the model quality is improved.

2. **Develop Orchestrator**:
   Orchestrators manage the workflow of the individual components. They handle the flow of data and actions of the application. Foundation models can also be used to determine the correct actions given an input and serve as orchestrators [251]. However, this requires the implementation of guardrails which ensure proper functionality.

3. **Test Integration**
   The correct integration of the rules and the models should be tested. One way of doing this is by using configuration settings and data samples for which the model output is known and ingesting these samples into the respective components.

4. **Business Logic and Safeguard Development**
   It is common practice to have a business logic which is separated from the models. This business logic should be developed together with domain experts and users from the client and can be used to verify or correct the output of the machine learning components. The business logic itself is often a rule-based system, which serves as a safeguard that mitigates model limitations. Another common use of business logic is to verify that model outputs are in

the right format, e.g., that an amount that a model identified corresponds to a decimal number. In applications with generative foundation models special safeguards are used to ensure that no harmful, hallucinated or out-of-domain output is generated [252].

5. **Test Business Logic**
   The developers should design tests which verify the proper functionality of the business logic.
6. **Package and Commit ML Components and Business Logic**
   The code and the packaged containers of the ML components and the business logic are committed to the Software Hub.

Application Integration

Once all components are ready, they are integrated into the complete application. Pipelines for automating the build, testing and deployment processes are set-up and everything is committed to the Software Hub. Next, we list the activities of the Application Integration subphase.

1. **Integrating the Components**
   Tools for the integration of the components are used to establish communication between them. This includes network configuration, setting up volumes and defining workflows.
2. **Test the Integration**
   End-to-end tests of the integrated application are defined and executed to assure proper communication and workflows between the different components.
3. **Automate Building, Testing and Deployment**
   The process of building, testing and deploying the application is automated via continuous integration and deployment tools.
4. **Commit Application**
   The configuration and test files, the code and build scripts are committed to the Software Hub.
5. **Document Application**
   The application documentation is committed to the Knowledge Hub and shared with the team and the client.

Rationale

Application development is an important part of many data science projects, but not tackled by most methodologies. Developing applications with AI/ML components is a distinct challenge which requires a lot of considerations regarding user experience, solution design and also integration with other systems.

We have designed this phase such that it builds upon available artifacts and can be executed in parallel to other phases such as the Modeling Phase. The application will gradually integrate parts of the other phases such as data and model pipelines and the models themselves.

This phase addresses 41 characteristics. Regarding the Process category it is mostly concerned with the characteristics from the Application Development subcategory, which intuitively makes sense. Further, it addresses all characteristics belonging to the Software/Code subcategory of the Artifact and Asset Management category. Finally, stakeholder involvement characteristics are addressed.

5.3.6 Application Testing Phase

The Application Testing phase focuses on assessing the quality of the application. To this end the application is deployed on a test environment, which closely mimics the production environment. Ideally, the application is integrated with the customer system and user tests are performed to evaluate user acceptance. If necessary, the application is optimized to meet performance, cost and user requirements. We will now list the subphases of the Application Testing phase. A compact summary of the phase is depicted in Figure 5.8. Detailed descriptions of the roles, artifacts and tools are in Subsection A.5.6 in the appendix.

Test System Deployment & Tests

Before deploying the application in a production environment, the system is deployed on a test system. This way the production processes are not disrupted but the team and the client can gather a realistic understanding of the application's performance. Following, we describe the activities of the Internal System Tests subphase.

1. **Set up Test System**
   A test system should be set up which mimics the production system. This way the application can be tested on close to production settings without interrupting any running systems.

2. **Deploy Application on Test System**
   The deployment pipeline is tested on the test system and the application is deployed for usage.

3. **Perform Automated Tests**
   Automated tests are run to assure that the application runs as expected. Stress tests are performed to test the reliability of the system and to simulate high load. These tests can be used to determine the scalability of the application. Further the application should be tested for safety and security.

4. **Debug System**
   The software engineers debug the application given errors that arise on the test system.

5. **Integrate with Customer System**
   If the client wants to integrate the application into their own software, this should be done on the test system. Software engineers should help the client IT experts with the integration.

6. **Perform Integration Tests**
   The integration of the application with the client software should be tested.

7. **Document Test Results and Metrics**
   The results of the application tests are documented in the Knowledge Hub and discussed with the client.

User Acceptance Tests

User acceptance is a key success factor of data science projects. To assure user acceptance, tests are performed with selected users. Next, we describe the activities of the User Acceptance Tests subphase.

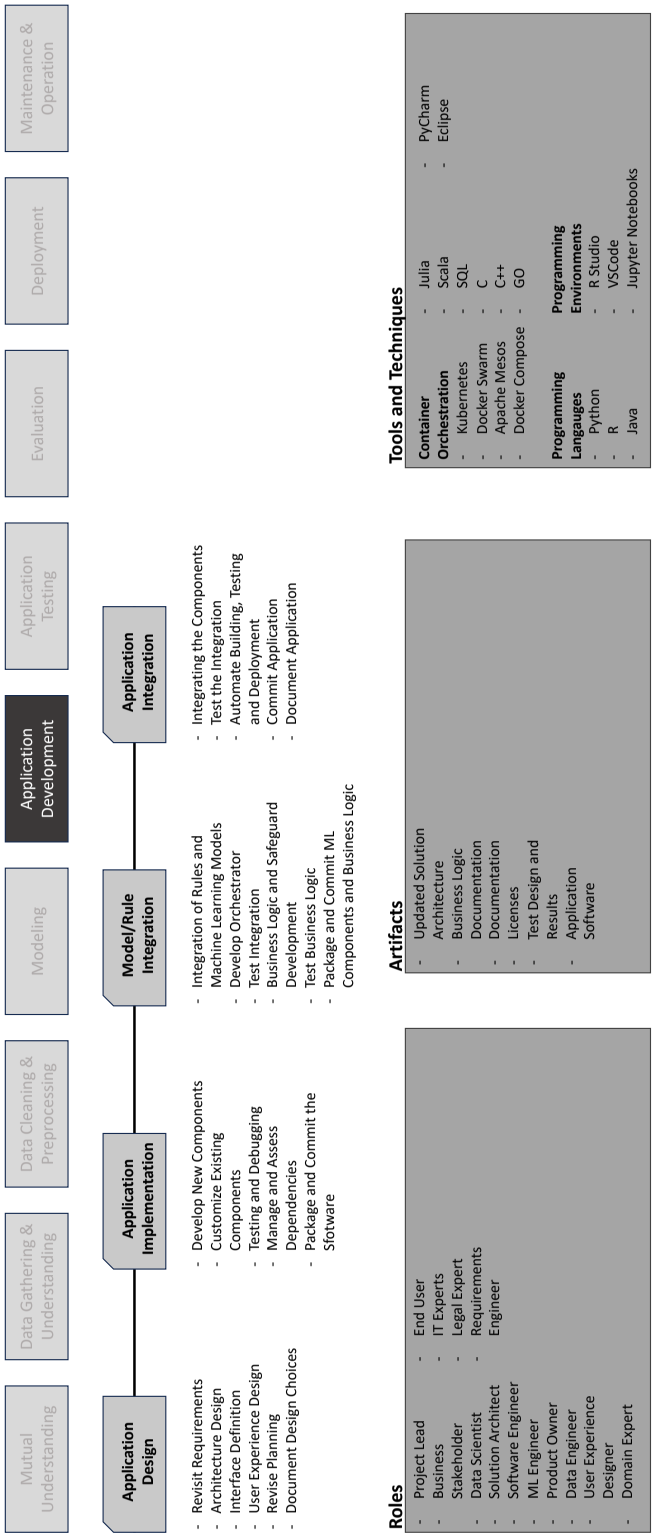

Figure 5.8: Overview of the Application Testing phase.

1. **Select and Brief Test Users**
   Representative test users should be selected. Ideally, these users represent the end-users which will use the system and which will determine the benefit of using the application. Proper briefing of these users is required, as they might not be used to machine-learning based software and the uncertainty and errors which occur while using machine learning model. This is especially important in early iterations of the project as model and software quality might still not be representative of the end product.

2. **Perform User Acceptance Tests**
   The application is tested by the test users with respect to characteristics such as the transparency, the usability and ease of use, the design and the overall functionality. These tests give an indication of the complexity of the application.

3. **Document Test Results**
   The test results are documented and shared in the Knowledge Hub and communicated to the client.

Application Optimization

Based on the insights of the tests, optimizations of the application are defined, scheduled and implemented. This might affect the project plan and requires prioritization and transparent communication with the customer. Next, we list the activities of the Application Optimization subphase.

1. **Assess Test Results**
   The test results are assessed together with the client. Potential optimizations are prioritized and the decisions as to what to optimize and what not are documented in the Knowledge Hub.

2. **Plan Optimizations**
   The selected optimizations are planned with regards to resources and schedule.

3. **Update and Communicate Project Plan**
   The project plan is updated according to the planned optimizations. The updated project plan is communicated to the client and shared in the Knowledge Hub.

Rationale

The Application Testing phase is dedicated to testing the application, its integration with the machine learning components and the integration in other systems. The tests focus on various quality aspects of the application as well as user acceptance. Tests are performed in an environment which closely mimics production settings.

This phase addresses 23 characteristics. It is mainly involved with Application Evaluation subcategory characteristics and characteristics which belong to the Software/Code subcategory in the Artifact and Asset Management category. Aside of that, it handles characteristics regarding the involvement of business stakeholders and users and regarding the infrastructure used in the project.

5.3.7 Evaluation Phase

In addition to the evaluation after modeling, it is necessary to evaluate the complete application with all its dependencies between model- and rule-based components. This evaluation should not only assess the performance based on machine learning metrics but also the business driven KPIs. This phase represents another project checkpoint which might lead to iteration cycles over all previous phases. It also serves as gateway for a deployment decision. Following, we describe the subphases of the Evaluation phase. A compact summary of the phase is depicted in Figure 5.9. Detailed descriptions of the roles, artifacts and tools are in Subsection A.5.7 in the appendix.

Application Evaluation

The application is evaluated with respect to machine-learning metrics, as well as business KPIs. Next, we list the activities of the Application Evaluation subphase.

1. **Performance Assessment**
   In this phase first a quantitative evaluation of the complete application is performed. This differs from the evaluation of single models as the whole pipeline is tested. This includes ingesting data, applying possibly multiple models and the business logic as well as other post-processing. The performance on an independent test-set is reported. The application is also evaluated with regards to fairness, bias and transparency of the results. Other important metrics include the speed and hardware demands.

2. **Safeguard Evaluation**
   When generative models are used, users and domain experts should also perform tests of the safeguards, i.e., they should try to break the safeguards. Interesting aspects are the factual correctness of the answers, whether the answer stays in the right domain, whether the answers are discriminatory, and so on.

3. **Business Assessment**
   Aside of performance metrics, the application is evaluated with respect to the business KPIs that had been defined. This could require time and involve end users and business stakeholders which need to use the application.

4. **Results Documentation**
   The results of the evaluations are documented in the Knowledge Hub and shared with the client.

Performance Investigation

The results of the evaluation are used to further investigate the performance of the application and to derive next steps. If the application meets all requirements, it is ready for deployment. Otherwise, the project either enters further lifecycle iterations or ends. We will now describe the activities of the Performance Investigation subphase.

1. **Error Investigation**
   The results of the evaluation of the complete application are investigated and possible sources of error are determined. If necessary, the output of single models is isolated and verified, to

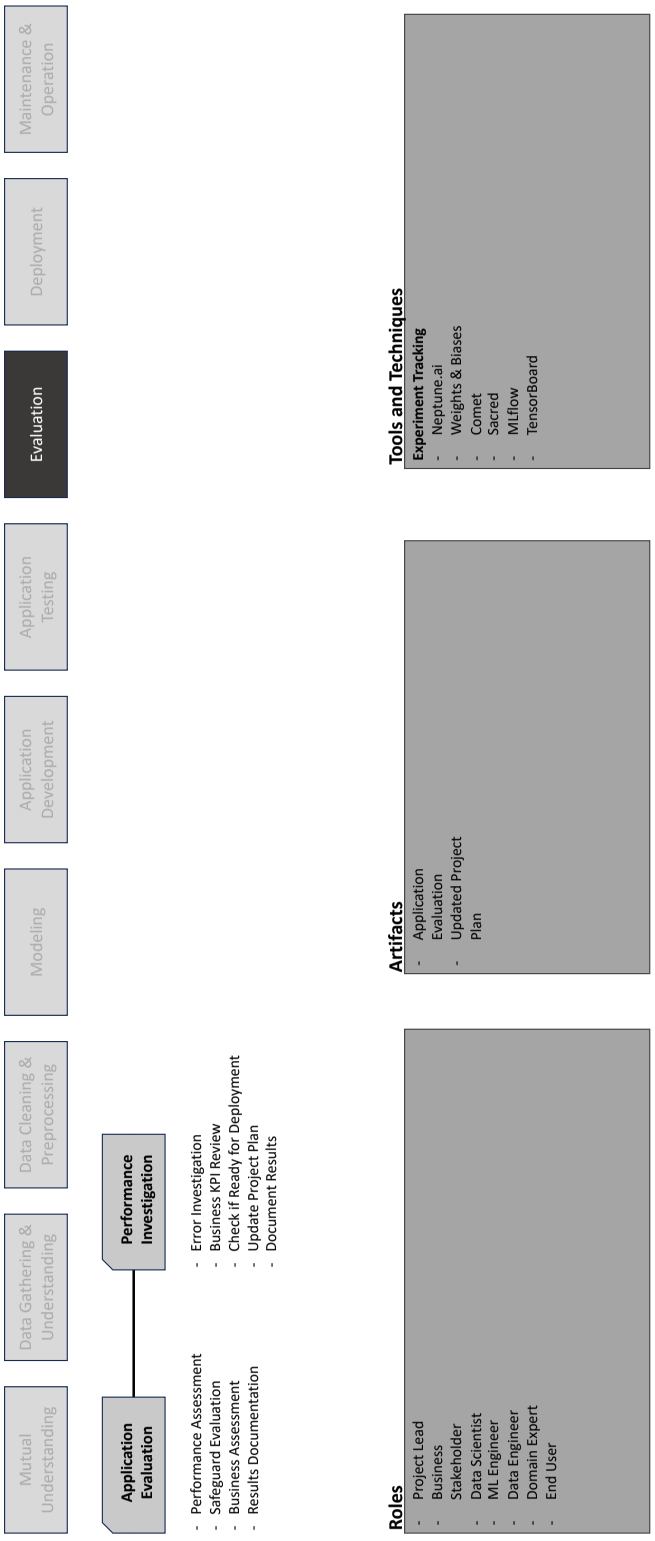

Figure 5.9: Overview of the Evaluation phase.

understand the sources of errors. Explainability methods and visualizations can support this step. They can serve as a basis for discussing results with domain experts and end users.

2. **Business KPI Review**
   The KPIs are analyzed with respect to their impact on the business. Various areas are analyzed, such as the usability of the system, the degree of automation, the trust of the users and so on. Further, the cost of using the system should be assessed, this includes the cost erroneous application outputs.

3. **Check if ready for Deployment**
   The client and the team make a joint decision whether the application is ready for deployment. This does not necessary mean that the development project ends here, it could be that an intermediate stage was reached, where the application is already ready for productive use but not finished, yet. If the decision is made that the application is not ready, another iteration of the project life cycle starts based on the identified potentials for improvement and the remaining project budget. If the application is ready for use, the deployment phase begins. This decision should involve important business stakeholders as well as management.

4. **Update Project Plan**
   Based on the results of the evaluation the project plan is updated. That includes either the planning for the upcoming deployment or the planning for the next iteration of improvement.

5. **Document Results**
   The results of the evaluation phase and the updated project plan are documented in the Knowledge Hub and shared with the client.

Rationale

We modeled the Evaluation phase such that it accounts for both the evaluation of multiple model and rule components, as well as the complete application. This joint evaluation of the complete application is a significant difference compared to other methodologies. Evaluation should consider machine learning metrics, business KPIs and user acceptance. Visualizing and explaining model predictions is considered an important factor for evaluation, as it allows the team and the client to get an understanding of the application behavior.

This phase covers 23 characteristics in all four categories. Regarding the Process category the phase mainly deals with characteristics concerning the model and application evaluation, but it also considers planning and communication with the relevant stakeholders. This is particularly important as the Evaluation phase serves as a gateway for either moving towards deployment, ending the project or doing further project iterations. To this end, out of the characteristics in the Management and Governance category, the involvement of all relevant stakeholders is considered as particularly important. In the Artifact and Asset Management category, characteristics regarding the performance of the application and models are mentioned as well as artifacts and documentation and other project management artifacts. Finally, concerning the Technology and Infrastructure category, explainability and visualization tools are mentioned to be important for evaluation and investigation of individual samples.

5.3.8 Deployment Phase

The Deployment phase is concerned with deploying the final application on the target system of the client and getting stakeholder approvement for the project. All handovers are done here and the project transitions into the phase of operation and maintenance. All reusable artifacts, as well as the gathered insights from the project, are shared with the team. Next, we describe the subphases of the Deployment phase. A compact summary of the phase is depicted in Figure 5.10. Detailed descriptions of the roles, artifacts and tools are in Subsection A.5.8 in the appendix.

Target System Deployment

The application is deployed on the target system and integrated with existing software. After deployment, automated tests are performed to ensure functionality. Following, we list the activities of the Target System Deployment subphase.

1. **Schedule Deployment**
   The deployment is scheduled in a way that minimizes down-times for the client.
2. **Schedule Monitoring and Maintenance**
   A monitoring and maintenance plan is created and scheduled.
3. **Prepare Target System**
   The target system is prepared for deployment by installing the necessary software and tools and setting up the hardware. The deployment pipeline is set up for the target system. Further the system is configured for monitoring.
4. **Deploy Application**
   The application is deployed on the target system for usage.
5. **Perform Integration Tests**
   The integration of the application with the client software is tested.

Customer Acceptance

Customer stakeholders are educated on the results of the project and in the usage of the application. The application itself is handed over to the IT experts of the client and a closing workshop is held with all relevant stakeholders. Next, we describe the activities of the Customer Acceptance subphase.

1. **Educate End Users**
   Workshops are held to educate the end users about application usage. This should include features, performance and limitations of the system. Ideally, the end users are briefed about the uncertainty of machine learning results and proper interaction with the system. If explainability methods and tools are applied, users should get a general overview of how to interpret them. Questions and answers about the system should be documented in the Knowledge Hub.
2. **Hand over Application**
   The documentation of the project and the application is handed over to the client. The IT of the client is educated in monitoring the application.

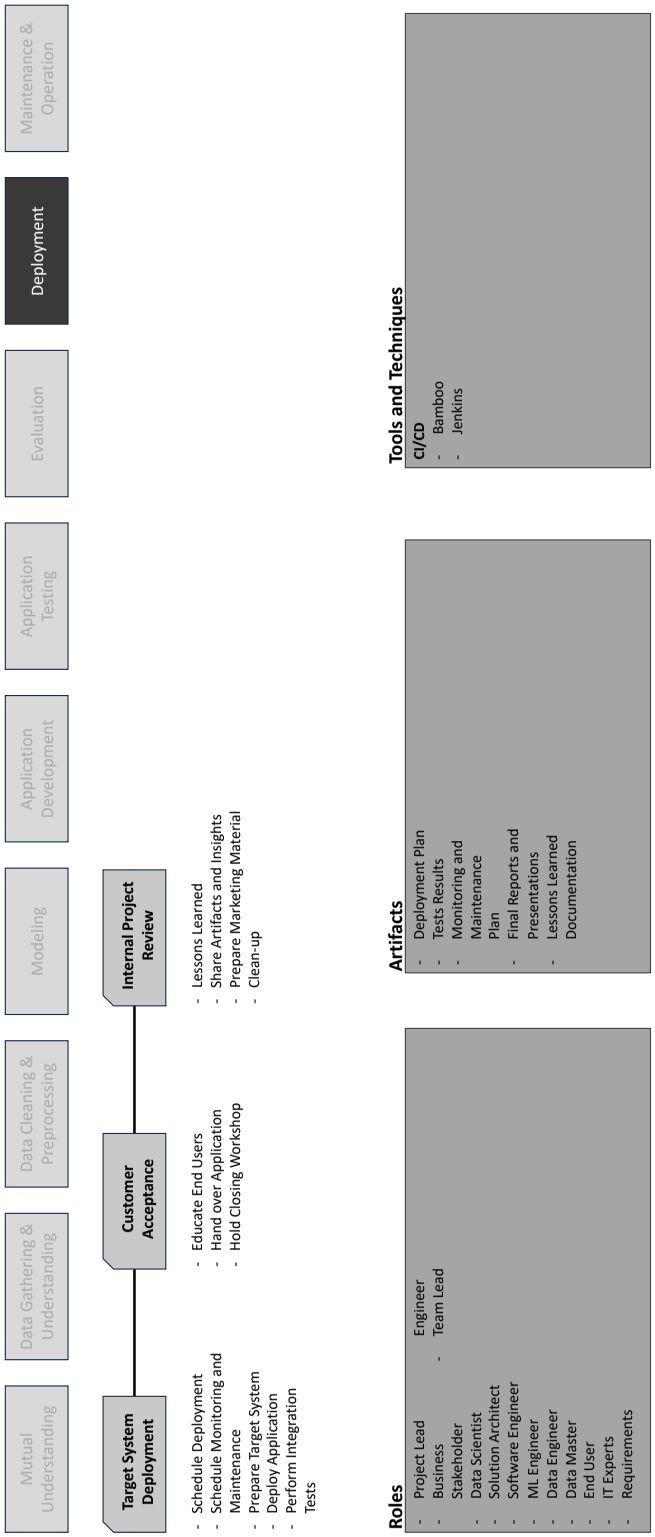

Figure 5.10: Overview of the Deployment phase.

3. **Hold closing workshop**
   In a closing workshop the project and its results are presented to all relevant stakeholders on the client side.

Internal Project Review

The project is reviewed internally and all important insights as well as reusable artifacts are shared with the team. Next, we list the activities of the Internal Project Review subphase.

1. **Lessons Learned**
   A lessons learned workshop should be held with all team members to improve the process for future projects. The team should also reflect on the project process. The lessons should be well documented in the Knowledge Hub and shared with the rest of the organization.

2. **Share Artifacts and Insights**
   Newly created artifacts such as software, models and data are shared with the organization as well as insights from the project.

3. **Prepare Marketing Material**
   The project and the reusable artifacts are included in marketing material by business development for future acquisitions.

4. **Clean-up**
   Clean up all artifacts which the team is not allowed to store beyond the project, such as data, background information or models.

Rationale

Our Deployment phase considers the transition of the application from a test environment to a productive environment. This represents an important step, as the application should not or only minimally disrupt the systems running on that environment.

Customer acceptance is the most important aspect of this phase as it is the main factor for project success. This entails educating the customer in using and interpreting the results and handing everything over to them. Finally, we also deem it important to share the knowledge and artifacts created in a project within an organization and to further disseminate the results.

The Deployment Phase addresses 22 characteristics, most of which belong to the Process category. In the Process category most of the characteristics concerning the Deployment & Operations subcategory are addressed. Just like the Evaluation phase, this phase also requires the involvement of all important stakeholders. Concerning the Technology and Infrastructure category, only infrastructure is mentioned in this phase. Lastly, characteristics regarding documentation and project management artifacts are covered.

5.3.9 Maintenance and Operation Phase

The lifecycle of data science projects does not end with deployment. The application and the according models and rules are subjected to changing environments and data. This can lead to performance degradation and wrong predictions and creates the need for maintenance and updates. The quality

of the application in production needs to be monitored to alert users and developers in the case of degrading performance. To counter degradation effects, automated workflows for retraining can be used. However, sometimes new project iterations are required. Next, we describe the subphases of the Maintenance and Operation phase. A compact summary of the phase is depicted in Figure 5.11. Detailed descriptions of the roles, artifacts and tools are in Subsection A.5.9 in the appendix.

Application Monitoring

Monitoring the application and the performance of models and rules is crucial for preventing an increase of wrong predictions over time. The application should be configured in a way that all important events and, if legally possible, all incoming data is recorded, as well as the application outputs and its performance. In addition to monitoring the application behavior, application usage should be tracked for improvement and to understand user acceptance. Next, we list the activities in the Application Monitoring subphase.

1. **Define and Setup Monitoring Metrics**
   Based on the insights from training and evaluating the deployed models and application, monitoring metrics should be defined. These metrics could include distribution over class labels, document types and lengths, feature distributions, model output distributions and much more. Next to these machine learning centric metrics, metrics which monitor the application usage, throughput, latency and the business KPIs should be defined.

2. **Setup Alerts**
   Based on expected values, thresholds for alerts should be setup, that automatically alert the IT when the system performance degrades.

3. **Log Input- and Output-Information**
   If possible, all data that goes through the system should be recorded for analysis and retraining purposes. If this is not possible, e.g., for legal reasons, statistics about the data should be collected for further analysis.

4. **Log Application Usage**
   Statistics about the application usage should be collected. This could include usage times but also direct feedback from users regarding their satisfaction and acceptance of the system.

5. **Commit Artifacts**
   The respective information about the application performance should be logged in the Model and Knowledge Hubs.

Error Analysis

When the application creates faulty predictions, end users and data scientists should get insights into the causes of wrong predictions. This could involve an overall performance evaluation but also the investigation of singular data samples. Users should have the possibility to obtain explanations for model predictions and correct faulty predictions. Such user feedback should be stored for retraining of models. In addition, the user behavior should be analyzed to understand potential improvements. Whenever possible, technical assistance for the users should be provided. Further, updates should

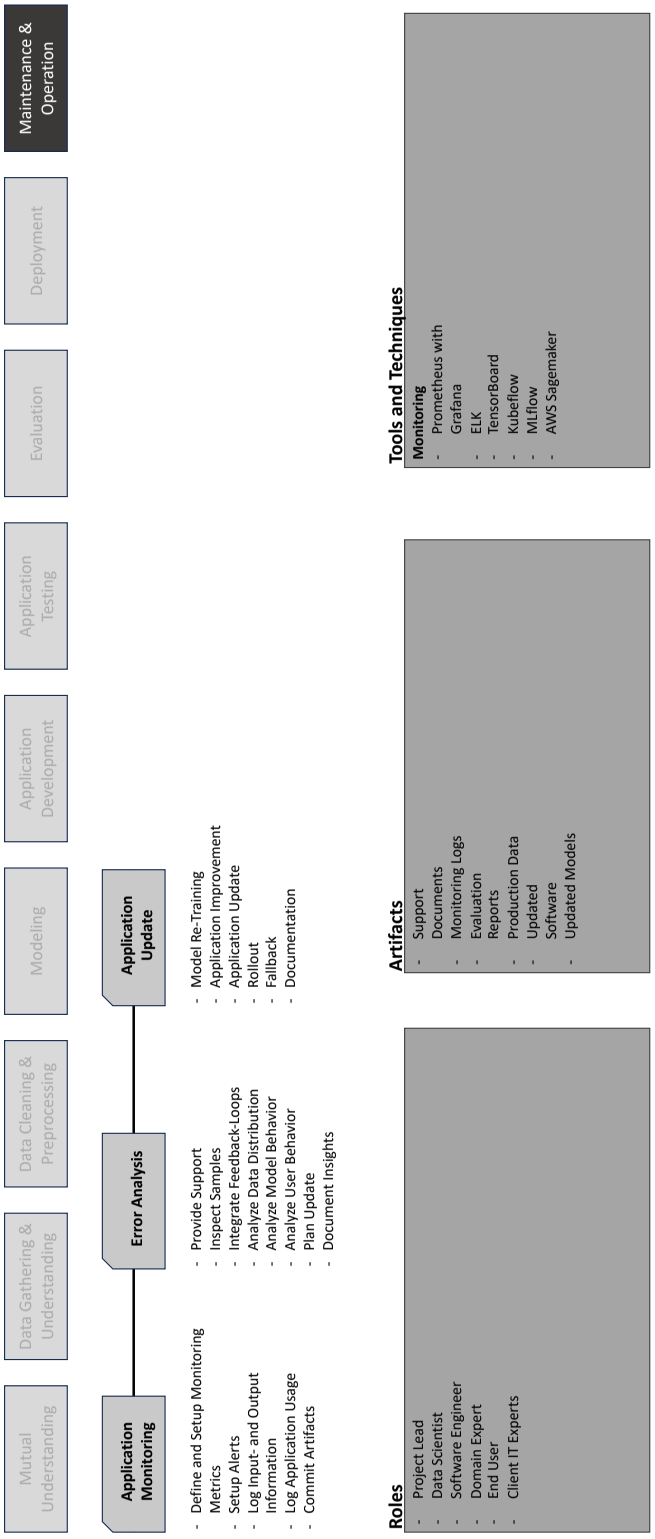

Figure 5.11: Overview of the Maintenance and Operation phase.

be planned based on results of the error analysis. Following, we describe the activities in the Error Analysis subphase.

1. **Provide Support**
   The team should provide technical assistance regarding the application and infrastructure. This includes bugfixing and answering support requests.

2. **Inspect Samples**
   Users should be able to inspect individual data samples and obtain model explanations on demand to understand wrong predictions and provide valuable input for improvements. Potentially, domain experts should be available to discuss possible reasons for predictions.

3. **Integrate Feedback-Loops**
   If possible, end users should have the possibility to correct wrong model outputs. These corrections should be stored as annotations and made available for retraining. This step could decrease retraining time significantly but might require user education. Correct labeling is not trivial and time of domain experts is scarce. Feedback loops could also be used to rate model predictions, e.g., generated texts. Such feedback could for example be used to learn to improve models based on methods such as reinforcement learning [72].

4. **Analyze Data Distribution**
   Once alerts about the model performance are triggered or as soon as maintenance intervals occur, the data scientists should analyze the distribution of the monitored data. This could give valuable insights in the cause of model degradation.

5. **Analyze Model Behavior**
   Similarly as in the Evaluation phase, the model behavior on new samples should be studied qualitatively and quantitatively.

6. **Analyze User Behavior**
   The user behavior should be analyzed and users should be interviewed to understand potential for improvement.

7. **Plan Update**
   Based on the insights of the analysis, the team should create a plan for possible updates and discuss it with the customer.

8. **Document Insights**
   The insights of the analysis are stored in the Knowledge Hub and communicated.

Application Update

The application, models and rules should be updated if necessary. Model updates are not trivial and fallback mechanisms such as human processing or rollback to previous versions should be enabled. If possible, model retraining should be automated. This subphase could trigger a complete iteration through the project lifecycle, so the goal should be to automate as many steps as possible. Next, we list the activities in the Application Update subphase.

1. **Model Re-Training**
   If necessary, a model re-training is scheduled. Depending on the cause of the model degradation, re-training can be as simple as fine-tuning the model on new data and as complicated as a complete iteration through the project lifecycle. If possible, the data and model pipelines should be used together with the CI/CD pipeline to automate the re-training as much as possible. It might be necessary to trigger a new iteration of the annotation subphase if newly annotated data is required. The model should be evaluated independently and embedded in the application. Its performance should be compared with the old model. Finally, if the model reaches a satisfying quality, it should be committed to the Model Hub and marked for deployment.

2. **Application Improvement**
   The application is improved based on the analysis of the system behavior, user acceptance and user requests. The new application version is tested and the software and code are committed to the Software Hub.

3. **Application Update** New models are integrated in the application. A new version of the application is built, deployed on the test environment and tested.

4. **Rollout**
   The new application is rolled out to production. If possible canary deployments are used to verify proper functionality of the application, before rolling it out to all production environments. Additionally, A/B testing can be used to compare application behavior before and after the update.

5. **Fallback**
   Fallback options should be in place in case there are problems with the new application or models. Fallbacks can include rollbacks to previous versions and other human or automated fallback options.

6. **Documentation**
   The new models and the application are documented and shared via Knowledge Hub.

Rationale

The Maintenance and Operation Phase is concerned with the lifecycle of the application and its models after their operative use has begun. We modeled this phase after best practices from MLOps. This phase is of particular importance, as the real usage of an application starts after deployment. It is often ignored by older methodologies. We put an emphasis on logging and evaluating the behavior of the application and its performance. Support, model retraining and application updates play an important role for the long-term acceptance of an application. After updating, we emphasize safety precautions, such as canary deployments, A/B tests, rollback options and further integration tests.

This phase covers 31 characteristics, mostly characteristics of the Process category dealing with evaluation and deployment and operations. Only few characteristics from the other categories are covered here, such as the involvement of users and domain experts, continuous knowledge and information management as well as model and data management and the use of explainability and visualization tools to interpret the application behavior.

5.4 Discussion

By design, our methodology covers the complete catalog of characteristics described in Chapter 3. Different parts of the methodology focus on different categories of characteristics. For example, we can observe that most of the Management and Governance category is already covered by the Mutual Understanding Phase. Artifact and Asset Management characteristics play an important role in every phase and especially documentation is handled in every phase. The artifacts such as the model, data and software/code are predominantly handled in separate phases, e.g., data in the Data Gathering and Understanding phase and the Data Cleaning and Preprocessing phase. In contrast to existing methodologies, which are centered around data, our methodology is centered around the usage of foundation models and considers the implications of using them. This does not imply that data has a lesser importance but rather that model artifacts get more important.

5.4.1 Reviewing the Guiding Principles

The coverage of the characteristics in the different phases makes the use of our guiding principles apparent. The Holistic Thinking principle is most apparent in the way we deal with artifact and asset management. It is handled throughout every phase of a project and results are always documented and committed for reuse by others.

In accordance with our guiding principles, users and domain experts are either separately or jointly involved in every phase of our methodology. They serve as sources of knowledge regarding the domain and data, help with annotations and modeling and serve as a benchmark for evaluation and acceptance testing. Further, we can observe that we recommend communication and collaboration with important stakeholders in each phase.

In our guiding principles we suggest embracing visualizations. According to our methodology, visualization tools should be used in the Mutual Understanding, Data Gathering and Understanding, Modeling, Evaluation and Maintenance and Operation phases.

Finally, the principle of frequent delivery and verification is also evident via multiple characteristics. Again, the Artifact and Asset Management characteristics show that results are committed and presented to the client frequently. Further, we propose multiple checkpoints with customers to revise planning, i.e., important stakeholders verify results and decide whether and how to proceed.

5.4.2 Suitability for NLU Projects

In Chapter 3 we derived project-relevant characteristics by analyzing 26 NLU case studies. We will now revisit these characteristics and see how and where they are covered in our methodology. We focus on the top-10 characteristics, which are each mentioned in at least five projects:

1. **Mutual Understanding**: Taking enough time for building a mutual understanding of both the domain, NLU technologies and the business goals crucially influenced the outcome of ten projects. We cover the according characteristics exhaustively in the Mutual Understanding phase and describe how to obtain meaningful use cases, business goals and KPIs, as well as which stakeholders to involve at which point in time. We further introduced an Acquisition subphase to highlight that the mutual understanding between clients and the team already begins before the project start.

2. **Team Organization**: The organization of a team, i.e. the correct staffing and skills, has played a major role in multiple projects. We provide role descriptions, give staffing recommendations and assign roles to each of the phases. We further describe the responsibilities of the different roles and suggest meeting modes. Further, we establish a Knowledge Hub and standardized project templates, to mitigate problems when team members are reassigned to other projects.

3. **Data Augmentation**: Data augmentation, and in particular data annotation, was an important characteristic of many projects. To this end, we introduce a detailed data annotation subphase and data augmentation activities. We also provide tool suggestions for both.

4. **Model Reuse**: Model reuse was mentioned as a success factor in eight case studies. It is a central concept in our methodology, supported by the holistic thinking guiding principle, the model hub, and implemented explicitly in the mutual understanding and modeling phases. We emphasize the importance of model reuse as a strategic component for teams and how this eases project initiations and modeling. Our methodology is centered around foundation model usage. Foundation models are inherently reusable and we discuss the implications of using them in almost every phase.

5. **Software/Code Reusability**: Reusability of software and code is a major component of our methodology. It is addressed in our guiding principles, via the Software Hub and throughout different phases such as the Application Development phase. Further, for each phase we give recommendations regarding available public software. We also include the product owner role, with the responsibility of managing software components across projects.

6. **Data Tools**: Data tools and especially annotation tools play an important role in several case studies, in which annotated data was not available from the start. We recommend various tools for data processing, as well as automated and manual data labeling. We describe their use in the data gathering and understanding and data cleaning and preprocessing phases.

7. **Involvement of Domain Experts**: Not involving domain experts in projects posed a challenge in many of our case studies. Their involvement was an important factor for success, in particular their contributions to domain understanding and data annotation. Our guiding principles call for involving domain experts whenever possible. This is reflected by the fact that they are included in every phase of our methodology. We also address the problem of having limited access to domain experts, e.g. by highlighting ways for automating some of their tasks, such as creating annotations automatically via LLMs or structured data.

8. **Stakeholder Communication**: The communication with stakeholders is important throughout our methodology. We provide clear role descriptions and responsibilities and highlight the involvement of client stakeholders in all relevant activities. Further, we establish a Knowledge and Information Hub to share and communicate project progress and results.

9. **Project Management**: Many of the case studies we analyzed suffered from project management issues. We suggest the involvement of multiple roles to manage projects, such as the project lead, team lead and business stakeholders. Further, we propose checkpoints at which the project lead should revise planning and communicate about the project plan with relevant business stakeholders. In our guiding principles we propose an agile approach to data science projects

and suggest the use of data science trajectories for better planning. Lastly, we suggest multiple project management artifacts across the whole project lifecycle.

10. **Visualization Tools**: Visualization tools helped domain experts and users to better understand modeling results and data insights. They also helped data scientists to debug their model behavior. We provide details on when and how to use visualization tools in the Mutual Understanding, Data Gathering and Understanding, Modeling, Evaluation and Maintenance and Operation Phases. Further, we list some visualization tools for NLU.

5.4.3 Conclusion

Based on our analysis of previous methodologies and our catalog of project characteristics, we propose a foundation model centric data science project methodology. Our methodology is the only methodology considering foundation models. Further, our methodology is the only methodology covering every category of characteristics of our catalog.

We build our methodology based on guiding principles which reflect important characteristics of data science projects. A central element of our methodology is the introduction of artifact hubs for models, knowledge and information, data and software. We encourage managing artifacts across projects, i.e., artifacts have life cycles which span over individual projects.

Our methodology structures projects into 9 phases and 34 subphases with various activities. For each phase we provide detailed activity descriptions, a detailed list of active roles and their responsibilities as well as lists of tools and techniques which support the phase. In every phase, we describe the practical implications of using foundation models, in particular pre-trained language models.

Lastly, we highlight that our methodology tackles the practical challenges which we examined in our NLU case studies. No other methodology provides guidance for all of the characteristics obtained from the case studies.

CHAPTER 6

Specialized Tools for Involvement of Domain Experts and End Users

In Chapter 5 we have presented a methodology which is tailored towards projects which use foundation models, in particular in NLU settings. We built this methodology based on representative project characteristics. We have seen that end-users and domain experts are a crucial part of data science projects and according to our guiding principles, they should be involved in almost every phase of a project. They provide valuable expert knowledge which can support data science teams in different phases of a project. In our methodology domain experts have a particularly important role during data augmentation, in particular labeling. Labeling domain specific data requires expert domain knowledge. However, it is a notoriously costly and difficult task [137, 145, 155, 157, 205] and domain experts are often not available [158]. In our case study of NLU projects we have seen that data augmentation was regarded as a key influence on the project outcome in 9 projects. In all of them, the project schedule was heavily influenced by the amount of time it took to annotate a sufficiently large dataset. Aside of labeling, domain experts can also provide valuable information for modeling, pointing data scientists to important features. Here, we face the same challenge of the availability of domain experts. Lastly, domain experts provide valuable feedback for evaluation. Outputs of machine learning models are not easy to interpret. We have seen that in many of our case studies, the availability of visualization tools improved the outcome of projects, as they served as a communication tool for data scientists and domain experts. Visualization tools help domain experts to understand the limitations and capabilities of models and enable them to interpret evaluation results.

In this chapter we introduce tools to efficiently involve end-users and domain experts and mitigate the challenge of their availability. In Section 6.1 we show how to involve domain experts and integrate their knowledge for efficient data annotation and improved modeling. Then, in Section 6.2 we will introduce visualization tools which help users to understand and evaluate model outputs. We focus on NLU settings with textual data.

6.1 Including Domain Expert Knowledge

In Chapter 3 we have seen that stakeholder involvement and in particular the involvement of domain experts is a critical success factor of data science projects. In our methodology domain experts are involved in nearly all project phases. However, a key challenge of projects is the availability of domain

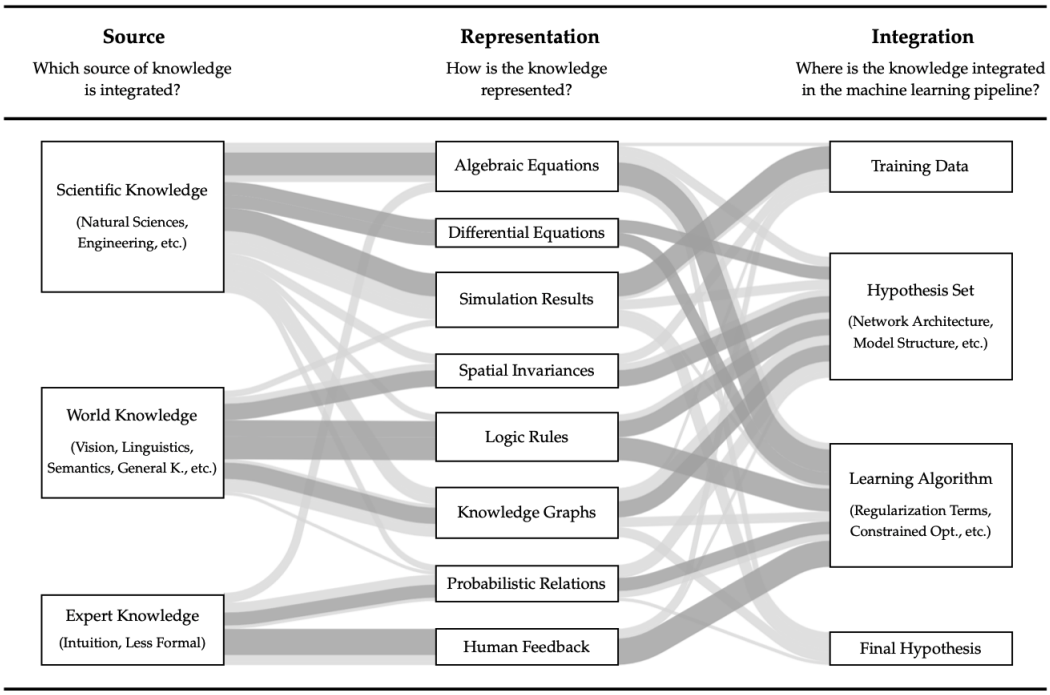

Figure 6.1: The different paths in which knowledge can be included in machine learning [21].

experts [158].

Informed machine learning is a paradigm which utilizes the integration of prior knowledge in machine learning [21]. This is particularly useful when training data is scarce. According to the informed machine learning taxonomy knowledge can be retrieved from different sources, can be represented in different forms and can be integrated in multiple parts of the machine learning pipeline. This is depicted in Figure 6.1. A popular way of using informed machine learning in NLU is to include knowledge stored in structured resources such as knowledge graphs. We are mainly interested in two paths of the graph: Integrating knowledge from structured resources such as knowledge graphs in training data, to annotate it automatically, and improving machine learning models when training data is scarce, by including expert knowledge in probabilistic relations.

6.1.1 Efficient Data Annotation

Data annotation requires a good domain understanding and hence a lot of time of domain experts. It is an important requirement for all supervised learning tasks. Fine-tuning models with such data shows clear performance benefits, even when using foundation models [72]. Typically, manual annotations by domain experts require a lot of time and effort, as for example in named entity recognition, the exact text spans of entity names have to be marked in documents and in relation extraction, entities have to be manually linked with each other with relation types. Data annotation can be made more efficient by using expert knowledge in structured form. As we have emphasized in our methodology, data hubs in NLU projects should not only contain unstructured data, but also relevant structured data such as knowledge graphs or terminologies.

Involving Domain Experts to Improve Weak Supervision

A popular approach for annotating data based on structured resources is called distant supervision [212]. In distant supervision, information stored in structured resources is matched with textual mentions to automatically annotate texts. Assume a knowledge graph with a node representing ”Barack Obama”. This node might have a directed connection to the node ”United States of America” with the name ”is_president_of”. This knowledge can now be used to annotate texts in multiple ways. Since we know that the relation ”is_president_of” requires a connection to a node of the type ”person” and a node of the type ”location”, we can now create automated annotations of entity mentions. All mentions of the string ”Barack Obama” can be annotated as entity type ”person” while all mentions of the string ”United States of America” can be annotated with the entity type ”Location”. Further, we could hypothesize that all sentences in which they occur together are representative for the relation ”is_president_of” and could assign this label to those sentences. However, this assumption is error prone. Take for example the sentence ”Barack Obama has left the United States of America for a vacation in Europe.”. Although both entities are mentioned in this sentence, the relation ”is_president_of” is not expressed in this sentence. To mitigate the effect of noisy annotations, we propose an extension of the distant supervision process [19]. In our extension we include an explainability component and a domain expert. For each relation type for which we create distantly supervised annotations, we obtain prototypical explanations, i.e., the most representative 3-grams for a relation type. In the case of the relation above, sensible 3-grams could be ”is president of”, ”was elected president” and so on. However, since we expect noise in the weak annotations, we also expect noise in the representative 3-grams, e.g., ”has left the”. At this stage we can now involve the domain experts by showing them the list of representative 3-grams we obtain after training. The experts can now select all 3-grams which are truly representative for the relation. All other 3-grams can then be used to filter noisy annotations. This can be achieved by removing the respective relation labels from all samples which contain the 3-grams. This process can be repeated multiple times. A major benefit of this denoising process is that the knowledge of domain experts is still utilized but reduced to a minimum of reading and rating a few representative explanations. We have observed two major upsides of this approach. First, it improved the results for many relation classes. Second, it allowed data scientists and domain experts to understand problems with the data and the models more quickly. For example, one observation that became quickly apparent was that the model overfitted on entity names and often predicted relations based on them and not on the actual context.

Using Seed Terms and Topic Modeling for Rapid Annotations

We propose another way of utilizing domain expertise for automated data annotations in [20]. In this case, the domain experts are involved in two subphases: In the beginning of the annotation subphase and in the end of the data exploration subphase. The goal is to label relevant documents in a large collection of documents. To this end, the domain experts are asked to create filtering criteria. These criteria can be based on content, e.g., term lists, location or time. The criteria are then used to filter the large volume dataset, to retrieve a more relevant subset of data. Given this subset, topic modeling algorithms are used to structure documents into semantic topics. Each topic is depicted with the five most relevant terms which belong to the topic, as well as the document which are most likely to contain this topic. Now, the domain experts can explore this structured topic view of the data and discover relevant topics. The documents which belong to the relevant topics can be assigned their respective

label. In our work we used this successfully to discover social media posts which are relevant for law enforcement agencies (LEAs). We obtained a list of events with time ranges, locations and terms which were of interest to the LEAs. Based on these filtering criteria we reduced the dataset to potentially relevant subsets. These were then structured into topics via the Latent Dirichlet Allocation Algorithm [55]. The LEAs helped us identify the topics which contained relevant documents. This way we reduced a dataset containing roughly one million documents to a set of a few hundred relevant documents.

6.1.2 Mitigating Error Propagation in Modeling

The typical setup in methodologies such as CRISP-DM is that a modeling phase results in the creation of a single model. In our methodology we have emphasized that it is not always viable to use a single machine learning model to solve a business problem. Problems often have to be further divided into subproblems, which might all require different modeling approaches. Often, these models are then dependent on the output of each other, which introduces common challenges such as error propagation [205]. In [23], we introduced a concept in which knowledge from domain experts was utilized to mitigate effects of error propagation. In information extraction settings for example, certain named entities are only relevant for certain document types. In the case of legal sentences, it could be that a classification model is first utilized to determine whether a convict receives a financial penalty or a prison sentence. Depending on the sentence type, either the entity type ”duration of sentence” or ”amount of penalty” is relevant and should be extracted by a named entity recognition model. However, if the classification is already wrong, the respective named entity recognition models will either make wrong or conflicting predictions. We propose to use the knowledge that only either one of the penalty types can occur, but not both simultaneously. By utilizing the certainty scores of each of the classifiers and the named entity recognition models, we can let another model learn which of the constellations is most likely to be true. We can incorporate arbitrary knowledge about the domain like this. This corresponds to the informed machine learning path with expert knowledge as source, probabilistic relations as knowledge representation and learning algorithm as integration. In this case the domain expert is involved in the modeling phase, however the decision to follow this process also effects the concept design of the target application.

6.2 Visualization Tools for End-User Feedback and Model Understanding

One of the guiding principles of our methodology is to ”embrace visualizations”, i.e. to use them frequently to support the various project phases. Visualizations benefit multiple stakeholders in a project. They help end-users, domain experts and business stakeholders to understand the possibilities of NLU methods and also to interpret results. On the other hand, they help data scientists to debug and understand model behavior and results.

6.2.1 NLU Showroom

Another guiding principle of our methodology is ”holistic thinking”, which among other ideas, calls for the reuse of software components. In our case study analysis, we have identified visualization tools

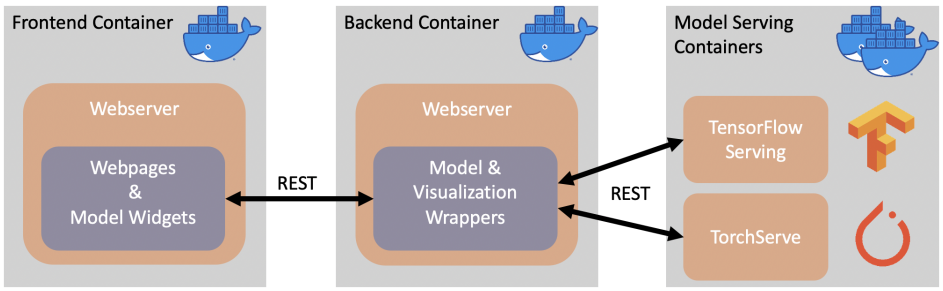

Figure 6.2: The modular architecture of the NLU Showroom [24].

as a critical success factor in five projects. We developed the NLU Showroom, a visual demonstrator which can be used across phases and projects to demonstrate the capabilities of various NLU methods and visualize their outputs [24].

Modular Architecture

The NLU Showroom is built with a modular architecture which consists of a frontend, backend and model hub. The frontend contains web-based widgets for processing user inputs and displaying various NLU outputs, such as word similarities, sentence similarities, entity mentions, relations and text generation. However, it can easily be extended to further tasks. The backend architecture communicates with the model hub and the frontend. First, it takes user requests from the frontend and translates them to model inputs. Then it takes the model response and translates it into a format that can be digested and visualized by the widgets. The model hub stores the models for the different tasks and serves them. To this end TensorFlow Serving and TorchServe are used. The architecture of the showroom makes it easy to integrate novel tasks and models. If adequate widgets are already available, then the models can be included and assigned to a certain widget. If novel widgets are included, only minor changes have to be made to the front- and backend. This makes it easy to use the showroom across projects, since the existing widgets can easily be reused whenever the same tasks are addressed in projects.

Improve Stakeholder Understanding

A key purpose of the NLU Showroom is to showcase the capabilities and limitations of current NLU models. By introducing a Mutual Understanding phase in our methodology, we highlight the importance of not only data scientists understanding the business domain, but also business stakeholders, end users, and domain experts grasping the technical domain. This is particularly important for expectation management. The NLU Showroom offers stakeholders an easy-to-use user interface for feeding their own custom inputs to models. It also offers model and use case descriptions as well as insights into which datasets were used to train the underlying models. This can be helpful during various phases of an NLU project. During the Acquisition Subphase, business stakeholders can explore the capabilities of NLU models even before an actual project starts. This can increase the trust in the capabilities of the NLU methods and increase the likelihood of reaching a contractual agreement. During the Modeling phase, the showroom can be used to deploy models for user and domain expert testing. They can experiment with intermediate models and give the data scientists feedback easily.

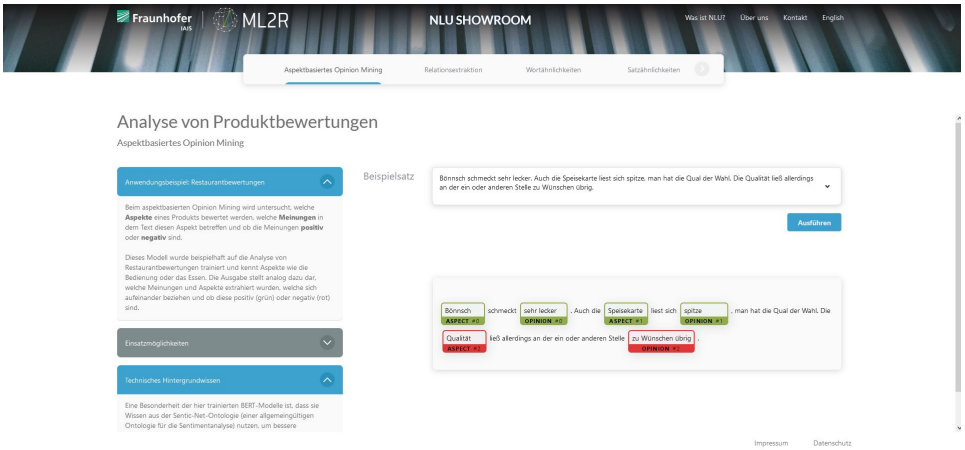

Figure 6.3: The user interface of the NLU Showroom, depicting the aspect-based opinion mining demo [24].

Improving the Data Scientist's Understanding

Visualizing model outputs is not only beneficial for client stakeholders but also for data scientists. The NLU Showroom can be used by data scientists in the same way as client stakeholders could use it. Data scientists can deploy their own models in the model hub and enter inputs to verify the model outputs. We used this to understand the model behavior in an aspect-based opinion detection setting. In this setting we had designed a solution which would detect aspects of a review such as "service", the corresponding opinions such as "friendly" and link them together. We found that the model was very resilient to spelling mistakes, negations and handling multiple aspects and opinions at once. However, by testing with custom inputs, we soon found that the model did not handle cases well in which multiple opinions were linked to the same aspect. This was very insightful and highlighted a shortcoming of the training and test sets we used, as they did not contain such cases.

Discussion

The NLU Showroom gives data scientists an easy to extend tool to integrate NLU models and evaluate them qualitatively. Through its emphasis on reusability, this allows data scientists to deliver models to end-users and domain experts rapidly in projects. They can use the NLU showroom to assess the outputs of models, given arbitrary inputs. Further, the showroom provides them with context regarding models, usages and datasets. This is not only useful during the modeling phase of a project, but already during the acquisition subphase, as it gives client stakeholders a good overview of capabilities and limitations of NLU models even before a project starts. This can be helpful for managing expectations and lowering the perceived risk that a client takes when starting a project.

6.2.2 Geospatial Topic Demonstrator

Often, data in NLU projects contains valuable meta-data such as geolocations. This meta-data is then further enriched throughout a project, e.g., by using NLU models such as sentiment detection models. In [25] we developed a demonstrator which can be used to explore textual data with geolocation meta-data visually. Our goal was to give end-users and domain experts a rapid overview over topics of

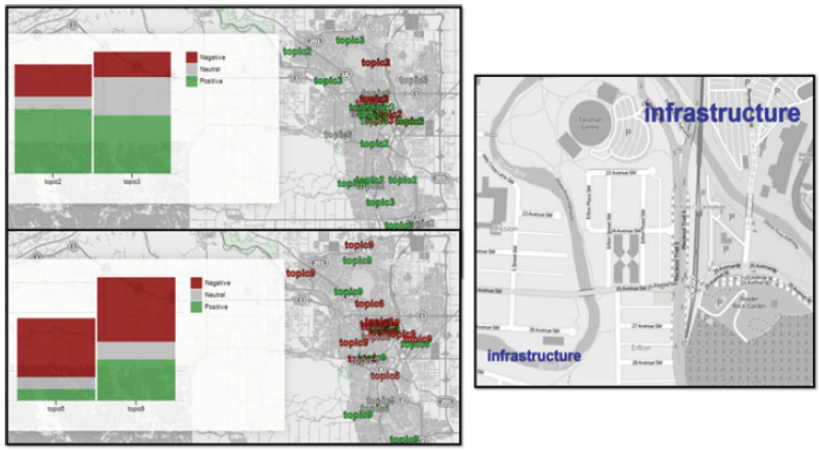

Figure 6.4: The user interface of the geospatial demonstrator [25].

interest. With our demonstrator the respective users were enabled to quickly discover points of interests on maps, by analyzing topic names and sentiment scores. We evaluated the tool in a community policing and emergency response setting.

Architecture and Functionality

The tool consists of a web-based frontend component, a backend and a model and data hub. The frontend visualizes documents on a 2-dimensional map, representing them with the topic label of the topic they were assigned to. In addition, it color-codes the topics based on the sentiment of the underlying document. Further, it provides aggregation views for an area of interest, to see the topic and sentiment distributions over the area. Users can enter relevant terms in the frontend, which trigger a keyword search and a topic model training. The keyword search, the topic modeling and the sentiment score assignment are handled by the backend. Finally, newly retrieved data is stored in the data hub and the topic and sentiment detection models are stored in the model hub. When using the tool, users specify terms of interest, which are then used to derive relevant topics and locate them on a map, together with the associated sentiment scores. The topics can be named by the user for easier and faster analysis. The model and data hubs are designed such that arbitrary topic modeling algorithms can be used, as the topics are stored in a model-agnostic way. This eases the reuse of the demonstrator over time.

Guiding End-Users

We found that the tool helps guide end-users to hot-spots, i.e. locations which contain many texts with topics of interest. In the case of the emergency response setting, the tool helped end-users identify locations which suffered infrastructure damage. This way, issues with power-supply, bridge closures and flooded areas were discovered, which proved to be a drastic reduction of effort in analyzing more than 10,000 Tweets.

Discussion

The Geospatial Demonstrator offers intuitive visualizations, which help end-users focus on relevant data, even in settings where there is a lot of noise, such as social media analysis.

6.3 Conclusion

In this chapter we have focused on the involvement of domain experts and end-users in different phases of our methodology. Our goal was to make more efficient use of their time while integrating their knowledge in the creation of NLU solutions and to make it easier for them to digest the outputs of NLU applications. We have introduced processes for automating annotations via integration of expert knowledge, while keeping the work of the domain experts to a minimum. In a weak supervision process, we utilized domain knowledge in the form of structured data and involved domain experts by letting them select data points which corresponded to unreasonable model explanations. We also introduced a way of using expert terminologies and topic modeling for reducing high volumes of data to relevant parts for domain experts. The work of the domain experts was limited to providing relevant terms and reviewing topic model results.

With the NLU Showroom we introduced a visualization tool which can support the Mutual Understanding, Modeling and Evaluation phases of projects. It eases the process for data scientists of delivering models to end-users and domain experts, so that they can qualitatively evaluate and explore them. It is focused on reusability so that it can be easily reused across projects and project phases and requires no adaptation when dealing with already integrated tasks.

The Geospatial Topic Demonstrator utilizes meta-data other than text, to depict model results on a 2-d map and with sentiment color-coding. The demonstrator is built in a way that is topic model agnostic and can be reused. It also provides end-users with the possibility to input relevant terms for guiding their search through large amounts of data.

CHAPTER 7

Conclusion

In this thesis we have derived the first foundation model-centric data science project methodology, which is tailored towards the specific characteristics of foundation model projects and the needs and expectation of modern enterprises. We took a unique approach of combining broad theoretical knowledge with practical experience. To this end we combined the results of two literature studies with the practical insights from 26 case studies.

Foundation models have substantially changed the way how companies execute data science projects. The reusability and adaptability of foundation models fundamentally changes the data science process, in particular the modeling phase of projects. Whereas modeling in former projects required exhaustive data preprocessing, feature engineering and modeling, modeling in modern foundation model-based projects can be as simple as prompting existing models. We argued that this is a paradigm shift which requires rethinking the execution of data science projects.

In Section 3.1.2 we investigated the development and evolution of data science project methodologies to understand what drove their development. The sheer number of methodologies indicates a need for the continuous development and evolution of them. We discovered that methodologies were often tailored and adapted towards changing project characteristics, i.e., requirements, challenges and success factors. We analyzed 27 data science methodologies and discovered that over time, their development aligned with paradigm shifts in data science. We proposed to group the methodologies in five major groups. The ”KDD and Data Mining Group”, which recognized the need for specialized data science methodologies, proposing a shift from pure software engineering methodologies. The ”Big Data Group”, which is tailored towards the ”Big Data V” challenges. The ”Agile Group” which identified the need for agile methods in data science, due to larger and more complex projects, facing the same challenges as software development projects. The ”Integrated Framework Group” which bundled software and infrastructure with methodologies and lastly, the ”Software Development and Operations Group”, which acknowledges the trend towards building data science applications and the requirement of operating and maintaining them. We argue that foundation models represent a novel paradigm shift that should be accounted for in data science methodologies. Since NLU is arguably the subfield of data science with the largest impact of foundation models, we included methodologies which are tailored towards NLU projects in a sixth group, called ”NLU Group”. However, we found that these methodologies not even mention foundation models or pre-trained language models.

To understand the characteristics of modern-day data science projects and the impact of foundation models, we studied literature on data science project management. In our literature review, we

identified numerous studies that examined the characteristics of data science projects, specifically focusing on their success factors, challenges, and requirements. As a result of this literature study, we obtained a list of relevant characteristics of general data science projects. These characteristics provide valuable insights, e.g., on team management and the general data science process.

However, because foundation models are relatively recent, their influence on data science projects is not extensively addressed by the literature and reflected in the characteristics. Therefore, we performed a second literature study in which we analyzed the foundation model literature. The second literature study complements the results of our first literature study by extending our list of characteristics with foundation model specific project characteristics, such as the adaptability of models, choosing between pre-training and adaptation of models and so on.

We deduplicated and aggregated the results of both literature studies and obtained a catalog of 163 aggregated groups of project characteristics. We organized the characteristics in a tree-like structure. The characteristics represent the requirements of a modern data science project methodologies for foundation model-based projects. Our catalog represents the most comprehensive collection of such characteristics. Previous literature studies only cover parts of our catalog and completely ignore foundation model characteristics. The top level of the catalog is represented by the four categories ”Process”, ”Technology and Infrastructure”, ”Artifact/Asset Management” and ”Management/Governance”. The lower-level categories contain more specific characteristics such as ”model reusability” and ”pre-training vs. adaptation”.

We assume that not all of the characteristics equally influence data science projects. Hence, we studied 26 projects of an NLU Team over the course of 5 years. 19 of the case studies represent projects in which we were directly involved. We analyzed the other 7 case studies by interviewing the respective project leads of the projects. We analyze the importance of each characteristic with respect to the project execution and outcome. Based on the results of our analysis, we obtained a subset of 45 characteristics, which were particularly important for the NLU projects. These project-relevant characteristics include foundation model specific characteristics, such as the reuse of existing models, but also general data science characteristics, such as the ”mutual understanding” between customers and data science teams. The results of the case study analysis support our approach of combining general data science characteristics with characteristics specific to foundation model usage. A limitation of our study is its focus on the projects of a single team. This represents a bias towards the characteristics of the team and their projects. Performing this study with other teams and projects could have led to another selection of relevant characteristics. Therefore, we argue that both, our extensive catalog and the project-relevant subset of it, are important representations of project characteristics. Especially teams other than the one that we studied will benefit from the broader catalog and can extract different subsets of it.

To validate our hypothesis from Chapter 3, that no existing methodology is tailored towards the usage of foundation models, we analyzed 27 methodologies with regards to their coverage of both, the complete catalog of characteristics and the project-relevant characteristics. Based on our complete catalog and the project-relevant characteristics, we assessed 27 data science project methodologies. For each methodology we analyzed whether it mentions or directly addresses the characteristics. We discover similarities among methodologies belonging to the same methodology groups with respect to the characteristics they cover. While, e.g., integrated methodologies perform strongly with respect to characteristics concerning the Infrastructure and Technology category, agile methodologies put an emphasis on team organization and stakeholder involvement. This showed that our proposed grouping of methodologies is reflected in the characteristics that the methodologies address. In a gap analysis,

we validate that none of the existing methodologies cover either the complete catalog of characteristics or the project-relevant characteristics. In particular, we discover that the methodologies all neglect foundation model-based characteristics and characteristics concerning the development of applications. This proves that a novel methodology, tailored towards modern business requirements and foundation model usage, is indeed required. In contrast to former methodologies, which consider data as the central artifact in data science projects, it should be centered around the development and usage of foundation models.

Based on the best practices we derived from previous methodologies and our catalog of characteristics, we derive the first foundation model-centric data science project methodology in Chapter 5. The central artifact in our methodology are foundation models and their properties. These models have a strong impact on all data science phases. Our methodology is built on four guiding principles, which reflect best practices from former methodologies and extend them with foundation model specific practices. We emphasize the need for holistic thinking, suggest involving end users and domain expert in every phase of a project, recommend to deliver and verify results frequently and propose to embrace visualizations throughout a project. Foundation models represent reusable assets, which are not bound to a single project. However, other artifacts such as software, data and knowledge and information should be reused along them. This leads to the decision to incorporate artifact hubs for models, data, knowledge and information and software as a central component of our methodology. These hubs can be external and internal and should be maintained across projects. Our methodology consists of 9 phases and 34 subphases with detailed activity descriptions, role descriptions, artifact descriptions and lists of tools and techniques to support the process. The high-level phases largely resemble the phases of the accomplished CRISP-DM methodology. However, we add phases for application development and testing, as well the maintenance and operation of models and applications. Further, we extend the Business Understanding phase and rename it to Mutual Understanding. We argue that it is equally important to understand a customer's business, as it is to educate the customer about technical possibilities and limitations. Lastly, we argue that a project should be embedded in the strategic process of a team. Our methodology is the only methodology which covers all characteristics from our catalog by design. It is also the only methodology to cover all project-relevant characteristics. It reflects the practical implications of using foundation models and better reflects the necessity of embedding projects into overarching processes and business strategies. Further, it acknowledges that data science projects have changed their scope, from exploratory projects to projects with the requirement of building applications.

In Chapter 6 we extend upon the tools which we integrated in our methodology. A central aspect of our methodology is the involvement of domain experts and end users. However, their time is scarce. To mitigate this problem, we introduced specialized tools, to make efficient use of their time. We developed two tools which utilized structured data sources for automated data annotation. In these tools expert knowledge is stored in the form of knowledge graphs and terminologies and then mapped into texts for annotations. In one of the tools experts only have to review model explanations, to filter noisy annotations. In the other tool, domain experts are provided with an aggregated view of the weakly annotated data and can provide their feedback. We further introduced a tool to mitigate the problem of error propagation in systems which include multiple models, that are dependent on one another. In this tool domain experts are only involved in creating a set of rules to express the dependencies between the different models. Finally, we introduced two visualization tools, which ease the understanding of model outputs for end users and domain experts and help data scientists debug their models.

In conclusion we developed the first foundation model-centric methodology which reflects the general characteristics of modern data science projects as well as the specialized characteristics of projects using foundation models. We took an evidence-based approach to developing the methodology by first assessing how methodologies were adapted over time, then deriving the requirements of a foundation model-centric methodology and finally developing it based on best practices from former methodologies and novel practices. Our methodology is the only methodology to account for all requirements which we derived.

7.1 Limitations and Future Work

We want to suggest multiple ways of extending the work in this thesis. Our methodology is grounded on the theoretical background of 27 methodologies, 163 project characteristics and 26 NLU case studies. Through tailoring the methodology towards project-relevant characteristics, we showed its practical relevance. However, we are missing an ex-post evaluation, which also considers the overhead of using our methodology. The ex-post evaluation of methodologies, e.g., by comparing methodologies via controlled experiments, is difficult and has only been done in laboratory settings [18]. However, the experience of working with our methodology should be evaluated by testing it in projects. We propose to follow the example of Dutta et al. [113], who evaluated their methodology based on a case study and reported the results of their project as well as the feedback obtained by interviewing project stakeholders.

One limitation of our case study analysis is that we followed a single team. We suggest broadening the scope of the case studies and to include more examples from a wider variety of teams. We believe that this can yield better insights into relevance of the different project characteristics in practical scenarios.

Our methodology assumes that teams have the capability of training their own models. However, due to the easiness of using publicly hosted generative large language models with their increasing capabilities, teams that have no experience in machine learning will start building applications based on foundation models. To reduce the overhead of applying our methodology for such teams, we propose to modify our methodology such that there is a focus on using publicly hosted foundation models.

In our last chapter, we introduced tools and processes to support the execution of data science projects by automating tedious tasks for domain experts and end users. Large language model-based agents have proven to be able to support and automate sophisticated tasks, such as autonomously developing mobile games [253]. We wonder to what degree project management might be automated due to large language model agents in the future. Tools such as the toolformer [254], langchain [255] and AutoGPT [256] could be combined to support project management processes. Recently, Meta-GPT was applied to automate software development processes by a single line of requirements, emulating multiple roles in a software development company [257]. We believe that it might be possible to develop agents which take on project management activities or even execute complete data science projects autonomously in the future.

APPENDIX A

Appendix

A.1 Data Science Methodologies

Here, we give a detailed description of the methodologies, which we presented and assessed in Chapter 4.

A.1.1 Knowledge Discovery in Databases

The Knowledge Discovery in Databases (KDD) process was introduced in [29] by Matheus et al. and further elaborated on in [30] by Fayyad et al. with the goal of structuring the process of projects for knowledge discovery, with data mining as one of their core activities. The process explicitly calls for a human-centered approach to knowledge discovery, meaning that the process revolves around a user which applies the process to extract knowledge from databases. This user is also referred to as data analyst or data archaeologist.

The biggest challenge described in KDD is the ever-growing amount of data that needs to be analyzed by data miners. The huge amounts of rows (instances) and columns (features) in typical knowledge discovery databases make a manual inspection of the data slow, expensive and biased. The authors call for an automation of the knowledge discovery task to scale up the human analysis capabilities. Due to the complex and error-prone nature of the knowledge discovery task, especially in ”real-world” projects, the authors argue for the need of a standardized process which guides the user and describes complex phases of interactions between the user and a possibly large database. They came to this conclusion after observing data mining projects and interviewing data miners. The main takeaway of their observations was that it was not the modeling aspects of projects that took the most time but rather supporting areas. The data miners struggled with tasks such as keeping track of and organizing information, system integration, data conversion, missing data and so on. Based on the tasks that the data miners performed and the challenges they faced, the KDD process was first suggested in [29].

KDD is a project methodology with an according project life cycle. It models knowledge discovery projects with various phases and addresses the topics of data storage and access, scaling of data mining algorithms to large datasets, interpretation and visualization of results and human-machine interaction and at least indicates how to perform the phases and which roles should be involved.

Life cycle, processes and tasks

The KDD process consists of 9 different phases which range from domain understanding to actions derived by the knowledge discovered in the knowledge discovery process. An overview of the phases can be seen in figure A.1. In the original publication, they refer to these phases as phases.

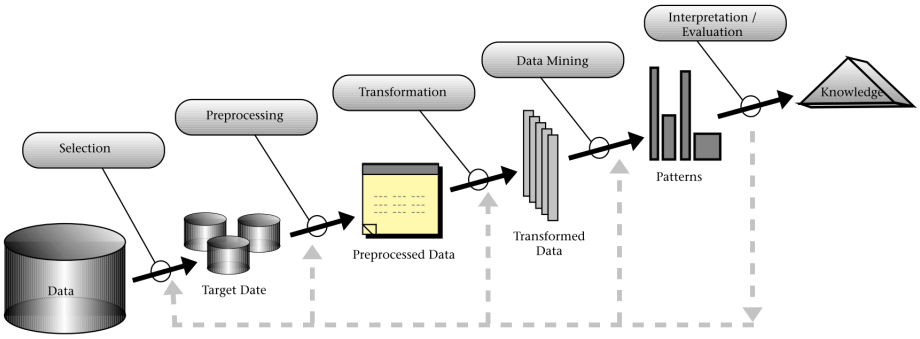

Figure A.1: The phases of KDD and the according lifecycle [30].

Phase 1: Domain Understanding In the first phase an understanding of the application domain is built as well as an initial understanding of the data. Further, prior domain knowledge from the customer is obtained and the goals are formulated from the customer's viewpoint. Often a hypothesis about the data is formulated which the customer aims to verify through the analysis.

Phase 2: Data Selection In the next phase a target dataset is created based on the information obtained in the prior phase. The data is reduced to the subset of data which is relevant to the analysis. In more exploratory projects the selection of the subsets and the first insights gained from them can lead to the formulation of hypotheses. At this phase domain expert knowledge is crucial.

Phase 3: Data Cleaning and Preprocessing After the initial selection of data, the data is cleaned e.g. by removing noise and further preprocessed e.g. by filtering irrelevant or duplicate entries and handling missing data. The authors hint at the possibility that irregularities might be caused by flawed data handling processes which need to be adjusted for future use cases. Another observation they make is that sometimes the data that seems irrelevant or faulty, is actually the interesting data containing relevant information for the customer.

Phase 4: Data reduction and projection Based on the formulated goals and hypothesis the preprocessed dataset is now further filtered. Only relevant features are kept.

Phase 5: Model to goal alignment In this phase goals of the process are aligned with categories of data mining methods such as classification, regression, etc. The best-fitting categories are selected for the next phases.

Phase 6: Exploratory analysis and model selection Depending on the goal of the process this phase either aims at picking the right data mining algorithms for discovering insights from data, building predictive capabilities or both. Further parameter ranges for the algorithms are selected.

Phase 7: Data mining This is the central step of the process. The user applies the algorithms to the data to find patterns or suitable data representations.

Phase 8: Result interpretation and evaluation The results of the data mining phase are evaluated

in this phase. If helpful, the results, model and data should be visualized at this phase to communicate and analysis the results better. This phase might lead to iterations of all previous phases.

Phase 9: Acting on the discovered knowledge Finally, in the last phase, the user makes use of the knowledge gained by the process depending on the goals of the customer. Examples of actions could be a final report, direct use of the knowledge or the integration of the knowledge in another system.

The authors state that the KDD process can have loops between any of the two phases and significant iterations.

Roles and responsibilities

The KDD methodology mentions three different roles, which are involved in a project. The user, also referred to as data analyst or archaeologist, is responsible for applying the KDD process. The domain expert supports the user by providing crucial background knowledge for hypothesis building and the business-person or customer, who provides information about the problem to be solved, i.e. they provide requirements and the business question and review results.

Artifacts

The KDD process does not specify any specific artifacts to be generated during the process. It mentions data as the central element of the whole KDD process and a data dictionary in order to understand the data, as well as databases to store the data in.

Tools and techniques

Tools mentioned to accompany the process are S, SAS and various machine learning algorithms such as decision trees, neural networks, etc.

A.1.2 CRISP-DM

The Cross Industry Standard Process for Data Mining (CRISP-DM) [2] was introduced in 2000 and succeeded the KDD process. Recent polls show that even 20 years after its introduction, it is still considered the most widely adopted data science methodology [100]. CRISP-DM structures data mining projects into four levels: Phases, generic tasks, specialized tasks and process instances. The first level, the phases, provide a generic view on data mining projects. They are divided into business understanding, data understanding, data preparation, modeling, evaluation, and deployment. All phases are centered around the available data. Each phase is comprised of generic tasks, which should be independent of the application and models used, and specialized tasks, which describe how actions should be carried out in specific situations. Lastly in the process instance level, particular instances are described based on the tasks defined on the generic and specific level.

The goal of the original CRISP-DM paper was to establish a standard for data mining projects, to transition the field of data mining into a more mature state [2]. To this end, the authors present a rather generic methodology and emphasize the need for specialization to specific domains and projects. The description of the phases and the generic tasks in [2] is rather high-level with a brief mention of outputs of all tasks. This is enriched with a user guide and more detailed descriptions of artifacts and tasks in [102]. The phases of CRISP-DM are embedded in a project life cycle which highlights the most common phase transitions.

Figure A.2: The Generic Process CRISP-DM cycle, showing the highest level with its 6 phases. [2]

Life Cycle, Tasks and Processes

The CRISP-DM phases, displayed in figure A.2, follow a cyclic nature. The outcome of each of the phases defines which phase should be visited next. According to CRISP-DM, typically more than one iteration through the cycle is performed in a project. The authors constitute this with the not straightforward nature of data mining projects. They further emphasize that transitions between phases are project-dependent and do not always follow a fixed order. They add that phase transitions other than the ones specified in their CRISP-DM cycle (see figure A.2) can occur. In the following subsections we will shortly summarize the six phases of CRISP-DM:

Phase 1: Business Understanding The authors suggest several generic tasks in the business understanding phase. One of the goals of this phase is to understand the background of the project, the business objectives as well as the business success criteria. The project situation is assessed to derive the available resources, project requirements, underlying assumptions and constraints. Risks and contingencies are determined, a project terminology is created and costs and benefits are listed. With the now available knowledge about the is-state and the business objectives, data mining goals are determined along with data mining success criteria. Given all of these information a first project plan is produced with an initial assessment of the tools and techniques that should be used.

Phase 2: Data Understanding According to the authors the data understanding phase is tightly coupled to the business understanding phase. This intuitively makes sense, as an understanding of the business objective will help to determine the correct data, while an understanding of the data should lead to a better project plan. This phase starts with an initial collection of data, which amounts in an initial data collection report. The collected data is then described in a data description report. Next, the data is explored and the data quality is evaluated. This leads to a data exploration and data quality report.

Phase 3: Data Preparation The third phase proposed in CRISP-DM is the data preparation phase.

Its ultimate goal is the creation of the final data set and its description. The phase begins with the selection of the data via a rationale for inclusion and exclusion. The selected data is then cleaned and the cleaning steps are reported in a data cleaning report. Next is the construct data task, in which attributes are derived and new records are generated. Finally, data from different sources is integrated and reformatted to assure compatibility with the modeling tools. [103] further describe a data reduction step in which irrelevant records are removed.

Phase 4: Modeling The modeling phase starts with the selection of modeling techniques based on modeling assumptions. Next, tests are designed to assess the quality of the model. The main objective in this phase is to build and assess the model. Building the model requires picking appropriate parameters and should not only amount in the models but also in a description of them. During the model assessment it might be necessary to revise the parameter settings. The authors note that models and their assessment often lead to the discovery of problems with the data, which might amount to a transition back to the data preparation phase and the construction of new data.

Phase 5: Evaluation While the model quality has been assessed in the prior phase, in this phase the models are assessed as to how well they meet the business objectives. To achieve this, the results of the data mining models are compared to the success criteria. If the models meet the success criteria, they are approved. The whole process of the project is reviewed and the next steps are determined, yielding a list of possible actions. Finally, a decision is made on how to proceed.

Phase 6: Deployment The last project phase is the deployment phase. Initially the deployment is planned. If the project requires running the data mining solution in a productive environment, monitoring and maintenance of the deployed solutions is planned. Depending on the needs of the user, deployment can range from a final report to software integration. The authors claim that the deployment is often performed by the user, but knowledge about the deployment nature is needed upfront. Lastly, the project is reviewed and the experience of everyone is documented.

Roles and responsibilities

CRISP-DM mentions 16 roles along the project life cycle. There are no explicit descriptions of all of the roles and their responsibilities and only for some of them the tasks are mentioned in the phase descriptions. The following roles are mentioned: Customers, Data Analysts, Business Experts, Internal Sponsors, Data Experts, Technical Support, Data Mining Personnel, Data Mining Engineers, Business Analysts, System Administrators, Database Administrators, Market Analysts, Statisticians, Domain Experts, Project Leaders and End Users.

Artifacts

CRISP-DM provides extensive descriptions of artifacts which are generated throughout a project lifecycle. They suggest 41 artifacts which are grouped according to the phases they are produced/used in. For easier readability we offload the detailed listing and descriptions of the artifacts to the appendix and just provide a general overview here.

- **Business Understanding**: The business understanding phase contains 12 different artifacts, which provide important project information for both the client as well as the data mining team. The artifacts in this phase are documents which mainly describe the goal and background of the project, the available resources for the project, project constraints and the project plan.

- **Data Understanding**: This phase contains four reports which describe how data is obtained, where it is stored, provides information about the data itself, interesting findings in the data and a summary of its quality.
- **Data Preparation**: Aside of the documents describing the data preparation methods and decisions, this phase also contains the dataset itself and its modified versions as artifacts. In total it contains seven different artifacts.
- **Data Modeling**: The data modeling phase contains eight artifacts which range from descriptions of modeling choices to the models and parameter choices themselves and result assessments.
- **Evaluation**: In the evaluation phase CRISP-DM introduces five artifacts. These artifacts contain the evaluation results, the approved models, a review of the whole data mining process, lists of possible next actions and a documented decision on how to proceed.
- **Deployment**: The deployment phase also contains five artifacts. These artifacts describe how the results of the project are deployed, they include a plan for maintenance and monitoring of the result, final reports and presentations as well as an experience documentation for knowledge dissemination.

Tools and Techniques

CRISP-DM does not provide a list of tools and techniques.

A.1.3 RAMSYS

The RApid collaborative data Mining SYStem (RAMSYS) methodology was introduced by Moyle and Jorge in [111] as an adaptation of the CRISP-DM methodology which focuses on projects in a remote collaboration setting. The authors argue that a lot of competence in the data mining space is scattered around different companies and research institutes which occasionally collaborate in projects. In such settings onside collaboration is difficult, so RAMSYS introduces artifacts and guiding principles which optimize the joint work of remote partners.

RAMSYS introduces the concept of a network of expertise in which nodes represent data mining units such as experts, expert teams or technical support teams. All these nodes together form a team. Some nodes form the management committee which has special obligations such as managing the client interface, defining project evaluation criteria and performing the solution selection.

The authors of RAMSYS share the principles upon which they built the methodology. It is built with them in mind to ease and optimize remote collaboration [111]:

1. **Light management**: The management committee is responsible for the flow of information and ensuring the quality of the solution, not for directly controlling the work of each team. To ensure the independent work of the different nodes, the problem definition and objectives should be clear from the beginning and shared with everyone.
2. **Start any time**: Any information available, that is necessary to start solving a data mining problem should be available to everyone at any time, i.e. information such as the problem definition, information about the data and the data, evaluation criteria and problem specific knowledge

3. **Stop at any time**: This calls for a simplicity first approach, preferring simple baselines as a starting point and then successively comparing them with more complicated solutions. This way results can be produced early on and in a procedural manner. Whenever the management committee issues a stopping signal the intermediate results are available for review and presentation.

4. **Problem Solving Freedom**: Along with light management this emphasizes that the management committee should not prescribe solutions but rather suggest them and that each team has the freedom to explore their own ideas, relying on their expertise.

5. **Knowledge Sharing**: All experiments and results by each modeler should be shared immediately with everybody else on the project.

6. **Security**: Despite the lightly managed network structure, privacy and data security need to be ensured at all times, especially due to the decentralized setting.

7. **Better Solutions**: Solutions by individual contributors should not only be considered as separately but also in a joint matter, as this could lead to an even better solution.

Because the life cycle adjustments made in RAMSYS built upon special artifacts which the methodology introduces, we shift the order of presentation of the components of RAMSYS and start with its artifacts.

Artifacts

In addition to the guiding principles, RAMSYS introduces artifacts for the sharing and storing of information, hypothesis and results. One of the most central parts of RAMSYS is an information management setup. RAMSYS calls for a process in which all project relevant information are shared across all nodes. Central questions to the data mining problem solving process are which information to collect, develop and maintain and how and where they should be kept. RAMSYS defines **problem information** as *the best current understanding of the problem*. This includes the problem definition, information about the data and the data itself, hypotheses about the data and the models and the current validity status of all hypotheses. To manage and store these information an **information vault** is suggested.

The information stored in the information vault are the following:

1. **Problem definition**: the business understanding and data mining view of the problem

2. **Distilled Knowledge from Related problems**: similar problems addressed in the past, applied solutions, literature and produced knowledge

3. **Evaluation criteria definition**: How should the solution be evaluated?

4. **Data**: The collected database, all meta-data, data quality assessments, derived data and information on the steps how to obtain the derived data from the original database

5. **The Hypothesis Investment Account**: All hypothesis and the problem specific knowledge derived during problem solving

The **Hypothesis Investment Account** (HIA) plays a crucial role in managing and keeping track of experiments. RAMSYS defines hypotheses as unproven statements about the problem or the problem solution. The assumption is that some of the hypotheses will be proven to be facts. Hypotheses can be suggested by any of the team members. They should always be stated clearly and shared with everyone on the project. The HIA contains all hypothesis threads for a project. It can be used to track them, similarly as in bug tracking but also for future projects as a lesson learnt. The following list provides an overview of all the information stored in the HIA:

1. **Hypothesis statement**: A space to list all hypothesis, propose ideas, argue whether and why hypothesis are valid and suggest experiments that might confirm them.
2. **Refutation**: A space where team members can argue that proposed hypothesis are false and provide experimental or theoretical results that support this.
3. **Corroboration**: A space where team members can provide evidence that supports the validity of an hypothesis.
4. **Proof**: A space to share proof that a hypothesis is true.
5. **Refinement**: A space to refine and specialize hypotheses
6. **Generalization**: A space to provide a generalized hypothesis.

In contrast to the centralized data management, modeling is decentralized but collaboration in different forms is encouraged. Different nodes can collaborate by cooperating, competing in challenges, work independently or in blended forms such as working independently but exchanging ideas.

Life cycle, tasks and processes:

RAMSYS builds on the same phases and process as CRISP-DM. However, the authors suggest that information generated during the project are stored in the information vault. Every time a hypothesis is generated it should be documented in the HIA. RAMSYS adds the model submission task to the modelling phase. In this task the modelers submit their best models for review and selection. The authors note that this step is only useful when the different modelers work in a competitive manner in which their models are evaluated against each other.

Roles and responsibilities:

The authors suggests that all data is stored on one node of the project network and is managed by a dedicated data master. This data master maintains the database and applies all suggested transformations. Transformations are stored in the HIA together with their expected effect. All transformations are subject to debates and need agreement by the team members. Only additive transformations are allowed to ensure compatibility with all prior analysis. All transformed variants of the database are stored in the information vault along with the instruction of how to transform the data. All transformations are also applied to the evaluation set to reduce effort. Even though the mastering of data is centralized, all nodes should always have access to the data and its transformations, while ensuring security and privacy restrictions. Similarly, information about the problem definition, evaluation criteria and problem specific knowledge should shared at all times and be clearly formulated.

In addition, RAMSYS includes a Management Committee. The management committee performs evaluation tests, issues hold of modeling, manages interface with client, schedules delivery and sets up challenges between data science teams.

Tools and techniques

The authors of RAMSYS additionally introduce tools to support the methodology. They provide a data master support tool, a HIA and modeler communication tool and a model submission and model evaluation support tool.

A.1.4 A Process Model for Data Mining Engineering

In 2009 Marbán et al. recognized a trend toward more and more applied data mining projects [86]. However, their observation was, that these projects are rather chaotic compared to software engineering projects, due to the immaturity of processes in them. Software engineering processes, methodologies and life cycles matured over the course of a couple decades. The authors argue that there is a need for data mining engineering, to effectively integrate, reuse and interchange results of data mining projects. While CRISP-DM has been an important milestone in the execution of data mining projects, it mostly disregards organizational, managerial and other activities which are not directly related to development. However, the authors highlight that there are enough similarities between data mining projects and software engineering projects to borrow process models from the software engineering domain and to account for the shortcomings of CRISP-DM.

Standard processes in software engineering

To develop a process model for data mining engineering, the authors analyze two standard processes in software engineering, the IEEE STD1074 process and the ISO 12207 process. They create a joint process model which adopts processes and activities from both.

The joint process model consists of 6 process groups: acquisition, supply, software life cycle selection, project management processes, development-oriented processes and integral processes. The acquisition processes are from the view of a company that outsources activities to another company and start with the proposal and end with the acceptance of the product. The supply processes take the opposite view and describe processes from the point of view of the company which performs the data mining tasks for the outsourcing company. This includes interaction management tasks. The software life cycle processes contain the process of identifying available software life cycles and selecting an appropriate one for the project. In the project management processes, the project is initiated, project monitoring and control is performed and project planning tasks are executed. With these processes the project structure is established and the project is coordinated and managed throughout the whole project lifecycle. The development-oriented processes are further subdivided into pre-development, development and post-development processes. The pre-development processes are concept exploration, system allocation and software importation. Development processes are requirements, design and implementation. After the development, installation, operation and support, maintenance and retirement form the post-development processes. Integral processes are processes which assure the quality and completeness of project deliveries. They encompass evaluation, software configuration management, documentation management and training.

CRISP-DM vs. the joint software engineering process

Since CRISP-DM is the most popular methodology in data mining, the authors compare it to the joint software engineering process that they formulated. They analyze which parts are already covered by CRISP-DM, which parts are applicable but not covered and which parts need modification.

According to them acquisition and supply processes are equally as important for data mining as for software engineering, but are not at all covered by CRISP-DM. The selection of life cycles is also very important and depends on the complexity of the project, the experience of the team in the problem domain, knowledge of the data, variability and data expiration.

Project management processes are only partially accounted for in CRISP-DM. Only the business understanding phase covers some of the necessary management tasks. The authors argue that this is rather critical, since data mining projects are often high risk and require good planning, including contingency plans. Project costs, benefits and ROI should be accounted for in the planning. While CRISP-DM covers the planning of required resources, it does not account for the allocation. One of the biggest shortcomings of CRISP-DM is the insufficient consideration of metrics. Only data mining results are evaluated, the project is not sufficiently evaluated in terms of budget, time and business value generated. Configuration management is completely omitted in CRISP-DM, although this appears to be a crucial process for data mining. It refers to the versioning, changing and modification of elements such as data, models and documentation. While CRISP-DM includes documentation tasks, it does not include tasks for the planning of documentation.

The development-oriented processes are the ones best covered by CRISP-DM. Most data mining methodologies focus primarily on this phase. During pre-development CRISP-DM does not cover the concept exploration, system allocation and requirements elicitation as detailed as the software engineering counterparts. There is also no equivalent to the importing software activity, although importing previous models plays an important role in data mining. The main development processes are rather different in software engineering and data mining but the development in data mining is well-covered in CRISP-DM. On the other hand, only a few post-development processes are accounted for in CRISP-DM. It considers planning a deployment, monitoring and maintenance and the generation of a final report. It completely omits the operation of the system by the user, the validation of the results by the user and assuring the right interpretation of users and maintenance of models. There are also no processes for retiring a solution if it is not appropriate anymore.

Integral processes are also only partially covered by CRISP-DM. While model evaluation and documentation development are more or less covered, software configuration management and training are completely omitted. This is crucial because of the amounts of data, models and documents which are generated during data mining projects. Training of end users and data miners to appropriately use data mining methods and interpret their results is also seen as a crucial step by the authors.

Life Cycle, Processes and Tasks

Based on the analysis and comparison of software engineering process models and CRISP-DM the authors propose their own data mining engineering (DME) process model, which includes project management processes, integral processes and organizational processes. Their proposed process model uses KDD at its core to describe the data mining activities. They do not introduce specific phases and a life cycle.

Organizational processes DME contains organizational processes to assure a more effective

organization by setting business goals, improving processes, products and resources. The authors suggest an improvement process in which organizations broadcast their best practices, the methods and tools the use across the organization. Further, they suggest an infrastructure process, to build the best environment for data mining projects and a training process, to train the staff which executes the projects.

Project management processes The project management processes are used to establish a project structure and coordinate and manage project resources throughout the project lifecycle. While CRISP-DM accounts for establishing a project plan with deadlines and milestones, the authors also emphasize other management activities such as controlling time, budget and resources. Projects in the data mining domain should also contain a life cycle selection process that considers other life cycle types than the waterfall with backtracking life cycle used in CRISP-DM. An acquisition and supply process should be included similar as in software engineering. In the initiation process the project structure is set up and a generic life cycle is mapped to a real life cycle. This also requires producing a project plan and identifying major iterations in the project. Further, the required resources for the project should be allocated based on estimates of human resource, budget and effort requirements. Metrics such as the ROI, accuracy, space/time, usefulness and more should be defined. During project planning processes, an evaluation plan, configuration management plan, system transition plan, installation plan, documentation plan, training plan, integration plan and project management plan is built. Further, project monitoring and controlling processes should be established. These processes cover risk management, monitoring whether the activities are going according to plan, retaining records to track the project metrics, identifying life cycle process improvement needs and the collection and analysis of metrics.

Development processes The development-related processes are already well-developed in many data mining process models because they are closest to the technical aspects of the projects and best researched. In the pre-development processes, i.e. before the project kickoff, the authors suggest a concept exploration process. In this process the idea or the need behind a project is identified, a potential solution is formulated and a feasibility study is conducted. Further, business goals and the according success criteria are defined and the tools and techniques available are assessed considering their fit for the project. The authors suggest a business modeling process, which could be close to business modeling in software engineering. Lastly, they propose a knowledge importation process, where knowledge from previous projects can be identified, an importing method is chosen and the knowledge is imported. The development processes consist of requirements processes and the KDD process model. The authors suggest that use case models, as used in software engineering, could be a way of specifying project requirements. After development, the post-development processes start. One part of them is installation, i.e. the transfer of the knowledge acquired via data mining to the user. The transfer can either be in a direct manner or in the form of an integration into software. The latter form is completely disregarded by CRISP-DM. In the operation and support processes, results of the projects are validated, the way users interpret these results are validated and technical assistance is provided to the users. Via maintenance processes, the feedback from the usage of the software is used to further improve it. If models or the software become obsolete, a retirement process is used to retire the obsolete elements from the system.

Integral processes Integral processes are necessary to complete the project and assure its quality. During evaluation defects in the project or its processes are discovered and tackled. Reviews are conducted, tests are developed and executed and evaluation reports are created. Configuration management processes are established to control system changes and maintain the coherence and

traceability of the system. The project members agree on baseline definitions, titling, labeling and numbering of items which should be tracked. The configuration of models should be controlled and changes should be tracked. Lastly, the status of controlled items should be tracked and a change history should be build. Further, documentation processes should be established. Finally, users should be trained in the usage of the solution. This requires the development of training material, the validation of the training program and finally its implementation.

Roles and responsibilities

The methodology does not explicitly define roles and responsibilities.

Artifacts

The methodology does not explicitly define artifacts.

Tools and techniques

The methodology does not explicitly define tools and techniques.

A.1.5 Big Data Managing Framework

In 2014 Dutta and Bose published the b̈ig data managing frameworkẅhich they evaluated based on a case study of the company Ramco Cements Limited [113]. The authors argue that at the time of their writing there was no framework which suited the complexity of big data and most companies still struggled with the execution of their first big data projects. They present a framework which incorporates the requirements of big data and yields a holistic roadmap for conceptualizing, planning and implementing projects with big data contexts.

The authors highlight that big data analytics projects are vastly different from regular IT projects, in that they are often shorter in duration, their discovery cycle leads to better cost control, they are more difficult to implement and often require a change in a company's culture. They argue that the prevalent frameworks of SEMMA and CRISP-DM are not sufficiently tailored towards the use of big data. By examining a case study with regards to their proposed framework they want to yield a scholarly way of developing frameworks.

Life cycle, processes and tasks

The big data managing framework consists of 3 phases with multiple subphases. It is depicted in figure A.3 The 3 major phases are strategic groundwork, data analytics and Implementation.

Phase 1: Strategic Groundwork The strategic groundwork phase builds the foundation for a successful implementation before modeling and analytics can be applied. First, the business problem is identified. They deem a good understanding of the business problem as critical for understanding the potential of big data analytics projects. There should be input by senior management and relevant business unit stakeholders for scoping. This phase should be used to demystify the possibilities of big data and set the right expectations. Once a good understanding of the problem is achieved, research is performed on possible solutions and technologies. The goal is to understand if there are solutions available to solve the business goals and to assess how others achieved similar goals. A probable

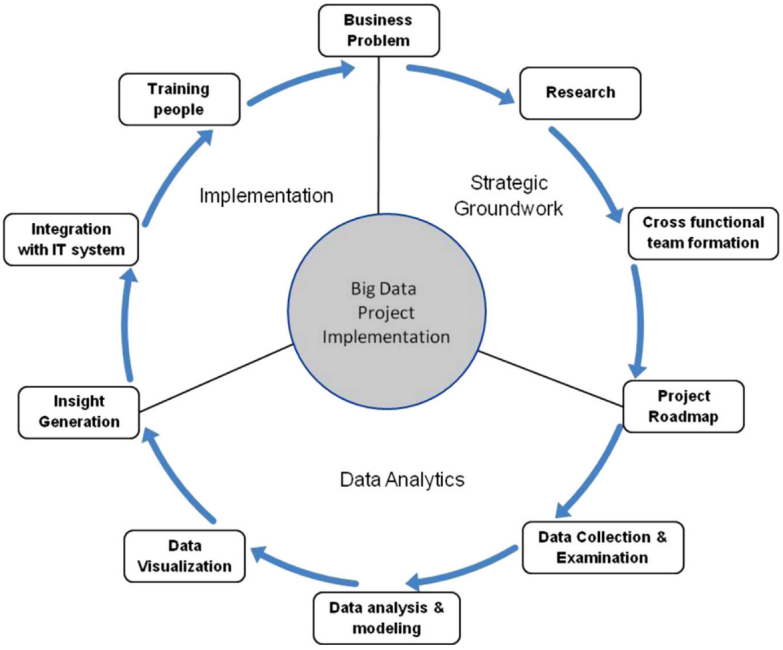

Figure A.3: The phases and subphases of the Big Data Project Implementation framework as used in the case of Ramco Cements Limited [113].

solution method should be conceptualized based on the findings. Next, the project team is formed and the authors stress that a cross-functional team is a key to success. The project should include business stakeholders, IT experts, data modelers and data scientists, experts in cognitive science or customer behavior as well as business decision makers. Lastly a project roadmap with milestones and timelines is developed in which major project activities and deliverables are identified and persons are assigned to activities and milestones. The project roadmap should be designed such that the project team can flexibly adapt to corrections and the focus of the project manager should be to ensure project execution and delivery.

Phase 2: Data Analytics The authors emphasize that the analytics and modeling phase is significantly different in big data projects compared to standard analytics projects. Big data requires handling more data, involving more scalable methods and might require novel visualization methods. This phase starts with data collection and examination of the historically captured data in the organization as well as external data. Once the data is collected quantitative analysis techniques and predictive models are applied to address the business problem. Data visualization methods are used for data insights generation. If no appropriate tools are available, they might need to be developed. Finally, with the results of the modeling and the help of the visualization techniques insights are generated with which value can be generated for the organization.

Phase 3: Implementations In the last phase of the project, models, trends and visualization tools are deployed and integrated with existing IT solutions. It has to be assured that the models work with the existing IT systems and that the systems, their architecture and processes meet the requirements of the solution. Furthermore, the integration processes need sufficient testing. After integrating the system, the people who should use it should be trained and motivated to use it. User acceptance is

critical for the success.

Roles and responsibilities

Six different roles are mentioned in the methodology, only some of them with clear responsibilities and tasks. The framework puts a strong emphasis on the involvement of management and senior management. They identify this as a critical success factor. In addition, they advertise cross-functional teams consisting of relevant business stakeholders, IT, management, data scientists/modelers, customer behavior experts and users.

Artifacts

The framework mentions six artifacts, however it does not present them in a structured way or give any more details about them. The six artifacts mentioned are a project roadmap, key project requirements, lists of deliverables, a project schedule, project milestones and progress reports, created in varying frequencies.

Tools and techniques

The framework suggests some tools and techniques to best understand the business problems and track the progress of the project. Namely they suggest performing strategy sessions, interviews, workshops and establish a progress reporting system.

A.1.6 Big Data Ideation, Assessment and Implementation

In 2015 Vanauaer et al. introduced a methodology to guide the introduction of big data in organizations [114]. Although this methodology is not a project methodology but rather a company transformation methodology it contains valuable ideas for data science projects. Especially the assessment of the value of big data based use cases and the requirements that a company needs to fulfill in order to implement them, can be valuable to the early phases of data science projects. Therefor we briefly summarize the methodology here.

The authors criticize that existing methodologies are often not scientifically grounded and base their methodology on IT value theory, workgroup ideation processes and enterprise architecture management. Their methodology follows the ideation, assessment and implementation of big data use cases in a company. They distinguish companies which have a business first and a data first approach to the introduction of big data. Business first (BF) refers to companies which introduce big data based on their business requirements, while data first (DF) refers to companies which want to pursue new business models via the use of big data.

Life cycle, processes and tasks

The methodology is split into an ideation phase and an implementation phase, which are both further split into transition and action subphases. The subphases contain multiple steps. Some of these steps are specific to the BF and others to the DF approach, while others are performed in both approaches. Both, the business first and data first view, as well as all the phases and subphases are depicted in figure A.4.

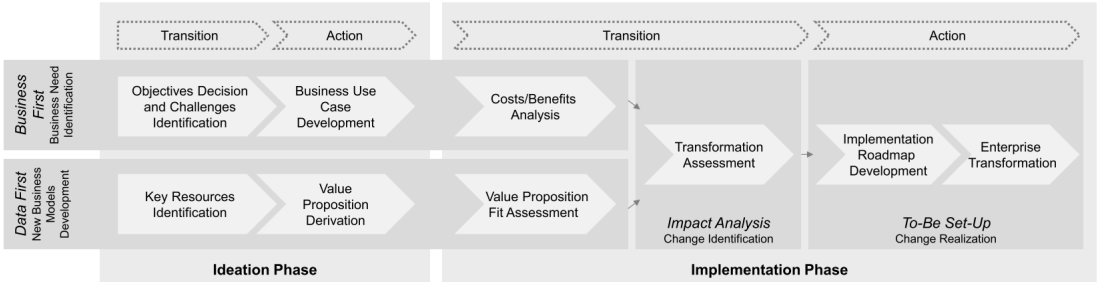

Figure A.4: The business first and data first views, phases and subphases of the big data ideation, assessment and implementation methodology [114].

Ideation Phase The ideation phase is concerned with the generation of new ideas and the application of solutions to new use cases to generate value for the organization.

BF approach In the transition subphase the BF approach mainly addresses the questions ”Which enterprise goals need to be fulfilled?” and ”Which challenges hinder the company in reaching the goals?”. Requirements engineering methods are used to answer these questions. The main assumption underlying the BF approach is that the objectives and challenges of the company can be solved via big data analysis, so an improvement of operations with new data is expected. As a result, measurable goals are collected along with the challenges and the use case development is scoped accordingly. In the action subphase the use cases are then developed based on the scope and context from the transition subphase. The use cases should now either yield a business improvement or solve a given challenge. In order to determine the correct use cases, domain knowledge of the operational processes is required. The authors suggest involving first- and mid-level management along with operations personnel and the data scientists. Together they can better discuss the potentials of operations and data. The action subphase concludes with a list of big data use cases or the realization that there are none.

DF approach The transition phase of the DF approach starts with the identification of key resources. New business models are designed by harnessing the potential of new data and analysis methods. In order to describe these business model, one has to consider the potential customers, resources needed and much more. The authors suggest using a business model canvas for this task. Along with the central value proposition the question of available data, which is a key resource, should be raised. This should not only cover already existing data but also other available data sources, either from the organization or external. The transition subphase is completed when a high-level overview of the company’s potentially capturable data is prepared. In the action subphase a value proposition and business model are developed on the basis of the available data. The authors suggest using methods like visual thinking, prototyping, storytelling and scenarios to generate ideas and encourage the innovators to combine multiple data sources. In this subphase the feasibility is not yet assessed so that every idea is considered. The subphase ends with complete sketches of the business model canvas. Aside of the management, sales, enterprise architects and strategists should be involved here.

Implementation phase

In the implementation phase, the business models and use cases are implemented.

BF approach The transition subphase of the BF approach starts with a costs-benefits analysis. The use cases are prioritized according to the impact on the organization objectives. This includes an assessment of the financial validity in the long-term use, whether the use cases enhance the fulfillment of enterprise objectives and a business case calculation.

DF approach The DF approach starts with a value proposition fit assessment. Aside of the financial validity, the fit of the business model to the organization is checked. This includes verifying whether all key resources are available and all requirements for the implementation are met or can be met with significant overhead. This subphase is completed when the business models are prioritized or dismissed.

Impact Analysis Now the approach shifts from an either data first or business first approach to a unified approach. In the transition subphase the transformations to the organization are assessed. In order to integrate the solutions in the enterprise architecture the needed resources need to be identified as well as the complexity of the transition to the desired to-be state. This includes an investigation of the interrelation between processes, data and technology. Changes to the IT infrastructure could pose severe costs and the organization needs to consider the cost of appropriate infrastructure for data transfer, storage and analysis. Depending on the requirements an in-house infrastructure or an external infrastructure can be chosen. According to the authors it might be a good choice to let experts consult the organization on these questions. The impact analysis concludes with a modeled to-be scenario and the decision whether to realize it or not.

To-be set-up The to-be set-up phase concludes the methodology. It is comprised of two action subphases. In the first action subphase an implementation roadmap is developed. The implementation needs to be carefully planned and the roadmap should consider technical and data security governance, legal restrictions and transition architectures. In the second action subphase the enterprise transformation is executed. The implementation roadmap is implemented based on methods from enterprise architecture management. Finally, after the implementation, the performance is measured and compared to the pursued objectives.

Roles and responsibilities

The methodology specifies roles and their responsibilities for each of the phases. It involves management roles as well as operational and supporting roles.

1. Business First
    a) Senior management strategists: working on transition objectives during ideation
    b) First/ mid-level management: working on transition challenges during ideation and implementation
    c) Operations personnel: working on implementation
    d) Data scientists: working on implementation
    e) Relevant Innovators from Business units: Support the implementation
    f) Enterprise architects: working on the transition, if an enterprise architecture model exists and on the impact analysis and to-be set up
2. Data First
    a) First/ mid-level management: working on the transition
    b) Enterprise architects: working on the transition, if an enterprise architecture model exists and on the impact analysis and to-be set up
    c) Senior management: Working on the implementation with support of strategists

d) Strategists: working on the implementation

e) Sales: working on the implementation

f) Data Scientists/Analysts: working on the implementation

g) Relevant Innovators from Business units: Support the implementation

Artifacts

The methodology defines various artifacts in the different phases of both the data first and business first approach.

1. Ideation
   a) Business First
      i. business requirements in the form of organizational goal and existing and known operational challenges
      ii. Use cases, which are measurably improving identified objectives
   b) Data First
      i. Knowledge on organization and potentially available data, market expertise regarding sales opportunities
      ii. Potentially sellable value proposition (service) with customer segments
2. Implementation
   a) Business First
      i. Use Cases, Models including KPIs of current situation, accounting numbers
      ii. Prioritized Use Cases for transformation assessment
   b) Data First
      i. Business models, organizational goals
      ii. Prioritized business models for transformation assessment
3. Implementation: Impact Analysis (both Data First and Business First)
   a) Enterprise architecture model of relevant parts of enterprise, use case/ business model to realize
   b) To-be Enterprise architecture model including assessment of changes
   c) decision to realize to-be scenario
4. Implementation: To-be set-up (both Data First and Business First)
   a) architectural inputs
   b) implementation and migration plan
   c) realized transformation

Tools and techniques

The methodology names tools and techniques which can be used in the different phases, again they are structured by phase and by business first or data first approach.

1. Ideation
   a) Business First
      i. Enterprise Architecture Models
      ii. Goal analysis techniques
      iii. Creativity techniques
   b) Data First
      i. Business Model Canvas
      ii. Enterprise Architecture
      iii. Creativity Techniques
2. Implementation
   a) Business First
      i. Controlling
   b) Data First
      i. Enterprise Architecture Models of strategy
      ii. Goal analysis techniques
3. Implementation: Impact Analysis (both Data First and Business First)
   a) TOGAF (The Open Group Architectural Framework) ADM B-D
   b) Enterprise Architecture Views
   c) Architecture Analysis
4. Implementation: To-be set-up (both Data First and Business First)
   a) TOGAF (The Open Group Architectural Framework) ADM phase E-F
   b) TOGAF (The Open Group Architectural Framework) ADM phase G-H

A.1.7 Big Data Management Canvas

The Big Data Management Canvas [34] aims to fill the gap of research on the computing part of big data analytics and management sciences. The authors realized that most existing methodologies and reference models either address the technical side of big data projects or the business value generation but fail to connect both. They argue that a joint approach is necessary for successful projects. They analyze the existing big data reference models proposed by NIST [40], OECD [258] and Lim [259] and highlight their benefits and shortcomings. Their Big Data Management canvas integrates ideas of these approaches and incorporates the 5V of big data, the value-oriented approach of the OECD model, the focus on action, visualization and access of the NIST model, the market focus of Davenport and

the value creation by Lin. Their canvas aims at optimizing the management of big data by optimizing the phases of data preparation, data analysis, data interaction, data effectuation and data intelligence. They provide both a business view as well as a technical view of all of the phases and argue that in all phases business users and IT should closely collaborate to assure that business value is always ensured.

Life cycle, processes and tasks

The methodology introduces five phases.

Phase 1: Data Preparation In the data preparation phase, business users identify relevant data sources for their use cases. The IT is responsible for data integration and extracts data from these data sources, transforms them and loads the data into a coherent database. The vast amounts of data available in big data settings make it necessary to keep scalability in mind.

Phase 2: Data Analytics The goal of the data analytics phase is to transform the raw data into information by data analysis methods and tools. To this end the IT provides the necessary analytic software to select, aggregate, cluster and extrapolate data. The business side is responsible for generating knowledge by analytic processing with data science methods. The big data requirements make it necessary to consider scalability and might benefit from using parallel computing architectures.

Phase 3: Data Interaction After generating analytics results, the users interact with the data to generate knowledge. The IT provide user interfaces for the interaction of human decision makers with the data analytics results. This might require user experience design and data visualization skills to promote interaction of users and data. From the business perspective, business units interact with the results from the data analytics to generate knowledge.

Phase 4: Data Effectuation After generating knowledge from data it is relevant to utilize this knowledge for value creation in products, services and the operations of organizations. IT can support this with data forwarding, i.e. proactively feeding the analytics results into operational systems. The business side utilize the results of the BDA to create the actual business value.

Phase 5: Data Intelligence The data intelligence phase spans over all four other phases of the canvas and assures the optimal deployment, distribution and utilization of the knowledge assets in big data management. The knowledge assets include the knowledge generated from the data, knowledge about the data and the data management as well as knowledge and skills necessary for data analytics and management.

Roles and responsibilities

The methodology does not explicitly define roles. It suggests that business and IT must always work together, in each layer of the canvas.

Artifacts

The methodology does not explicitly suggest artifacts, however they show examples of artifacts from three sample use cases.

Tools and techniques

The methodology does also not explicitly suggest tools and technologies but shows examples from three sample use cases.

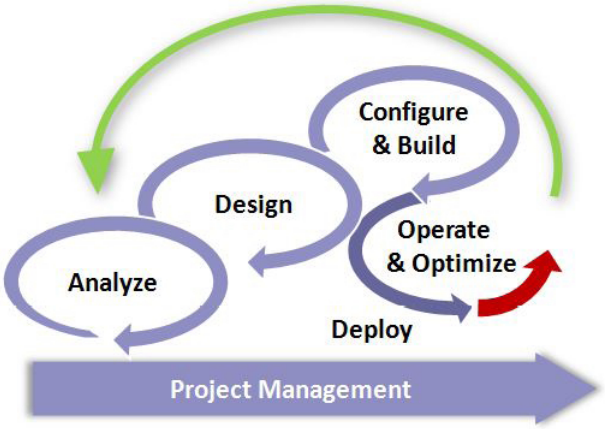

Figure A.5: The ASUM-DM cycle with its 5 phases and the project management stream.

A.1.8 ASUM-DM

ASUM-DM was introduced in 2015 by IBM [116]. While it pertains many ideas from CRISP-DM, it also extends and modifies it to tackle some of its shortcomings. Most notably ASUM-DM incorporates continuous project management, it puts a large focus on agile development, prototyping and deployment and also ensures maintenance of deployed solutions on proper infrastructure.

Life cycle, processes and tasks

Figure A.5 shows the 5 phases of ASUM-DM: Analyze, Design, Configure & Build, Deploy and Operate & Optimize. In contrast to CRISP-DM the cycle also contains a continuous project management function that accompanies all of the phases to ensure project success. Similar as in CRISP-DM, ASUM-DM does not follow a linear structure. While phases are dependent on each other, there can be multiple iterations through phases. ASUM-DM even encourages rapid iterations through the whole cycle. Figure A.6 shows the detailed breakdown of all phases into activities.

Phase 1: Analyze In the analyze phase all functional and non-functional requirements of the project are defined and are agreed on by all stakeholders in the project. This phase is also used to define success criteria.

Activities in the Analyze phase: The phase begins with a meeting with sales to clarify project details and the identification of available and necessary resources. Next is the assessment of the customer readiness for implementation and the identification of solutions to mitigate obstacles. The project team then prepares a kickoff and conducts it together with the customer. After the kickoff the business understanding is formed and the data mining goals are converted into a project plan. Additionally, documentation about the business understanding is created. Lastly data understanding is generated by exploring the data to avoid problems during the data preparation phase.

Phase 2: Design The design phase is used to define all components of the solution and identify dependencies among them. Resources are identified and a development environment is installed. Finally, iterative sprints are used to develop prototypes that can be used to clarify requirements and evaluate the components.

Activities in the Design phase: The design activity comprises of the design of technical as well as

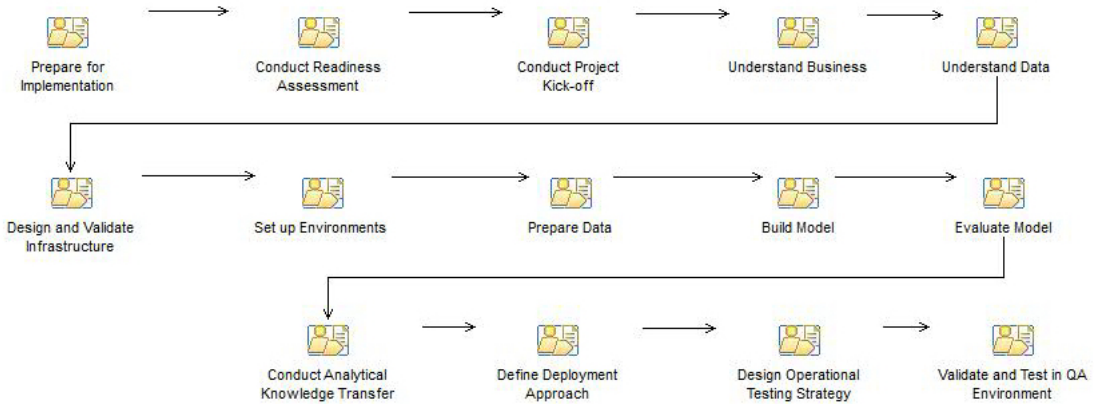

Figure A.6: The detailed activities view of ASUM-DM.

the security infrastructure. According to the design and requirements the appropriate environments are either set up onsite or on cloud. The data preparation activities are then similar to the CRISP-DM cycle. It is noted that this is one of the most crucial and time-consuming activities. Appropriate models for the data mining tasks are selected, tuned and evaluated. The design choices are communicated to the clients team and finally deployment approaches and an operational testing strategy are defined.

Phase 3: Configure & Build The components are configured iteratively and incrementally, build and integrated in the Configure & Build phase. The tests of the components are now expanded to multiple environments and validation is planned based on the V-model.

Activities in the Configure & Build phase: The validation and testing of the system is performed on an quality assurance environment. To this end the system and performance are validated and user acceptance tests are performed. Based on these results, tweaks and fixes are undertaken. Finally, a production deployment decision is made.

Phase 4: Deploy A support and maintenance plan is created and the solution is migrated to the production environment. The business users are notified and the plans are communicated to them.

Activities in the Deploy phase: The non-analytical knowledge is transferred to the client to assure that the solution can be used once the project ends. Maintenance and monitoring activities for the production environment are scheduled and then the solution is deployed. Support is now part of the project and the system launches. Lastly the project closure is prepared with close out meetings and internal and external reports.

Phase 5: Operate & Optimize The solution is maintained and quality assurance is employed to guarantee a successful use of the solution

Activities in the Operate & Optimize phase: The activities of this phase include monitoring of the model, ensuring proper operation by optimizing and improving the system, the support of the user community, managing of infrastructure and assessment of the systems performance, quality and benefits.

Project Management During the whole process, project management is used to help with communication and assure a healthy project progress. However, ASUM-DM does also suggest how to implement project management. Project Management processes along with descriptions are given and structured into the Initiate, Plan, Execute and Close process groups.

Agile Development in ASUM-DM Whenever possible ASUM-DM suggests combining a hybrid

approach of agile and traditional implementation. Especially in the beginning when requirements are assessed, they are refined in iterative prototyping sprints. IT- and business personnel work closely together to assure a proper direction of the project. After validating the results of the prototyping sprints, iterative and incremental development is used to build and configure the solution.

Roles and responsibilities

IBM provides detailed descriptions for the roles in ASUM-DM. They introduce 16 roles and an assignment of all tasks to the roles. ASUM-DM introduces the following roles: client application administrator, client business sponsor, client data analyst, client database administrator, client key system user, client network administrator, client project manager, client security administrator, client stakeholders, client subject matter experts, client support manager, client tool administrator, data minder/data scientist, enterprise architect, project manager and SPSS project manager. For detailed descriptions of these roles we refer the reader to the appendix.

Artifacts

The ASUM-DM teaser page [116] lists artifacts which are used in the different phases of project management in the methodology but does not explicitly list artifacts generated during the data mining processes, e.g. data and models. The project management artifacts are specific to IBM internal processes and documents. They are structured according to the phase they belong to and whether they are used as input or output.

- Initiate
  - Input: Sales process outputs for IBM Analytics products, Business Case, Contract for licenses and services, Pre-project reports (Proof of concept, pilot projects, hardware and technical recommendations, support options selected, etc)
  - Output: Project initial team mobilized and engaged, Approved Project Success Plan, Approved Financial Management Plan, Approval to commence Design work
- Plan
  - Input: Project initial team mobilized and engaged, Approved Project Success Plan I, Approved Financial Management Plan
  - Output: Approved Project Success Plan II, Updated Financial Management Plan, Project Team fully engaged in the project, Approved Design, Confirmation to commence Configuration and Deployment
- Execute
  - Input: Approved Project Success Plan II, Updated Financial management Plan, Project Team fully engaged in the project, Approved Design
  - Output: Attain project objectives as defined in the PSP on time and within budget
- Close
  - Input: Customer approval that the Configure and Deploy phases are complete, Customer approval that the project objectives have been achieved

- Output: Go-Forward Plan, Released resources, Formal project closure, internal project review and lessons learned

Tools and techniques

The ASUM-DM methodology is supported by the SPSS modeler software from IBM, which is a dedicated commercial tool for executing data mining projects.

An Extension of ASUM-DM for cross-organization and -discipline projects

Angee et al.present an extension of the ASUM-DM methodology [104]. They argue that CRISP-DM and ASUM-DM in their original versions are not suitable for projects with big data settings and multi-discipline multi-organization set ups. They propose to add multiple activities to the ”Analyze-Design-Configure & Build” phases of ASUM-DM as can be seen in figure A.7. In the data understanding phase, they add the activity to describe data against big data’s 5 V’s. The idea behind this activity is to better understand challenges posed by the data and to understand effects on the requirements and success criteria. They propose to add activities for the development of a functional prototype, which includes defining a prototype workflow based on the Business Process Model and Notation (BPMn), a visualization of the analysis results and the evaluation of the prototype against the business requirements and success criteria.

To tackle challenges which occurred because of the multi-discipline multi-organization setting, the authors suggest project management measures. They suggest that the project plan is created jointly by a project management team. This gives them the opportunity to review their common business understanding. Based on the understanding, objectives and goals can be formulated and mapped against available resources, risks and constraints. It also helps with finding suitable methodologies for the tasks and teams. The authors also suggest every team to have weekly internal meetings and quarterly cross-team meetings to assure information exchange. Another suggestion is the usage of a joint document management system to document and share developments, decisions and information.

A.1.9 Foundational Methodology for Data Science

In 2015 IBM introduced the foundational methodology for data science (FMDS) [115]. Their observation was that data science was a rapidly growing market for which more and more technologies were emerging. These emerging technologies automated steps in model building and application and made data science more accessible. However, it also introduced the need for adjusted data science methodologies. With their methodology IBM explicitly tackled ”top-down” projects in which a business problem leads to data science questions.

Their methodology bears resemblance with known methodologies such as KDD, SEMMA and CRISP-DM but enriches them to cope with emerging trends such as big data, text analytics in predictive modeling and the aforementioned automation of machine learning processes. Their aim is to introduce a methodology which is a guiding strategy regardless of technologies, data volumes or approaches.

Life cycle, processes and tasks

FMDS consists of 10 phases and describes a highly iterative process to account for the nature of data science projects in which models are often not only trained and deployed once but refined over the

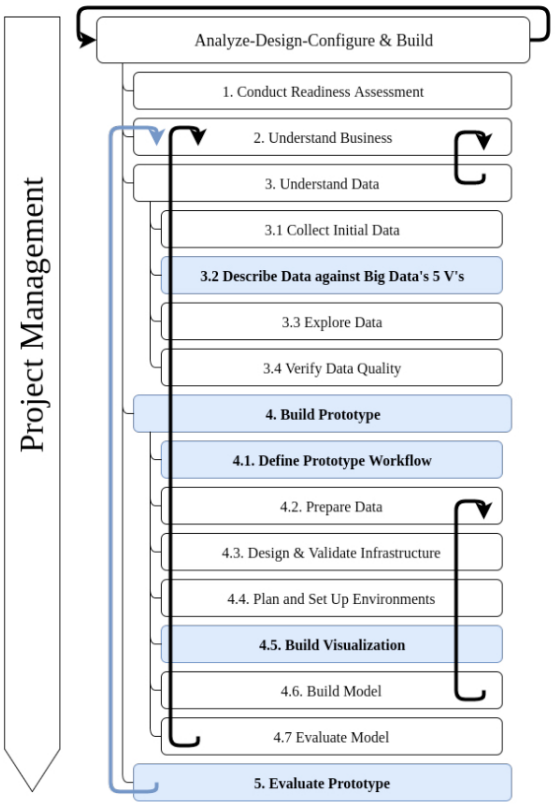

Figure A.7: The extension of the ASUM-DM phases ”Analyze-Design-Configure&Build”. Changes by the authors are highlighted in blue. [104]

course of multiple iterations. The life cycle and the phases are depicted in figure A.8.

Phase 1: Business Understanding FMDS starts with a business understanding phase in which business sponsors play the most important role. They deliver a problem definition, the objectives of the project and requirements to the solution from a business perspective. This phase lays the foundation for a successful resolution of the business problems. The authors argue that the business sponsor should not only be involved during this phase but during the whole project, to ensure that the project remains on track for the intended solution, to provide domain expertise and to review intermediate findings.

Phase 2: Analytic approach After obtaining a good business understanding, the data scientists match the business problems with analytical approaches. The goal of this phase is to express the problems in terms of statistical or machine-learning techniques and identify the most suitable ones for the project.

Phase 3: Data requirements In the third phase data requirements are elicited based on the chosen analytics approach. These requirements cover the content of the data, its formats and suitable representations. Domain knowledge is very useful in this phase.

Phase 4: Data collection After collecting the data requirements, suitable data resources are identified and gathered. The authors emphasize that it is important to keep the business problem and domain in mind while gathering the data. They suggest holding out expensive data investments until

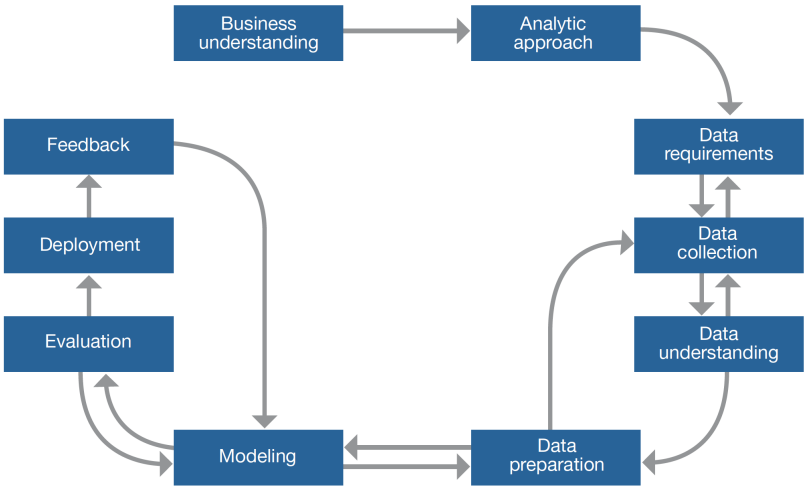

Figure A.8: The 10 phases of the foundational methodology for data science and its life cycle [115].

the data scientists gain more insights into the data and models in later phases. Notably the authors also suggest that data sampling and subsampling are still important, but less so due to more powerful hardware. They argue that there are benefits to using larger datasets such as better model and better representation of rare events.

Phase 5: Data understanding After obtaining the data, the data scientists should get insights into it by using descriptive statistics and visualization techniques. This helps to better understand the data content, assess the data quality and generate first insights. A possible result of this phase is that more data is needed or that the data is not suitable for the project.

Phase 6: Data preparation Before the modeling can start, the data needs to be prepared to adequately suit the model. This includes cleaning the data, combining data, transforming the data and feature engineering. Feature engineering can benefit from domain knowledge. Features can also be enriched with text analytics, e.g. by adding sentiment scores. This phase is considered to be the most time-consuming and the authors see potential for automation in this phase.

Phase 7: Modeling The modeling phase is used to build predictive or descriptive models according to the analytic approaches of choice. Some of the approaches require a training set to build the model. This phase is highly iterative and may lead to refinements in the data preparation phase and the model specification. Data scientists will typically try multiple algorithms with different parameter settings.

Phase 8: Evaluation After completing the modeling phase, the model quality is evaluated. It is important to adequately assess how well the model solves the business problems. This phase includes computing diagnostic measures and produce tables and graphs as outputs to enable model quality interpretation. In a supervised machine learning setting the quality should be assessed on a held-out test data set. In particularly sensitive projects it might be useful to compute statistical significance of the model results.

Phase 9: Deployment Once the business sponsors confirm that the models have satisfactory quality, they are deployed in a productive or suitable test environment. Typically, deployment is not directly done in the final environment, or only partially. First, the performance of the deployed solution is assessed in the deployment environment. The authors also note that the result of this phase does not

necessary need to be a deployment of software but could also be a generated report, depending on the goal of the project.

Phase 10: Feedback In the last phase the business sponsors and data scientists collect feedback from the implemented model on performance and gather insights into the impact on the deployment environment. This feedback can be used to refine the models and improve the model quality and usefulness. The authors argue that feedback collection, model optimization and deployment can be automated.

Roles and responsibilities

The methodology mentions the roles of a business sponsor and a data scientist as well as groups of people who are involved with the deployment of a solution. They provide brief descriptions of some of their tasks:

1. Business sponsor: Defines the problem, objectives and solution requirements from a business view, involved throughout the project to provide domain knowledge, review results and ensures that the project remains on track
2. Data Scientist: Defines the analytical approach to solve the business problem, performs data collection and revises it if necessary, perform descriptive analysis of the data including the study of data quality and initial insights, perform feature engineering by utilizing domain knowledge, build models and test it
3. Deployment personnel: People from within an enterprise who will use the model together with a development team

Artifacts

Artifacts are not explicitly defined in this methodology.

Tools and techniques

Tools and techniques are not mentioned in this methodology.

A.1.10 Towards Methods for Systematic Research on Big Data

In 2015 Das et al. published the paper ”Towards Methods for Systematic Research on Big Data” [5]. In the paper they thoroughly analyze what characterizes data-driven research. They state that data-driven projects are often of an exploratory or ad-hoc nature, which makes it difficult for data scientists to set expectations properly or create a project plan. They attribute this to a lack of processes and methodologies which appropriately fit these types of projects.

They deliver multiple definitions of the term data science, ranging from the usage of machine learning and statistics on big-data and multi-structured data to extract knowledge, to the building of data products. They follow the definition of the 5V to define big data and state that data products need to meet big data requirements. They distinguish hindsight-oriented and insight-/foresight-oriented projects in data science and highlight that there are key differences to traditional database-focused methods for knowledge discovery. While data science projects often yield probabilistic answers and

require strong hardware for exploration at scale, database-focused methods yield concrete answers and require fast access and summarization of results.

The authors present four use cases which are all in the domain of text analytics. The use cases cover churn prediction for newspaper subscribers, an exploratory analysis of the use of emotion-category related words, exploratory analysis and classification of the language of successful bloggers and labeling of tweets associated with brands and products.

Life cycle, processes and tasks

Das et al. deliver an exhaustive characterization of research projects. They distinguish typical research projects from data-driven research projects. In typical research projects the researchers collect data and build and validate a model based on the data. In data-driven research, data is given first, then patterns and information stored in the data are revealed and a research goal is established based on these insights. Further, the goal is not static but evolves with more insights from the data.

A key characteristic of research according to the authors is the clarity about the purpose of the research. The purpose is the driver of considerations of design, measurement, analysis and reporting of the research project. Typical research purposes are the contribution to fundamental knowledge and theory, illuminating societal concerns or problems in search for solutions, determining if a solution works, improving a policy or program which is being implemented or understanding solving a problem as quickly as possible. Depending on the data and research goals, qualitative, quantitative or mixed-methods are used for evaluation. The data can be available from internal or external resources, which can either be public or private. In big data settings the data is often heterogeneous, complex and available in masses, requiring methods which scale well. Further, data ownership and distribution play an important role and it should be a goal to reduce overhead when computing in a distributed manner, effectively consolidate results and address data security issues. There are various types of experiments which need consideration, such as validating hypothesis, studying specific objects, studying models instead of concrete objects, empirically studying data to generate new ideas, qualitative and quantitative experiments, forecasting and retrospection. The type of analytics applied for the research also yields an important characterization. The authors distinguish predictive tasks, namely classification, regression and recommendation from descriptive tasks, such as cluster analysis, anomaly detection and association analysis. In contrast to traditional research, in data-driven research often thousands of hypotheses are generated, calling for automation of their testing, and data sets are not always sampled in a representative manner. The final means of characterization that the authors mention is infrastructure considerations. These considerations have to include the mode of data acquisition, where to store data, how much computational resources are available, whether resources should be backed up, whether project members with technical and domain expertise are available as well as support personnel. Here, the characteristics of big data need to be considered.

Agile Analytics The authors suggest using a methodology guided by Agile Analytics [260]. They follow the recommended development style suggested by Collier. The practices of an enterprise should be tailored toward its environment with high quality, high value analytic systems in mind. Improvements should be iterative and incremental evolutionary, i.e. in chunks of user-valued functionality with frequent adaptions to user-feedback. Collier advertises value-driven development. In each development iteration, user-valued features should be produced. These features should meet the needs of the user, which often care most about presentation and access to information which help improve their work and business. A strong emphasis should be put on production quality by debugging

and testing each feature during an iteration cycle. Processes should yield just enough structure and flexibility and whenever possible routine processes should be automated. The project manager should enable collaboration among all relevant stakeholders and should enable the teams to perform rather than dictating them, what to do.

The Research Project Process The process that the authors suggest for research is centered around data. Figure A.9 shows the different phases in the process and how the data is transformed after each step during the process. Although the data is the central element of the process, the final research output and hence the question of what to get from the research and data should be known at the start of the process, because they determine how to handle the data and what to aim for.

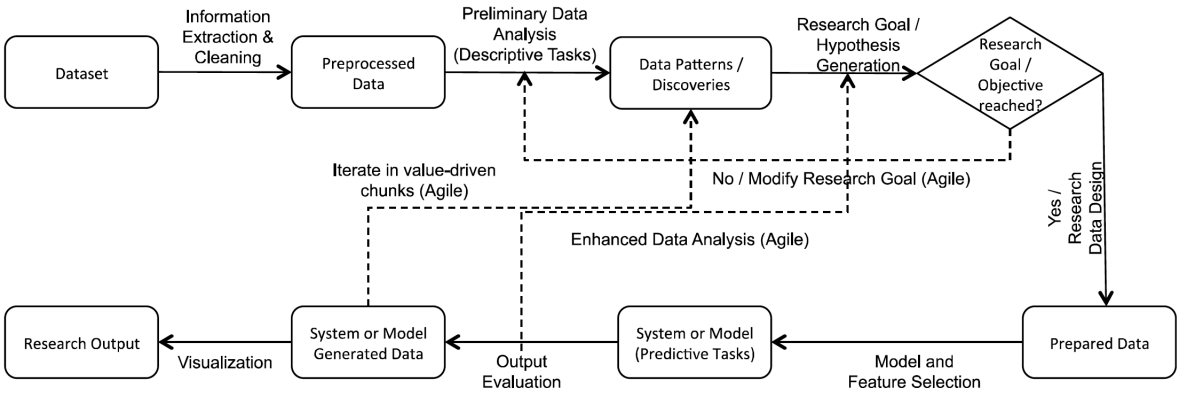

Figure A.9: The process for systematic data-driven research as proposed by Das et al. [5].

Starting from an initial dataset, the authors suggest an information extraction and cleaning step, in which irrelevant data is removed. The now preprocessed data is used for a preliminary data analysis, to find the target and starting point of the research by revealing relevant information from the data and clustering or narrowing down the data to relevant parts. Based on the discoveries and data patterns, a research goal or hypothesis is generated. A well-defined research goal is a key success factor for a research project and will determine subsequent steps and requirements for the data output. Based on the research goal, the data is further prepared and analyzed to assure the fit of the data for the research question. Appropriate models and features are selected to convert the data into the desired outputs. The model results are evaluated with respect to the research goal. After the evaluation, more iterations of the previous steps can commence, to enhance the model performance. The results of the evaluation are visualized and communicated. Any or all steps of the process might require iterations to reach the desired outcomes. It is advisable to start with smaller subsets of the complete dataset, to ensure quick results. To optimize future projects datasets as well as data processing scripts should be stored for reuse.

Roles and responsibilities

The authors do not provide definitions of roles and responsibilities.

Artifacts

No concrete artifacts are mentioned in the process aside of the data itself.

Tools and techniques

No specific tools and technologies are referenced.

A.1.11 KDDS + Data Science Edge

In 2017 Grady et al. introduced the Knowledge Discovery in Data Science Process and along with it the Data Science Edge process model [31]. They argue that big data systems introduce new challenges and requirements which are not sufficiently covered by KDD and CRISP-DM and call for a modernized standard process that addresses the rapid developments and challenges in machine learning application development and deployment.

They see several shortcomings of KDD and CRISP-DM which they address. According to them KDD is not well-integrated with management processes, is not aligned with software and agile development methodologies and should provide method guidance instead of checklists. CRISP-DM on the other hand does not deal with the challenges of big data analytics such as data storage, big data algorithms and analytics techniques and it also not addresses automation of processes and systems development methodologies.

Grady et al. criticize that CRISP-DM more or less follows a waterfall approach which they deem not appropriate and suggest a more agile procedure. The main disadvantage of waterfall models that they see is that failure, especially in later phases, often leads to the necessity of restarting the process or at least going into very early phases of the project. Agile mitigates or eliminates three critical deficiencies of the waterfall model:

1. **Human factors in software development:** Agile focuses and promotes collaboration in a team. It encourages product owners, stakeholders on the customer side and developers to collaborate frequently. Teams should be kept stable over the course of the development process to enhance performance and the group dynamics over time. Due to short iteration spans of 2-4 weeks rapid customer feedback is facilitated, the teams can respond more quickly to changing requirements, mitigate the risk of mistakes and enable continuous integration and testing.

2. **Changes in cost and risk factors:** A core principle of agile is to deliver software early and often. This leads to an early return of investment and allows the development to more quickly reach the level of a minimum viable product (MVP)

3. **Understanding the nature of complex systems:** Agile considers software projects as complicated or complex systems, whereas waterfall treats software development projects and software systems as simple. The iterative delivery of agile better suits the development of complex systems.

Life cycle, processes and tasks

KDDS introduces 4 phases in which DSE embeds 5 phases. Each of the 5 phases results in the generation of data at increasing levels of maturity.

Phase 1 - Assess: The Assess phase aims at deriving goals, a solution architecture, the project requirements and critical success factors. The authors recommend starting this phase with an assessment of the return of investment and the readiness of the organization before starting the work on the project. This phase includes the **Plan** phase, which might involve multiple organizations, such

as service providers, data providers and other stakeholders. Depending on the project setup it might be necessary to implement more coordination tasks and to do some additional contractual work to settle privacy and security issues. In this phase it is also important to assess any regulatory and policy issues, the acceptance of a solution by the relevant stakeholders and whether the staff that should use the final solution is appropriately skilled.

Phase 2 - Architect: In this phase, the previously determined requirements, are translated into solutions for a new system. This might include the analysis of alternative solutions. The authors emphasize that operational concerns about the deployment should already be considered in this phase to avoid problems in later phases. This phase contains the **Collect** phase in which data is captured and stored. Big data requirements such as throughput make the choice of database especially important. As much information as possible about the data should be collected to correctly distribute it, if possible. If data needs to stay with a data provider, policies for querying, curating and analyzing the data need to be established. After this phase the data is available as collected. The collect phase is followed by the **curate** phase in which the data is explored, cleansed, contextualized and stored. For exploration, visualization techniques might be considered to improve human understanding of the data quality, distribution and potential features. Privacy concerns must be considered, especially when fusing datasets, as this might lead to critical connections. Since big data specific databases and algorithms are able to handle huge volumes of data, sampling might play a smaller role than before. However, the authors argue that metadata becomes more important and suggest clear provenance and history of the dataset for proper curating. The data quality should also be evaluated at this phase.

Phase 3 - Build: The third phase consists of the development, testing and deployment of the technical solution. A big difference to traditional software development is that data and model quality might affect the duration of this phase. Problems with the data quality could lead to further data cleansing and transformation, whereas problems with the model quality might require additional modeling or the selection of new models. This phase contains the **analyze** phase in which the data is queried, the modeling occurs, the models are evaluated and finally stored. The expectation of the system users play an important role here and should be appropriately managed. Iterative hypothesis generation and testing could be helpful. DSE also suggests using simpler methods first as they might be sufficient while reducing computational complexity. In distributed big data settings, concurrency, latency and the possibility to parallelize the algorithms play an important role in algorithm selection. At the end of this phase the data is available as contextualized and organized information.

Phase 4 - Improve: The last phase considers the operation and management of the system and incorporates the analysis of innovative ways of improving the performance. This phase includes the **act** phase which deals with the UI of the system, interpretation of its results, explainability, deployment and export. Visualization is crucial to communicate results to non-technical decision makers. Visualizations might include infographs and interactive dashboards and combine human factors with graphics design. The deployment must consider additional requirements to the data exchange between organizations or systems with different security classification levels.

Roles and responsibilities

The methodology does not explicitly define roles but mentions product owners, stakeholders, users and developers. It also recommends to keep teams stable so that they optimize their performance in joint working. Further the methodology states that teams benefit from an agile approach because of faster learning cycles.

Artifacts

The methodology does not explicitly define artifacts.

Tools and techniques

The methodology only mentions NoSQL as a data storage option.

A.1.12 Agile Data Science Methodology

The agile data science methodology was introduced in 2017 by Jurney [6]. The goal of the methodology is to help teams turn research into analytics applications while embracing an agile approach. Jurney argues that most agile approaches to data science have been unsatisfactory because they did not account for the fundamental differences between software development and data science. According to him the biggest differences are that data science is looser than software engineering and requires coping with uncertainty. Traditional software development methodologies do not account for these characteristics.

The Agile Data Science Manifesto Jurney introduces the agile data science manifesto which is adapted from the agile manifesto. The goal of the manifesto is to enable teams to ship working software and to align data science with the rest of an organization. He stresses that there is a misalignment between research and engineering which leads to waterfall-like projects. The research which data scientists need to perform in project is time-consuming and unfit for traditional scrum sprints. During the research time, engineers wait for results from the data scientists, pressing the applied researchers for time. The manifesto is aimed at the productive synthesis of research and engineering. It accounts for big data requirements, i.e. processing data at scale, and promotes the use of web applications to share and deliver insights into data. It puts an emphasis on iterative work, product and insight delivery, meeting user needs and maximizing the focus of features on the essential parts of a product. The principles of the agile data science manifesto are:

1. **Iterate, iterate, iterate: tables, charts, reports, predictions** - Iterations are key to success. Typically, useful insights come only after repeated failures. Preprocessing of data needs iterative refinements, model building is an iterative process of feature engineering and hyperparameter tuning, visualizing results and building applications are also iterative processes.

2. **Ship intermediate output: Even failed experiments have output** - Even incomplete assets have value to the team and the customer and should be documented and shared. All work should be shared with team members and if possible with end users, this requires regular committing to source control.

3. **Prototype experiments over implementing tasks** - In contrast to software engineering, data scientists mostly perform experiments instead of having fixed tasks. They need to iteratively experiment to generate insights.

4. **Integrate the tyrannical opinion of data in product management** - The perspective of data is as important as the perspective of customers, developers and business stakeholders. It has to be included in every discussion and must be grounded via exploratory analysis and visualizations. The data decides what is possible and what is easy or hard.

5. **Discover and pursue the critical path to a killer product** - The team needs to understand what is most crucial for a products success and focus on this part. Discovering the correct goal requires experimentation. Goals might shift and require continuous refinements. The team leader should guide the team towards the correct goal and the path of success.

6. **Get meta. Describe the process, not just the end state** - Due to the research nature of data science, there is not always something complete to shift at the end of a sprint. However, the process of obtaining insights can always be documented and shared and presented.

7. **Climb up and down the data-value pyramid**

Life cycle, processes and tasks

The author describes the data-value pyramid, a five-layer pyramid consisting of tasks in a data science project. Each level from bottom to top describes the increasing value generated while transforming data from its raw states to actions. The methodology does not contain phases.

Layer 1: Records The record layer describes obtaining data sets from various sources and making it available to an application. It also encompasses displaying data and plumbing individual records.

Layer 2: Charts and Tables The second layer, charts and tables, describes the refinement and preliminary analysis of the data. Here the data is cleaned, aggregated, visualized and questions about the data are formed.

Layer 3: Reports This layer describes the exploratory analysis of data. Data is now structured, linked, metadata and tags are added and the team explores and interacts with the data and shares insights.

Layer 4: Predictions In the fourth layer value is generated of the data by building models which enable recommendations, understanding, inference and learning.

Layer 5: Actions The last layer describes the generation of new actions or the Improvement of existing actions based on the generated insights.

The data-value pyramid does not have to be followed step-by-step and in an particular order. Teams can move up and down layers. However, skipping steps in the pyramid can lead to technical debt.

Roles and responsibilities

Jurney argues that embedding data science teams in organizations requires adjusting expectations, as these teams will not deliver predetermined outcomes but rather descriptions of applied research processes. The benefits of a clear shipping date get lost in favor of transparency of the work, its processes and results and a continuous progress report.

Since data science comprises various disciplines such as analysis, design, development, business and research, Jurney suggests that teams require a broad set of team members. In particular, he describes 13 roles and provides descriptions of their responsibilities. The 13 roles involved a project according to him are the customer, business development, marketers, product manager, user experience designer, interaction designer, web developer, engineers, data scientists, applied researchers, platform or data engineers, quality assurance engineers and operations/DevOps engineers.

Jurney argues that generalists, who can fulfill multiple roles, should be preferred over specialists, who specialize only in one particular role. Figure A.10 shows an example of team members acting in up to three different roles in a project. Jurney argues that having too many people and thus larger

teams involved in product development, leads to communication overhead and waterfall-like projects. He emphasizes that team members should regularly and iteratively share their intermediate progress with each other.

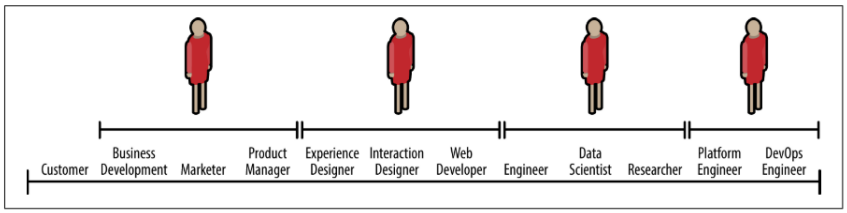

Figure A.10: An exemplary depiction of team members filling multiple roles, taken from [6].

Artifacts

There is no explicit mention of artifacts.

Tools and techniques

Jurney dedicates a whole chapter of his book to tools which should be used in an agile setting. Amongst other, he recommends using JSON, Apache Parquet, Apache Kafka, S3 storages, Mongo DB and Flask.

He also advertises code review and pair programming as techniques for collaboration. Both help to increase quality and avoid problems when key stakeholders are on sick leave, because knowledge about code is spread over multiple stakeholders.

A.1.13 Agile delivery framework for BI, fast analytics and data science

In 2016 Larson et al. analyzed how agile principles could be applied to business intelligence, fast analytics, i.e. visual analytics, and data science methodologies [117]. They take agile principles and check how well they fit to the different fields. The principles are:

1. individuals and interactions over processes and tools
2. working software over comprehensive documentation
3. customer collaboration over contract negotiation
4. responding to change over following a plan

We will restrict our summary of the methodology to its data science parts and omit the business intelligence and fast analytics (visual analytics) parts.

According to the authors agile methodologies have certain properties which are useful for data science. Agile in the context of data science yields methodologies which are less formal, more dynamic, focused on customer needs and focused on the utility of information rather than software development. Agile is a good fit in big data contexts as it relies less on formal requirement specifications and more

on rapid progress and iterative discovery and validation. Larson et al. explain that the aforementioned principles have some consequences for the methodologies. Experienced team members will work together better than inexperienced team members which follow structured processes and tools. While documentation is very valuable, it should not consume too much time of the development process, as the system is subject to rapid changes. Documentation should hence be easily consumable and maintainable. Communication with customers should be ongoing to validate results, development directions and to manage expectations. Agile teams should further be prepared for frequent changes and be able to respond quickly.

Following the agile principles data science projects should be performed in an iterative and incremental manner. Deliverables, such as code, models, visualizations or other artifacts, are best delivered in increments to account for changing businesses and technologies and to allow for better risk management. As part of a roadmap deliverables should be defined and scheduled. The increments define the scope of deliverables. The iterations on the other hand are used to refine the quality of deliverables.

Life cycle, processes and tasks

The lifecycle suggested by the authors contains 7 phases. The authors define data products as result of data science projects. Examples for data products are prediction engines, classifiers or recommendation engines.

Scope phase: In the scope phase, a problem statement is created and the scope of data sources is defined to obtain data quickly for analysis.

Data Acquisition and Discovery phase: In the data acquisition and discovery phase data is acquired, possibly without fully understanding it yet. To support this process data can be visualized to get a first understanding of the various data sources.

Analyze/Visualize phase: In this phase the data is analyzed in an iterative manner. As part of an exploratory analyses visualization techniques can be used. In addition to visual analysis, this phase includes data profiling, values analysis and structural analysis.

Model/Design/Develop phase The model/design/development phase consists of analytical modeling for descriptive, predictive or prescriptive analysis. In this phase, models are fitted to the data. This phase is performed in timeboxed iterations and increments.

Validate In the validation phase, the model is validated with the aim of minimizing its error. This again, is an iterative process and closely tied to the Model/Design/Develop and Analyze/Visualize phases.

Deployment Lastly the deployment phase begins in which the model is deployed in the production environment. Support is given by end users, whose feedback leads to revisions.

The lifecycle models iterative loops within the Analyze/Visualize phase, the Deployment phase and cross-phase loops between the Analyze/Visualize, Model/Design/Develop and Validate phase. Throughout the whole project, business and IT stakeholders should closely collaborate.

Roles and responsibilities

The methodology mentions some roles in different phases but does not provide details on their responsibilities and tasks.

1. Business users: determine information requirements and business questions together with IT professionals

2. IT professionals: determine information requirements and business questions together with Business users

3. End-users: Apply working system to analyze actionable information, give feedback after deployment.

Artifacts

No artifacts are defined in the methodology

Tools and techniques

The only tools mentioned in the methodology are R and Hadoop.

A.1.14 MIDST

Introduced in 2019 by Saltz et al. [118] MIDST is technically not a project methodology but rather a set of tools that aims at optimizing analytics projects and supporting methodologies. The tool is build on the conceptual framework of stigmergic coordination, i.e. coordination in which project members provide stimuli to others to build upon and continue their work. Key components of the tool are the visibility of work for all project members, genres of documents to quickly understand the flow of work and combinability of work.

The framework is inspired by free/libre open source software (FLOSS) development. The authors suggest that FLOSS projects show great task coordination with minimal communication overhead between the team members. According to the authors data science projects are lacking guidance on collaboration. The key challenges of task coordination are to divide projects into manageable pieces, assigning the work to project members, tracking the progress, dealing with interdependencies and finally integrating everything. To the surprise of the authors, teams in FLOSS seem to overcome these challenges with minimal communication, mainly via code-centric collaboration.

They advertise combinability of the work of data scientists. Ideally, work is combinable and improvable in modular increments. The authors encourage small, focused commits of work to assure combinability and to avoid large changes which are hard to understand. Modularity of a solution facilitates combinability and task-modularity reduces the complexity of problem solving by allowing team members to focus on small changes rather than the entire solution. This also reduces coordination efforts.

Life Cycle, Tasks and Processes

MIDST does not define a life cycle or specific tasks and processes but rather gives recommendations on how to collaborate. It advertises information sharing across all project members and incentivizing code-centric collaboration as in FLOSS projects.

Roles and responsibilities

MIDST does not provide role descriptions.

Artifacts

Saltz et al. attribute a central role of collaboration to documents. In data science projects such documents are often notebooks, containing code, documentation and results and visualizations. The documents can not only be used to perform the work but also to track its progress.

A key feature of documents is their visibility. Making work visible to others and using transparent systems that detail processes and functions of an organization supports information exchange and communication. Aside of current work also dependencies with other projects should be made visible as well as revision histories of work. To assure the visibility and understandability of their work, data scientists should check their work in early and frequently.

The authors emphasize that it is helpful to have genres of documents, i.e. different types and templates of documents. This eases understandability for the project team. A genre system in data science could either follow a kanban-like structure or the typical phases of data science projects.

Tools and techniques

MIDST itself is a tool, which provides functionality according to the aforementioned aspects of data science projects such as more collaboration, document genres, document sharing, code sharing, etc. It is a web-based frontend which works with scripts from the R programming language.

A.1.15 Analytics Canvas

In 2018 Kühn et al. introduced the Analytics Canvas [119]. While the Analytics Canvas does not describe a complete project methodology it provides a semi-formal specification technique to capture analytics use cases, the necessary company-wide data infrastructure and requirements for interdisciplinary domains. The authors aim to tackle the problem of communication and joint understanding in multi-stakeholder settings, where the stakeholders have different backgrounds, e.g. IT, business and analytics. They list communication and enterprise-wide digitization strategies as keys to success for companies and lack of collaboration across disciplines and lack of consistent methods and processes as most important pain points in analytics projects.

The authors criticize CRISP-DM for not explicitly addressing the primary strategic goals of an analytics project because they consider a goal-oriented development process very important. For their Analytics Canvas they follow the four types of analytics as described by Steenstrup et al. [223], which are Descriptive, Diagnostic, Predictive and Prescriptive Analytics. In the model Steenstrup et al. specify for each analytic category how much human input is needed and how sophisticated the analytics program shall be.

Life Cycle, Tasks and Processes

In the four-layer model for describing analytics use cases, the four categories by Steenstrup et al. are used to describe the data analysis type. The four-layer model starts with the understanding of the domain which yields an analytics use case. Typically, higher management is involved in this step.

Next, it is important to understand the data, including the generation of a data definition and the data sources to use. This step involves domain experts. Third, the data is acquired, stored and preprocessed with the help of IT experts. Lastly, the data is analyzed and a solution for the use case is implemented by the data scientists.

The data analytics canvas extends this model with an additional data description layer. It also introduces 9 ”constructs”: Production resources, analog data storages, IT systems, analog and digital documents, descriptions of data-types, data pools and analytics use cases. These constructs are either available or missing in the layers and can be linked via connections. Together these elements form the analytics canvas.

Roles and responsibilities

The Analytics Canvas describes five different types of roles, management, data scientists, IT experts, analytics architects and domain experts. They are involved in different layers of the model.

1. Management: Mainly involved in domain understanding
2. Data Scientist: Responsible for the data analysis
3. IT Expert: Mainly involved in storing data
4. Analytics Architect
5. : Domain Expert: Mainly involved in deciding which kind of data sources to use

Artifacts

The filled out Analytics Canvas itself is an artifact which contains multiple information. It lists information about:

1. Type of Data Analysis
2. Available Data Pools
3. Data Descriptions
4. Data Sources

Examples of filled in analytics canvas instances can be seen in figure A.11 and A.12.

Tools and techniques

The analytics canvas itself is a specification technique, which provides useful information to all stakeholders in order for them to gain a joint understanding of analytics use cases.

A.1.16 Data Science Workflow

In 2012 Guo introduced the research programming workflow [224], which he later renamed to data science workflow [121]. To design the workflow he summarized his own experience, interviews, a survey and case studies on research programming.

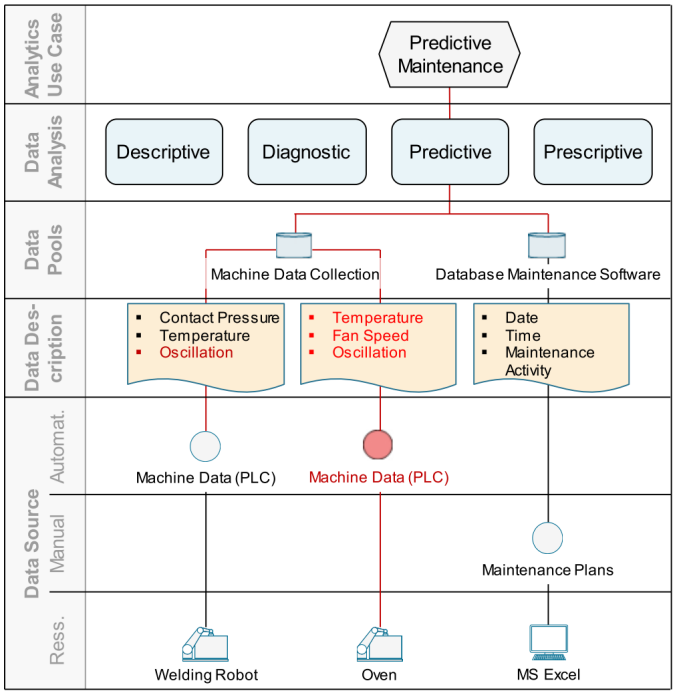

Figure A.11: An instance of the Analytics Canvas for a predictive maintenance use case, taken from [119].

Life Cycle, Processes and Tasks

The workflow Guo observed, depicted in figure A.13, consists of four main phases: preparation of data, analysis, reflection and dissemination. It centers around data and modeling issues and gives a good impression of the challenges that data scientists face.Fa

Preparation phase Before the actual analysis can begin, the data scientists need to acquire the appropriate data and reformat it to fit the desired analysis. Hence, the phase starts with a data acquisition step. Gao suggests, that the data scientists can access multiple internal and external data sources, such as online repositories, logs, sensors, etc. He identifies data provenance as one of the main problems in this phase and emphasizes the importance of storing the source of the data and the time it was obtained to ensure reproducibility and enable tracking of downstream errors. A related issue is data management, i.e. how to name, organize and version files and keeping track of differences. Lastly, storage might be an issue if datasets are particularly large. After acquiring the data, it needs to be reformatted and cleaned to fit certain analytical models. Cleaning encompasses tasks such as handling missing values, formatting errors, etc. According to his insights this is considered to be the most tedious and time-consuming part of the work in data science. However, Gao emphasizes that the data scientists can already gain knowledge about the data and the validity of certain hypothesis and the appropriateness of models in this phase. Lastly, data integration from different sources might pose a challenge.

Analysis Phase The analysis phase of the workflow is described as the core activity. It consists of writing, executing and refining data analysis programs. More precisely, Gao identifies a repeated iteration cycle of editing, executing, inspecting to gain insights, debugging and re-editing of scripts and the results generated by them. He further identifies three factors which slow-down the process:

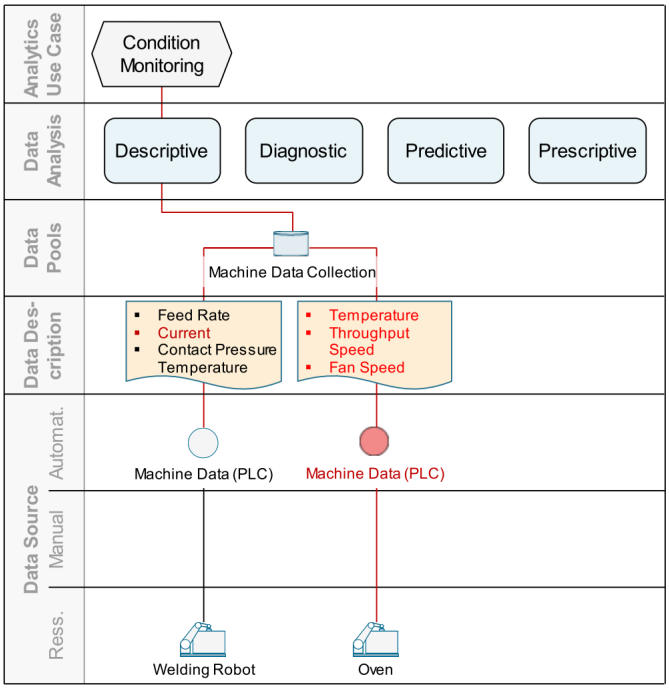

Figure A.12: An instance of the Analytics Canvas for a descriptive analysis in a condition monitoring setting, taken from [119].

the absolute running time of the scripts, the incremental running time by modifying and rerunning scripts and crashes from errors. Gao notes that data scientists often refer to external resources such as websites to improve their code iterations.

Reflection Phase After executing the analysis, a reflection phase starts in which the data scientists, think about their results and communicate them. Reflection can be done in multiple ways such as note taking, meeting and discussions with colleagues or making comparisons and exploring alternatives. Gao also suggests that this is supported by visualizations of the results.

Dissemination phase The workflow ends with a dissemination phase in which results are either disseminated as written reports, memos, presentations, white papers or academic papers. A major challenge for the dissemination is the aggregation of all relevant information from sources such as notes, outputs, emails, etc. Data scientists might additionally want to share their software with colleagues to enable reproduction or the usage in different use cases. They might also want to share intermediate results to get feedback and ideas by their peers. Lastly, Gao points out that in order to ensure reproducibility the state of a system at a given time should be preserved.

Roles and responsibilities

The methodology does only account for the role of a research programmer.

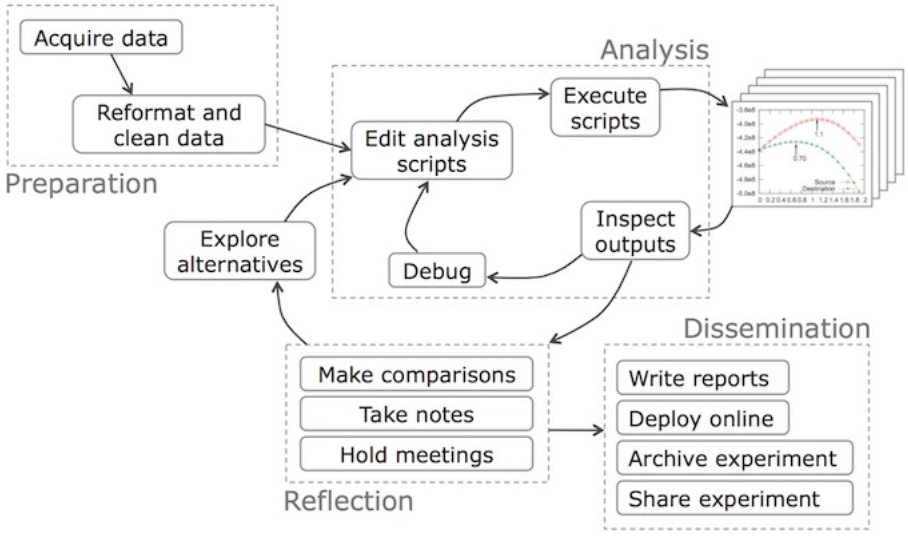

Figure A.13: The data science workflow by Guo [121].

Artifacts

The methodology does not provide specific definitions of artifacts. Artifacts mentioned by the author are:

1. Raw Data including metadata (provenance)
2. Cleaned and reformatted data
3. Scripts
4. Notes
5. Visualizatons and status reports
6. to-do lists
7. written reports
8. documentation
9. packaging

Tools and techniques

To complement the workflow, Gao also developed tools for support. The first tool is the ”Proactive Wrangler”, which helps the research programmer with data cleaning and reformatting and offers suggestions as to which actions are useful. Data transformations can be exported directly as python scripts. ”IncPy” is a tool which supports the research programmer with data analysis scripting, code

management and management of data dependencies. It also facilitates the naming of intermediate files which are produced during the life cycle. ”SlopPy” prevents data analytics workflows from crashing to speed up the process of data analysis, while simultaneously logging the occurring errors and providing useful information for debugging Burrito combines notetaking with experiment tracking, to better organize tasks and experiments. Finally, ”CDE” helps the research programmer to deploy, archive and share prototypical software with less overhead than with typical VM deployments.

A.1.17 SEMMA

SEMMA is an acronym for Sample, Explore, Modify, Model and Assess. It was created by the SAS Institute and is incorporated in the SAS Enterprise Miner product. The understanding whether SEMMA is a data mining methodology or not varies. In 2006 the SAS Institute explicitly stated that it is “a common misunderstanding [...] to refer to SEMMA as a data mining methodology. SEMMA is not data mining methodology but rather a logical organization of the functional tool set of SAS Enterprise Miner for carrying out the core tasks of data mining” [120]. However, today they refer to it as a data mining process and data mining methodology [225] and it is often referenced as data mining methodology in literature [94] [226] [227] [228]. Due to its focus on data preprocessing and modeling, it could be considered a model development methodology. Nevertheless, we want to shortly summarize the concepts behind SEMMA as it overlaps with the data understanding, preparation and modeling phases of the CRISP-DM methodology and can provide useful insights for a natural language understanding methodology. SEMMA does not provide roles and artifacts and only a rudimentary life cycle description.

Life Cycle, Tasks and Processes

SEMMA consists of five phases. Although it also mentions deployment explicitly, it does not mcontain a deployment phase. The life cycle SEMMA proposes is more or less sequential, with a back link from the assessment phase to the sampling phase. However, SAS mentions that multiple iterations per phase can be necessary before achieving desirable results. They also acknowledge that some of the tasks might not need to be performed at all. The phases and tasks SAS proposes are visualized in figure A.14.

Phase 1: Sample In the sample phase, data is subsampled to obtain a smaller but still significant sample of the data. The reasoning behind this step is to lower complexity and runtime but still assure that statistically representative patterns are present in the data. SAS also recommends creating training, validation and tests splits of data to prevent overfitting the model and to obtain an assessment of the generalization capabilities of the models.

Phase 2: Explore In the second phase the data is explored visually and with statistical methods such as clustering and anomaly detection. The goal is to generate a better understanding of the data. Insights generated with this exploration allow better modeling in the later phases.

Phase 3: Modify In this phase the data is modified to best suit the data mining process. This includes steps such as feature augmentation, reduction, transformation, outlier detection, label space reduction or selection of specific clusters or subgroups. This phase should also account for the change of data over time.

Phase 4: Model Appropriate models are chosen to answer business questions or discover patterns in data. These models are trained and optimized based on the training and validation data. SAS explains

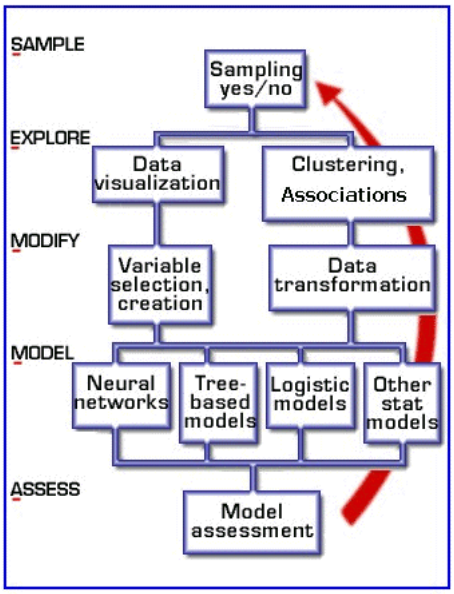

Figure A.14: The SEMMA methodology as presented in [225].

that the choice of model is dependent on the task that should be solved and might require exploration.

Phase 5: Assess In the final phase the model from the prior phase is assessed on held-out data. The goal is to get an estimate of the model performance, usefulness and reliability. If there are particular data samples for which it is already known how a model should behave, they can be taken into account for evaluation.

Deployment Although model deployment is not part of the SEMMA phases it is explicitly mentioned to be the outcome of modeling. The authors argue that this is the part where value is generated based on the data mining results. The tooling that SAS provides yields deployable code for the model itself as well as the processing of the data. The authors claim that standardizing the production of deployment code enables data miners to focus more on the modeling tasks and safe time.

Roles and Responsibilities

SEMMA does not define any specific roles other than the user who applies the SEMMA process.

Tools and Techniques

SEMMA is embedded in the SAS Enterprise Miner. It provides a front-end for its users and tools to perform the different steps of the SEMMA process as well as guidelines how to apply them.

Artifacts

SEMMA does not specify particular artifacts but provides methods to generate outputs and logs for its tasks in the Enterprise Miner.

A.1.18 Microsoft TDSP

In 2016 Microsoft introduced their own methodology for data science projects called the Microsoft Team Data Science Process [7]. Their aim was to create a process which is agile, iterative and helps to deliver intelligent applications efficiently. The methodology emphasizes team collaboration and communication between roles in a team.

Microsoft names four key components of their methodology:

1. A data science lifecycle definition
2. A standardized project structure
3. Definition of infrastructure and resources
4. Tooling and utilities

Life Cycle, Tasks and Processes

The data science lifecycle of TDSP is designed in a way that allows it to be combined or used complementary with other methodologies such as CRISP-DM A.1.2 or KDD. Similar to these methodologies, TDSP has an iterative nature. While TDSP focuses on projects which aim for the deployment of models, it can also be applied to exploratory projects. Its goal is to move data-science projects toward a clear engagement end point. Each of the phases of the process contain a goal definition, steps for fulfilment and artifacts which should be created during the phase. The complete lifecycle is depicted in figure A.15.

Phase 1: Business Understanding The main goals of the business understanding phase are to identify key variables which serve as model targets, metrics which can be used to determine the quality and success of the models and to identify relevant data sources for the project.

In order to achieve these goals multiple steps are taken. First the **project objectives** are defined. Here, the data science teams collaborate with the customer and other relevant stakeholders to understand and identify the business problems which should be tackled. Project goals are defined by asking the stakeholders precise questions, e.g. to determine the nature of the machine learning problem. The project team is defined by specifying roles and responsibilities and a high-level milestone plan is formulated jointly and iteratively refined during the project. Finally, SMART metrics are defined. Second, relevant data sources are identified. Relevant data is data which helps the data scientists to answer the questions that define the project objective. Data needs to be relevant to the business question and an accurate measure of the models target variable and features of interest. It might be necessary to acquire additional data from external data sources.

Phase 2: Data acquisition and understanding The goals of this phase are to produce a clean, high-quality dataset, whose relation to the target variable is suitable and well-understood, move the dataset to the target environment and build a solution architecture of the data pipeline.

First, the data is ingested into the target environment, running the analytics operations like training and prediction. Then the dataset is cleaned of noise, missing values and discrepancies. Data summarization and visualization techniques are used here to audit the quality of the data. Next, a deeper understanding of the data is built to understand patterns, obtain relevant information for model selection, assess the connection of the data and the target variables and finally decide whether more data is needed. TDSP stresses that this process is iterative. Finally, a data pipeline is set up which

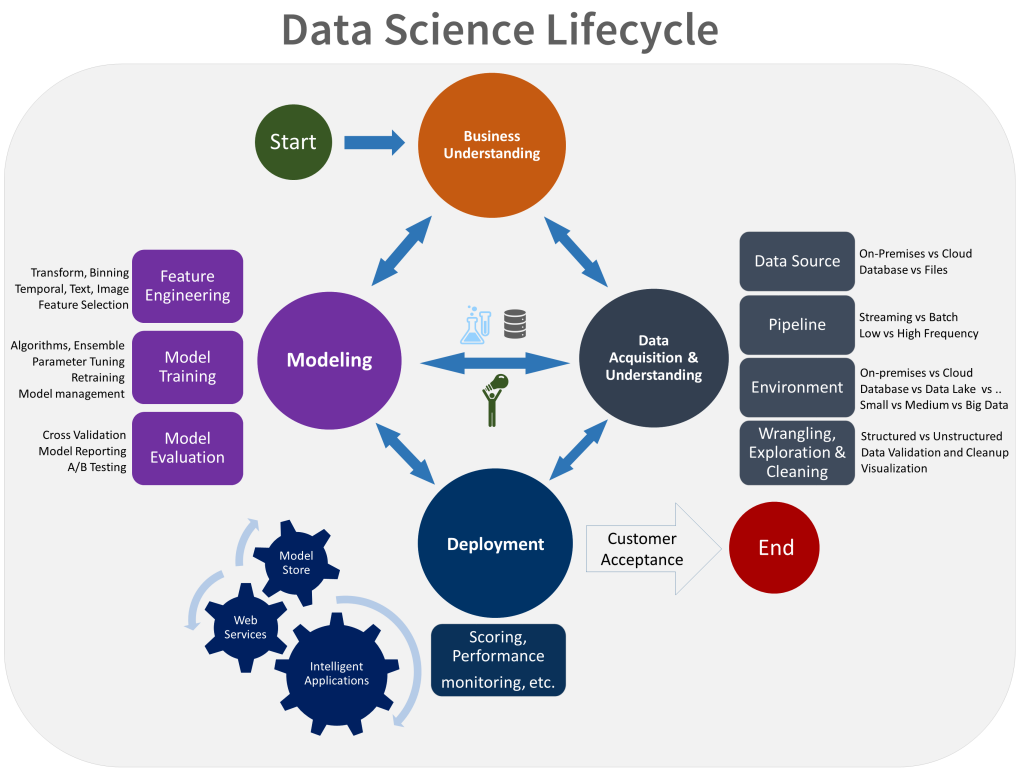

Figure A.15: The Microsoft TDSP lifecycle [7]

handles ingestion, cleaning and scoring of data. Depending on the project requirements this pipeline is either built in a batch-based, streaming-based or hybrid fashion.

Phase 3: Modeling

In the modeling phase optimal data features are determined, an informative machine learning model that predicts the target variable most accurately is selected and a machine learning model suitable for production is created.

According to TDSP, the Modeling phase starts with feature engineering. Features are created from raw data, which involves the inclusion, aggregation and transformation of raw variables. Domain expertise and insights from data exploration are used to find and include relevant variables as well as to exclude irrelevant variables. After finishing the feature engineering, models are trained and compared to success metrics. The model training follows a standard machine learning process of splitting the data into a train and test set, building models, tuning and evaluating them as well as determining the best solution. The model evaluation step is used to make a checkpoint decision: Does the model provide sufficient quality and confidence for production on the test data, are alternative approaches more suitable or does the team need more/other data or features? Finally methods for model interpretability are used to explain the model behavior in general or individual predictions via visualizations and the fairness of the model is assessed.

Phase 4: Deployment

After finishing modeling, the models are deployed to a production(-like) environment together with a data pipeline in order to verify user acceptance.

Models are operationalized in a way that allows other applications to consume them. To this end open APIs are specified. Additionally, telemetry and monitoring methods are build into the production model and the data pipeline.

Phase 5: Customer Acceptance The last phase of TDSP is the customer acceptance phase. Here, the goal is to finalize project deliverables and to confirm that the data pipeline, model and deployment in the production environment satisfy the customer's objective.

In this phase the system is validated by customer tests, verifying that the deployed model and pipeline meet the customer needs. The project is handed off to the customer or the entity which runs the system in production and it is additionally validated that the business requirements are met and questions are answered with acceptable accuracy. Finally, all documentation is finalized and reviewed.

Roles and responsibilities

TDSP defines the roles of a group lead, team lead, project manager, solution architect, data engineer, data scientist, application developer and project lead. It includes detailed description for the tasks of the different roles and their responsibilities. TDSP assumes that data science teams are embedded in data science groups and that the data science teams are responsible for project execution. The tasks which are defined for the different roles range from the setup of the project to the final project deliveries. TDSP emphasizes the creation of standardized project structures which are utilized across projects which benefit the team communication, information sharing and lower the barriers for new project members.

Artifacts

A key element of TDSP is the creation of standardized project structures. According to TDSP all projects should share the same directory structure and use templates for project documents. Code and documents should ideally be stored in version control systems and tracking should be done via agile project tracking systems. TDSP recommends using a fixed set of tools and utilities for data storage and project execution to increase consistency across projects and lower boundaries. TDSP also suggests using shared data stores, to store raw and processed datasets. This way data duplication is avoided, which prevents inconsistencies and infrastructure costs and analysis become reproducible for every team member.

TDSP describes various artifacts which are generated over the course of the project, they indicate which artifact is generated in which phase:

- **Business Understanding**: Charter document, specification of data sources, data dictionaries
- **Data Acquisition and Understanding**: Data quality report, solution architecture, checkpoint decision
- **Modeling**: Models, features
- **Deployment**: Status dashboard, final modeling report, final solution architecture
- **Customer Acceptance**: Exit report

Tools and techniques

The whole methodology is tailored towards the Microsoft Azure tool stack and infrastructure.

A.1.19 Data Analytics Lifecycle

The data analytics lifecycle (DAL) is mentioned in a patent, which describes a process for data analytics projects as well as the according tools for the automation of parts of the lifecycle, especially tailored towards cloud settings and infrastructure [122]. It was published in 2016. The motivation behind DAL is that data analytics projects are becoming more and more complex due to increasing dataset sizes and more varying structure of datasets. According to the authors this leads to increasing uncertainty in project cost estimation, inflexible solutions and inadequate, costly solutions. They argue that this calls for improved data analytics techniques for business users and data scientists. DAL tackles various scopes of projects such as research projects, data mining projects, services building and other forms of projects which tackle business projects. DAL structures these projects into six typical phases, which are interconnected in an iterative manner. It describes multiple artifacts which are generated in the lifecycle of an analytics project and describes the roles of a data analyst and business user.

Life cycle, processes and tasks

Each phase of DAL is divided into multiple steps which are not necessary sequential and often times can be tackled in arbitrary order or even in parallel. The lifecycle is depicted in figure A.16.

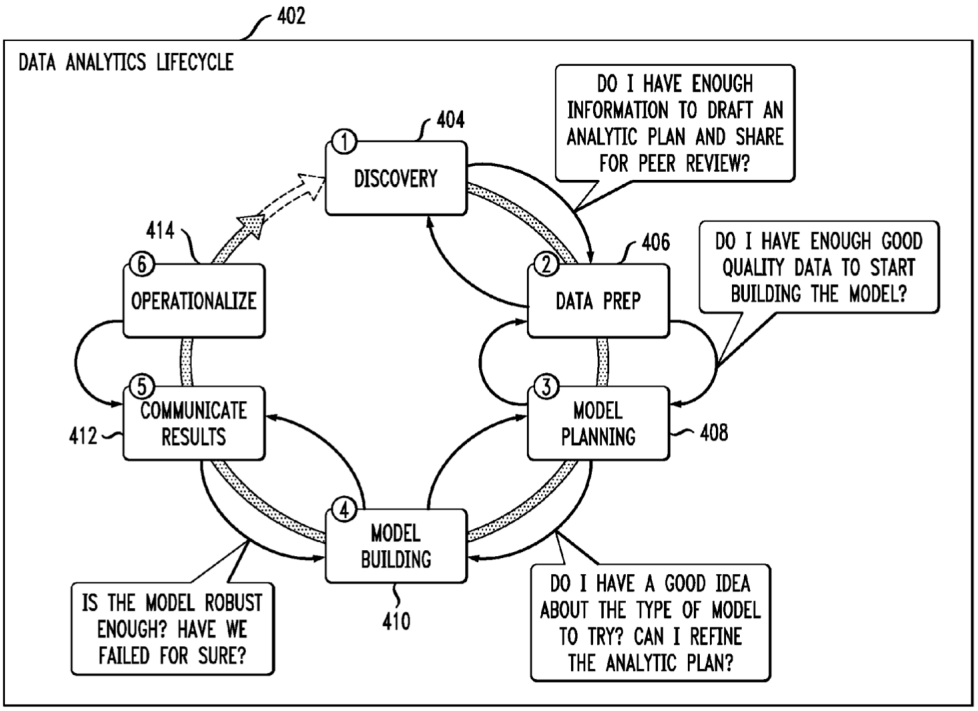

Figure A.16: Depiction of the data analytics lifecycle in [122] including references to accompanying modules to automate the process and questions which are typical for certain phase transitions.

Discovery Phase The goal of the discovery phase is to create an artifact called the initial analytic plan. This plan describes the business problems, problem categories, hypotheses, potential algorithms/models to use and the data. The initial data analytics plan will be shared with the other project members at the end of the discovery phase, however it is subject to change based on the following phases. The data discovery phase consists of four steps. The first step is the determination of the problem category based on analytical methods, i.e. matching the problem to methods such as clustering, classification, text mining and so on. The next step is the assessment of resources for problem solving. Resources include people, process technologies, data sources, systems and time. Third is the framing of the problem, such that it can be tested with data and analytical methods. Lastly, available data sources are identified and a first assessment of the data is performed. This includes previewing of data, providing high-level understanding of the data, reviewing it and determining its structure. The last step also includes a summary of data needed for the kinds of problem in the project. All of these steps amount in the data analytic plan.

Data Preparation Phase Goal of the data preparation phase is to transform the data, such that it is in acceptable shape, structure and quality for subsequent analysis steps. The authors suggest setting up a sandbox environment to experiment with the data. To begin with, data analysts should familiarize with the data. This includes listing all data sources, checking whether key data is available and appropriate for the problems to solve. After the initial familiarization, data conditioning is performed. Data is cleaned and normalized and unnecessary data is discarded. Lastly the data scientist should perform a survey of the data and visualize it if possible. Typical parts of the survey include descriptive statistics and assessments of the data quality.

Model Planning Phase In the model planning phase, the data analyst converts business problems into a data definition and potential analytic approaches. This is strongly linked to the data analytic plan and overlaps with the data preparation phase. It is also possible that multiple model planning iterations are necessary, as multiple models can potentially be suitable to solve the business problems. The data structure is assessed again in this phase, to determine whether it is suitable for the different algorithmic approaches. Depending on the data structure, i.e. unstructured data such as text or structured data, appropriate tools and techniques are determined. The data scientist or a business user should also make sure that the analytical techniques are appropriate to meet the business objectives and to prove or disprove hypotheses about the data. After the initial data exploration in the data preparation phase, the data is now further explored with respect to the variables and their relationships. Expert knowledge should be utilized to select appropriate variables and methods. The selection of the variables requires knowledge of the models to be used and could be enhanced with dimensionality reduction. The authors encourage further data visualization in this phase. Finally, the selected models should be converted to in-memory applications.

Model Building Phase The model building phase presents a central, albeit rather short phase. Data is split into a training, test and production dataset and the model is fit on the training data. It might make sense to start with a subset of the training data, to test the training process. This phase is closely interconnected with the model planning phase and might require multiple iterations. Due to often very resource demanding building of workflows and models, the authors advise to make the effort to build an appropriate environment, considering hardware and software requirements.

Results Communication Phase After finishing the model building, the data scientist should review and interpret the results and create the final analytic plan. The interpretation of the results includes the comparison with the initial hypothesis and the assessment of the business value, e.g. in time or money saved. Key findings and major insights should be documented and presented.

Operationalizing Phase Finally, the analytics project is deployed on a production system. Before full-scale execution it is advised to make prior tests. These tests can yield valuable insights into run times and model quality and can help to set up ongoing monitoring and model alerts, which indicate that the system is not performing as intended, perhaps due to data shift and requires retraining. The authors call these prior tests a pilot program.

Roles and responsibilities

The data analytics cycle mentions eight roles and does not specifically define them, however some tasks for each role are mentioned in the methodology. A key part of the methodology is that a data scientist or business user is guided through the project life cycle by software modules.

1. Data Scientist: develops initial analytics plan, identifies business problem, determines problem category, assesses available resources, identifies data sources, aggregate sources, preview the data, provide high-level understanding, review the data, determine needed structures and tools, scope the data needed for the problem, shares data analytics plan with team, assesses suitability of dataset for problem solving, clean and normalize data, examines distribution of data, evaluates consistency over all data, generates hypotheses, describes potential data sets, chooses potential models, assess data structure to determine tooling (e.g. unstructured vs structured), assesses models w.r.t. robustness, develops data sets for testing, training and production, interprets results, assesses results, documents key findings, communicates results
2. Business User: determines problem category, assesses available resources, identifies data sources, aggregate sources, preview the data, provide high-level understanding, review the data, determine needed structures and tools, scope the data needed for the problem, assess data structure to determine tooling, develops data sets for testing, training and production, interprets results, assesses results, documents key findings, communicates results
3. Domain Expert: validates ideas of data scientists/business users, provide knowledge for variable selection
4. Business Stakeholder: provide knowledge for variable selection, need to understand how model affects processes
5. Data Analyst: filter out irrelevant data
6. Production Engineers: operationalize work, assure that solution works in production environment, assess run times, assess model w.r.t. goals and expectations
7. Database Administrator: filter out irrelevant data
8. Business Sponsors: Champion projects

Artifacts

The data analytics cycle mentions different artifacts which are used, produced and altered during an analytics project. Central to the methodology is an analytics plan which provides guidance but is also adjusted based on new findings.

1. Initial data analytic plan: Created in the discovery phase, serves as basis for all of the work. Helps identifying the business problem, relevant hypothesis, relevant data, model/algorithms. Used as quality gate to end the discovery phase.
2. List of data sources
3. Data definition
4. Refined data analytic plan: A refinement of the initial data analytic plan at the end of the data preparation, model planning and model building phase
5. Final data analytic plan: Final version of the plan at the end of the results communication phase, listing key findings and major insights as well as assessments of the initial hypotheses

Tools and techniques

Dietrich et al. describe a system for the execution of the data analytics cycle with models for each of the phases, to support the data scientist along the complete project lifecycle. DAL contains references to helpful tools for the data preparation and model planning phase.

1. Data preparation phase
   a) Hadoop
   b) Alpine Miner
   c) R
   d) Data analytics sandbox: a sandbox environment to perform experiments on the data
2. Model planning phase
   a) Map/Reduce
   b) Natural language processing (NLP)
   c) ML techniques such as Clustering, Classification, Regression
   d) Graph theory

The authors also specify that the system should have a graphical user interface to assist its user. They focus on a setting in which this system is hosted on cloud infrastructure and also discuss the need for tooling to handle provisioning of resources in cloud settings. According to them an aim of the overall system should also be to aid in automating as many steps as possible of the lifecycle.

A.1.20 Development Workflows for Data Scientists

In 2017 Ciara Byrne published the Development Workflows for Data Scientists [124] in collaboration with the data science team from GitHub. She conducted interviews with data science teams from various companies to learn about their data science workflows and ways to optimize such workflows. She argues that there was an explosion of data science interest in various fields but that data science as a field has not yet matured as software development did. She explicitly asks the question what data science can learn from software development.

In her understanding the core difference between data science and software development is that data science produces insight, while software development produces code. In the data science domain, code is a way to get insights about questions and not usually intended to end in productive systems. However, wrapping the generated insights into code makes them more easily accessible. She distinguishes projects in which it is already clear what to build, from exploratory projects in which data analysis and experiments are required to define the end of a project.

Properties of a good Data Science Workflow

Byrne defines a workflow as "the definition, execution and automation of business processes toward the goal of coordinating tasks and information between people and systems". She argues that data science processes are still evolving in contrast to more mature software development processes. According to her, data science workflows should also be adapted to the specific goals of a team and a project.

She lists some best practices and guiding principles for workflows. One principle is to "produce results fast", i.e. initial results should be presented at an early phase and then more thorough analysis should be performed subsequently. If possible, steps for known data should be automated. In order to achieve this and lower the risk of repeating mistakes in the future, it is a good practice to similarly record all steps, also for negative results. For new data, the recommendation is to investigate it thoroughly and manually to derive maximal knowledge. Another principle, which ties in with this, is to ensure reproducibility and reuse of results. A good practice is to share within and across teams and to encourage reuse of code and models of other team members. Lastly, Byrne suggests to use automated workflows to evaluate compliance with regulations and model quality, if possible.

Byrne encourages knowledge sharing as done by Airbnb, via a knowledge repository. This knowledge repo should make data science work discoverable to the whole company and ideally contains methods of making the knowledge shareable or even subscribable, e.g. via topic tags.

Life cycle, processes and tasks

Since the design of a data science process is dependent on the teams that implement it, Byrne gives three examples for data science processes. One generic data science process, one specific to GitHub and one specific to BinaryEdge.

According to her, data science processes generally all consist of similar steps, namely the definition of specifications, design, code writing, code testing and reviewing, documentation, system integration, deployment to production environment for business purposes.

The Data Science Process The first general process she describes, consists of 5 phases. It starts with the data scientist asking an interesting question. In order to do this it might make sense to ask customers about previous experiences with their data and whether they tried to ask similar questions. In addition, the data scientists should become more familiar with the subject domain and involve the people who actually use the final solution or work on the subject. Product managers and application engineers should be consulted to define metrics and provide the necessary instruments for the data scientist. After finding an interesting question, the data scientists should obtain the data needed to answer the question. This process is guided by questions like, how the data was sampled, whether it is relevant and whether there are privacy issues which need consideration. After obtaining the data, the data exploration phase starts. This includes plotting the data, detecting anomalies and patterns. Based on the insights of the data exploration, the modeling phase can start. Here an appropriate model is build, fit to the data and validated. Lastly, the results are communicated and visualized to answer what

insights were gained about the question, whether the results make sense and whether there is a story behind the data. All phases are iterative and contain backloops.

The GitHub Data Science Process The GitHub data science process is more elaborate. It also contains a list of tools that the GitHub data employees typically use for each phase. The GitHub process starts with a problem definition, in which the team clarifies which problems they want to address. Next, they define success metrics with which they measure and track the progress of the project. Then, they start with the data definition and collection. GitHub uses an Insights team which collaborates with engineers to obtain data, while considering security, compliance and anonymization. They use containerization, provisioning tools and automation to optimize this phase. Then, they preprocess the data, by cleaning, scaling and normalizing it, if necessary. After preprocessing they start exploring the data, it's distributions, outliers and eventually subgroups. This phase often starts in the command line to obtain basic statistics. To structure the data for easier use, they recommend using tools which build automated folder structures according to templates. They also downsample data to quickly prototype, especially in the case of big datasets. Before building a productive model, they do model prototyping, i.e. they create exploratory models to share and discuss insights. In the model design they already consider factors such as training speed, inference speed and whether the model really serves the business user needs. At this phase it is possible that they discover the need of refining the problem definition. When finishing model prototyping, the infrastructure for running the model is build and tested. The model is then productionized by scaling it appropriately and testing it on the production system. APIs are constructed to serve the transformed data and model outputs. Evaluation is a continuous task, which does not only include the assessment of the model quality but also feedback to the model. Testing machine learning models deviates from testing in software development, but is just as important. They recommend to not only test prediction accuracy but also the user experience. The whole suggested process is iterative. The process is depicted in figure A.18.

The Real-Life Data Science Development Workflow The Last Data Science Process that Byrne presents is by BinaryEdge. It contains many steps and generated artifacts. The process can be seen in figure A.18. We will just focus on important additions to the two former processes. In the phase of preliminary data analysis BinaryEdge produces a preliminary data report and collects questions and points of interest about the data. They add rigorous reporting to the exploratory data analysis and document all steps, results, conclusions, problems and questions that occur during the exploratory analysis of the data. BinaryEdge distinguishes data visualization and storytelling tasks from machine learning tasks in their process. They also add the task of reporting findings, e.g. in blog posts.

How to improve workflows Lastly, Byrne gives suggestions as to how to improve data science workflows. She states that collaboration and knowledge sharing should be emphasized and supported with standard formats, collaboration tools and ways to make previous work discoverable. She encourages data scientists to learn from the long established best practices of software developers, but stresses that the tools and workflows of software development need adaption to suit data science use cases.

Roles and responsibilities

Byrne argues that a good workflow requires clear team roles. In the core of the workflow she sees data scientists, machine learning engineers and data engineers. According to her, these roles are not fixed but rather represent a continuum, i.e. people in one role can also help with tasks of other roles and she encourages collaboration across roles. She also notes that many specialized roles arise, which are

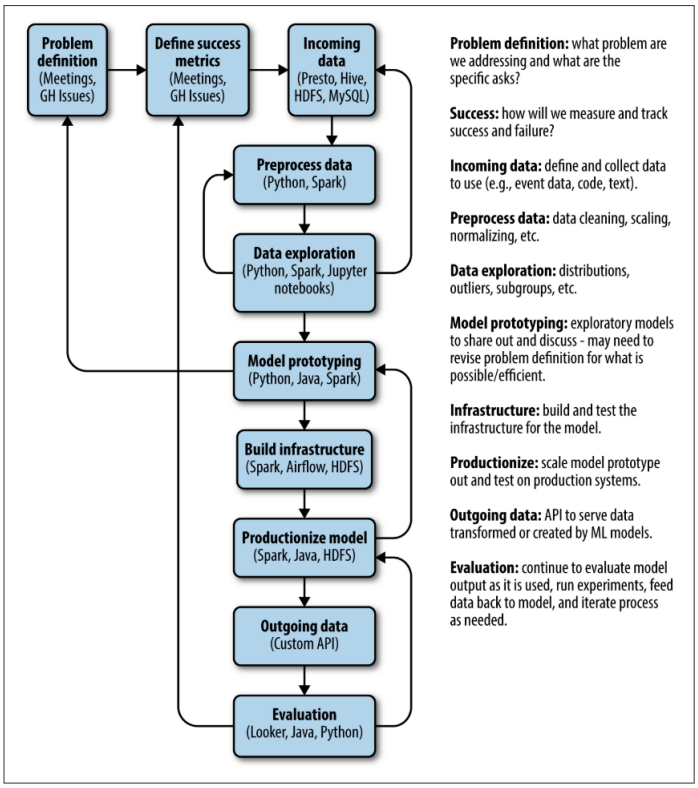

Figure A.17: The GitHub data science workflow [124].

specific to certain subfields of data science such as e.g. NLP. She does not provide descriptions for all the roles she mentions.

1. Data scientists: explore the data and implement MVP versions of functions, features and products
2. Machine learning engineers: responsible for the performance and scaling up of the solutions
3. Data engineers: concerned with providing and designing tooling and infrastructure for the solution and the data scientists
4. AI product manager: translates business requirements into a test set
5. Scrum Master
6. Designer
7. Software Developers
8. DevOps
9. Auditor

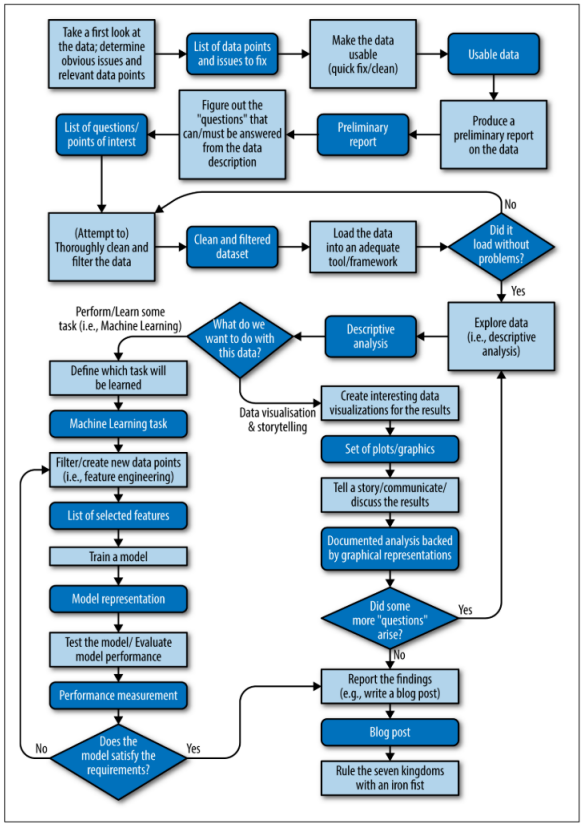

Figure A.18: The Real data science workflow [124].

Artifacts

The BinaryEdge process includes some artifacts:

1. Preliminary data report
2. lists of questions and points of interest about data
3. Data exploration report
4. Documentation of steps, results, conclusions, problems and questions

Tools and techniques

While Byrne does not explicitly state which tools and techniques are useful, the GitHub data science workflow contains useful tools and techniques for every step of the workflow.

1. Meetings: Problem definition, definition of success metrics
2. GitHub issues: Problem definition, definition of success metrics
3. Presto: during definition and collection of data

4. Hive: during definition and collection of data
5. MySQL: during definition and collection of data
6. HDFS: during definition and collection of data, building and testing of infrastructure, productionizing models
7. Python: preprocessing data, data exploration, protoype modeling, evaluation
8. Spark: preprocessing data, data exploration, prototype modeling, infrastructure building, productionizing models
9. Jupyter Notebooks: Data exploration
10. Java: Model prototyping, productionizing models, evaluation
11. Airflow: building infrastructure
12. Custom API: to serve data
13. Locker: Evaluation

A.1.21 Domino DS lifecycle

The Domino DS lifecycle was published as a whitepaper by Domino Labs [123]. It uses insights from data science teams and claims to be based on more practical realities than CRISP-DM. The authors argue that data science project management is still very immature, but very much needed by the industry. The lifecycle is built around challenges which were observed in projects from data science teams and a root cause analysis as to what caused these challenges. Key root causes which were identified are insufficient stakeholder management, limited culture of introspection, staffing problems, no culture of iteration and delivery, disconnection from business problems, lack of reusability, sub-optimal organization and tooling.

The authors propose three core principles:

1. Expect and embrace iteration while not letting iterations delay projects.
2. Enable compounding collaboration by incentivizing reusability of components.
3. Anticipate auditability needs and preserve all relevant artifacts for the development and deployment of models.

Life cycle, processes and tasks

The lifecycle consists of the phases ideation, data acquisition and exploration, research and development, validation, delivery and monitoring. It is depicting in great detail in figure A.19.

Phase 1: Ideation According to the authors, the ideation phase contains some of the most important work of a project. Here, the project scoping is done, which includes identifying business objectives, determining success criteria, reviewing prior art and estimating the ROI. Problems should be prioritized over data and the main goal is to gain a good understanding of business processes and pinpointing

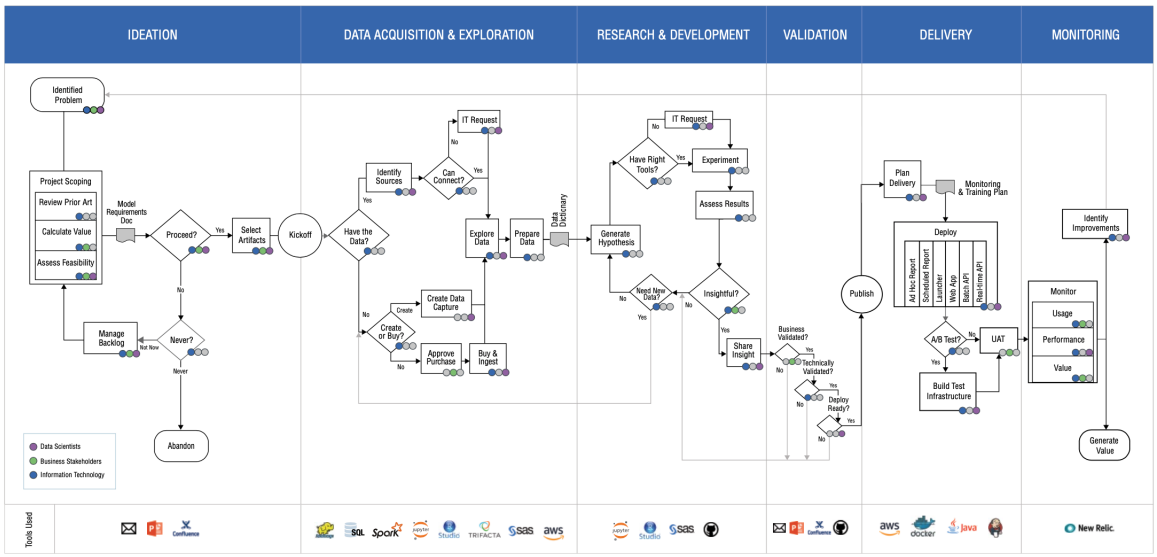

Figure A.19: The Domino DS lifecycle, including useful tools for each phase and assignments of stakeholders to tasks [123].

the potential points in processes which can be augmented or automated. The authors suggest to visually depict the business process and the parts which are tackled by the project to ensure correct understanding with the business stakeholders and to avoid missing even more impactful parts of the processes. They also suggest practicing and mastering the calculation of the order of magnitude of ROI. A central element of this phase is to review past projects. This can be boosted by maintaining a hub of past work and creating templates for model requirements. The artifact selection marks the end of this phase. In this step the stakeholders agree on the type of the final deliverable. The authors suggest including IT and engineers in this process and to also create mockups of the potential deliverable. It might also be helpful to create synthetic data and baseline models to show the potential impact of a model.

Phase 2: Data Acquisition and Prep The authors argue that the overall design of a data science solution is dependent on the data which is available. Good data understanding is a key to creating analytical value from data and both together form an iterative process. It is a good practice to ask the stakeholders for their intuitions and knowledge about the data to gain insights about the usefulness of features. Good teams should consider datasets as a reusable component which is shared across projects and might want to establish feature stores to avoid repetitive preprocessing and cleansing. If external data is used, it should be tracked where and to which impact the data is used to justify future purchases. It is also beneficial to establish good processes for assessing external data in processes which include data vendor selection, data evaluation, procurement and ingestion.

Phase 3: Research and Development This phase is the perceived heart of the data science process. The Domino DS lifecycle encourages starting with simple baseline models and suggests to deliver insights continuously in short iteration cycles. This gives room for internal and external feedback, ideas and helps tracking progress. A major point of Domino DS is to keep track of business KPIs consistently rather than just optimizing model metrics. Standard hardware and software configurations can help to ease the onboarding of new project members.

Phase 4: Validation Automated validation checks can be used to support human inspection. This helps humans focus on the critical aspects of a system. Discussions about model development choices should be preserved in the context of the experiments so that the team members can better learn from them in the future. Null results should not simply be omitted but also be documented to avoid making the same mistakes multiple times.

Phase 5: Delivery Dependent on the choice of the result artifact type, its delivery may differ. It can take the form of a report, a dashboard, a modeling pipeline or a model deployed in a production system. All different artifacts should be linked to better preserve feedback and to save time. Team members should be incentivized to promote their models to production and always flag dependencies, such as the data a model was trained on, the transformations and tools which were used to alter it.

Phase 6: Monitoring Monitoring of the deployed models is crucial and requires collecting semantic and statistical measures. The authors suggest to code value range constraints into monitoring and to include IT and engineering teams in this phase to implement guardrails.

Roles and responsibilities

The domino ds lifecycle suggests that a full stack data scientist is not an appropriate role for data science projects. The competences should rather be split over multiple specialized roles, namely the data scientist, data infrastructure engineer, data product manager, business stakeholder and data storyteller. The staff should be organized in a so called ”hub and spoke” model, i.e. a central team which is responsible for building core technical infrastructure and decentralized teams which address business problems and maximize the adoption of solutions.

Artifacts

No artifacts are specified.

Tools and techniques

Domino DS provides an overview of tools which are helpful in each of the project phases:

- **Ideation**: Email. Microsoft Powerpoint, Confluence
- **Data Acquisition & Exploration**: Hadoop, SQL, Spark, Jupyter, R Studio, Trifacta, SAS, AWS
- **Research & Development**: Jupyter, R Studio, SAS, GitHub
- **Validation**: Email, Microsoft Powerpoint, Confluence, GitHub
- **Delivery**: AWS, Docker, Java
- **Monitoring**: New Relic

A.1.22 Data Science & AI lifecycle

The data science and ai lifecycle (DSAL) by John Thomas mostly aims at teams which serve as data science units within a company. Thomas argues that many organizations build data science teams staffed with skilled people but still fail to bring data science projects into production and integrate the

results in their processes. He calls for a systematic approach to operationalizing AI, which requires an end-to-end lifecycle. The original version of the lifecycle was published in 2019 [229], a newer, revised version was published in 2021 in the book "Operationalizing AI" [47] by Thomas et al.

Life cycle, processes and tasks

The lifecycle is split into 5 phases and highlights the involvement of 8 different project roles:

Scope The first phase in the lifecycle is called scope. In this phase data scientists and business users scope the organization and its data for potential use cases and prioritize it. The tasks in this phase include the exploration of business cases, the assessment of feasibility, the prioritization and selection of use cases, establishing business KPIs and swim lanes for each use case. Thomas suggests analyzing the business ideas in detail and to always formulate clear business KPIs. This helps the data science teams to cope with large amounts of requests. Data scientists and business users should be aware that there could be major differences between business value and model performance, it is therefore necessary to specify the business KPIs before working on problems.

Understand In the understand phase a dataset is built which suits the needs of the use cases. This phase involves the data scientist, a data provider, a data steward and a data consumer. All of the different roles together agree on which heterogeneous sources to use and define usage policies. The data provider helps to discover relevant datasets, tracks the lineage of the dataset, creates a data catalog, curates and classifies data and provides access to it. Documenting the data lineage may prevent using data for the wrong use cases. The data steward is responsible for creating policies for the different data assets and grants access to them. The data consumers are responsible for finding and identifying potentially relevant data for the business problems, understanding the data, if necessary, adding to the data, exploring it and reviewing it for the respective business problems. Once explored and reviewed, the data needs to become accessible for all stakeholders who work with it. Thomas emphasizes that good data is central to successful projects and calls for giving data science teams enough freedom and incentive to discover, explore and understand data. He also stresses that moving data is a costly process and should be avoided if possible.

Build The build phase centers around identifying and using the right tools and techniques to prepare the data, engineer features, build and train models. It starts with a data analysis and the preparation of data for the model training. This includes preprocessing, feature engineering and possible data transformation. In the next step the models are conceptualized, build and then trained. Possible results of this are a model canvas and implementation code. After training the model, it is evaluated. In the next step the data engineers and data scientists tag the model for review and eventually publish it to a model catalog. This phase marks the start of using continuous integration and deployment (CI/CD) pipelines.

Deploy and run After publishing the model, it is reviewed for deployment. The review includes code review, third party oversight and unit tests. If successfully reviewed, the model is tagged for deployment. It is then deployed to a ML runtime engine and monitoring and evaluation are set up. Finally model management also belongs to the tasks of this phase. The deploy and run phase includes software engineers, data scientists and business users.

Monitor and manage In the last phase the model is configured for monitoring and integration. The AI Ops team and the business user collect insights about the model quality, its throughput, its metrics and fairness. If necessary particular transactions of the model can be explained on-demand.

Roles and responsibilities

Thomas et al. define 8 roles, which they call personas, and assign them to the different phases of the life cycle. Figure A.20 depicts the phases of the life cycle including the roles which are active in each phase.

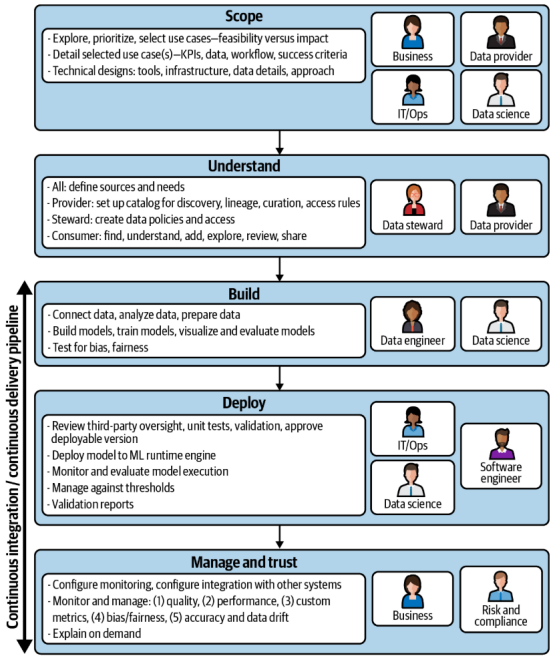

Figure A.20: Description of the tasks in each of the phases of the AI cycle and the roles active in each phase according to [47].

The 8 roles are further described in the book:

1. Leadership: Business owner, leads the data product teams. Helps defining minimum acceptable performance, focuses work on business goals, advocates data science efforts. Could be C-level, business owner, data science lead, has expertise but is not practically involved in modeling.

2. Data Analyst: Help finding relevant data, build reports and dashboards to deliver insights. Have a good understanding and intuition of the data and its value.

3. Data Scientist: Supports the process of acquiring and cleaning data, performs feature engineering in close cooperation with domain experts. Train and evaluate models with regards to performance, transparency and explainability and are often also involved in deployment or responsible for it themselves.

4. Data Engineer: Develop pipelines for providing data for model training, persist the predictions of models and maintain the infrastructure the data scientists work with.

5. Data Steward: Grants and restricts access to data, enforces privacy policies.

6. Product Manager: Create a roadmap for AI products, translate business needs into product requirements. Needs to communicate with the business and the technical side.
7. Risk Manager: Assess various parts of models such as the fairness and compliance before deployment in a productive setting. Should have access to model monitoring and management, are typically involved in the later phases of the life cycle.
8. Software Engineers: Complement the work of data scientists, often work on software that consumes the predictions of a machine learning model.

However, the authors acknowledge that there could be many more roles involved in the operationalizing of AI. A complete list of roles, without further description is provided by them in figure A.21.

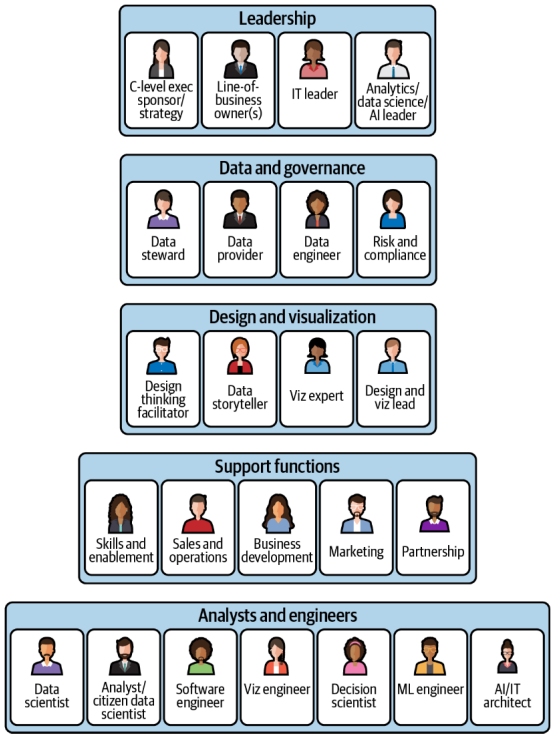

Figure A.21: A complete overview of roles which are involved in operationalizing of AI according to [47].

Artifacts

The authors do not explicitly specify artifacts.

Tools and techniques

Thomas et al. mention a couple of tools for different aspects of the AI lifecycle.

1. Languages

a) Python
b) R
c) Scala

2. Machine Learning Frameworks
   a) Scikit-learn
   b) XGBoost
   c) Tensorflow
   d) PyTorch
   e) Spark MLLib
   f) Keras
   g) Spark
3. Authoring Tools
   a) R STudio
   b) Data Refinery
   c) Jupyter
4. CI/CD
   a) Bamboo
   b) Jenkins
   c) Confluence
   d) Kubeflow
   e) Travis
5. Repository Services
   a) GitHub
   b) Bitbucket
   c) Jira
   d) SVN
   e) Frog artifactory
6. Productivity Enhancement
   a) Visual modeling
   b) AutoML
   c) AutoAI
7. Development Tools

   a) DataRobot
   b) H2O
   c) Watson Studio
   d) Azure Machine Learning Studio

8. Multicloud
   a) Azure
   b) AWS
   c) GCP
   d) IBM cloud

9. ML Approaches
   a) Classic ML, e.g. regression
   b) Transformer Networks

A.1.23 MLOps

Many of the newer methodologies introduce phases and tooling for the automation of phases of the data science lifecycle and the operation of machine learning solutions. Recently the term MLOps has been coined to describe a combination of machine learning and DevOps practices. Kreuzberger et al. have performed a systematic literature review and expert interviews to understand the principles, components and workflow of MLOps [125]. They introduce their own definition of it: "MLOps (Machine Learning Operations) is a paradigm, including aspects like best practices, sets of concepts, as well as a development culture when it comes to the end-to-end conceptualization, implementation, monitoring, deployment, and scalability of machine learning products. Most of all, it is an engineering practice that leverages three contributing disciplines: machine learning, software engineering (especially DevOps), and data engineering. MLOps aims at productionizing machine learning systems by bridging the gap between development (Dev) and operations (Ops). Essentially, MLOps aims to facilitate the creation of machine learning products by leveraging these principles: CI/CD automation, workflow orchestration, reproducibility; versioning of data, model, and code; collaboration; continuous ML training and evaluation; ML metadata tracking and logging; continuous monitoring; and feedback loops." [125] Symeonidis et al. also screen current tools and techniques in MLOps and describe it as collection of techniques and tools for deployment of ML in production with the aim to automate ML processes using practices and approaches derived from DevOps [8]. The ultimate goal of MLOps is to lift machine learning based solutions from a proof-of-concept state towards usage in production environments. Kreuzberger et al. derive nine principles of MLOps:

1. CI/CD automation: continuous integration, delivery and deployment, build, test, delivery and deploy steps and fast feedback for developers

2. Workflow orchestration: coordinate tasks of ML pipeline according to directed acyclic graphs

3. Reproducibility: reproduce experiments with exact same results

4. Versioning: versioning of data, model and code - also for traceablity
5. Collaboration: collaborative work on data, model and code - collaboration and communication, reduce role silos
6. Continuous ML training & evaluation: periodic retraining of ML model, monitoring, feedback loop and automated ML pipeline and evaluation run
7. ML metadata tracking/logging: track metadata for each ML workflow task and each training job iteration, including hyperparameters, model performance, lineage, etc.
8. Continuous monitoring: periodic assessment of data, model, code, infrastructure resources, model serving
9. Feedback loops: integrate insights from the quality assessment into development and engineering and retraining

Symeonidis et al. also underline the importance of CI/CD and further emphasize the importance of automated model retraining.

Life Cycle, Tasks and Processes

Kreuzberger et al. describe a MLOps life cycle which spans the complete data science life cycle, from business understanding to operation of solutions. Symeonidis et al. rather see MLOps in the phases of data collection and preprocessing, feature engineering, model creation, training and testing and monitoring and improvement. Since the phases of Symeonidis et al. are a subset of the phases of Kreuzberger et al. we follow the more comprehensive take:

Phase 1: MLOps project initiation The project starts with a business problem identified by a business stakeholder who believes that it can be solved with machine learning. Based on the problem definition a solution architect defines a target architecture and the overall machine learning system. The solution architect further picks appropriate technologies for the system. A data scientist then translates the business problem into a machine learning problem. Afterwards the data scientist and a data engineer obtain an understanding of which data is needed, they locate appropriate sources, validate the data and check it for appropriateness for the task, e.g. by verifying that the data is appropriately labeled.

Phase 2: Requirements for Feature Engineering In the next phase data and software engineers define requirements for a feature engineering pipeline. These updates are regularly updated based on feedback which is obtained in later phases. The data engineer writes code for the CI/CD pipeline, an orchestration component and defines the infrastructure resource configuration.

Phase 3: Feature Engineering Pipeline Based on the requirements a feature engineering pipeline is built. The pipeline connects to raw data, extracts it from the sources, preprocesses it via transformation and cleaning and finally calculates new and more advanced features based on predefined feature engineering rules. These rules are improved over time. Finally, the features are written to a feature store.

Phase 4: Experimentation A data scientist will then perform experiments with support of software engineers. The data scientist therefor connects to feature store systems, prepares and validates their data, creates a train/test split of it and then picks models, trains them with different parameterization

and then evaluates them. Finally, the model is exported and model code is committed to a code repository. DevOps and ML engineers then write the code for automated ML workflow pipelines. These pipelines are then built, tested and delivered via CI/CD.

Phase 5: Automated ML workflow pipeline In the last phase the automated workflow pipeline is managed by DevOps and ML engineers. The pipeline should allow for retraining when new, unseen data is available and should store metadata with concern to the training and experiments. Once a solution is deployed, ML engineers should monitor the system performance.

Roles and responsibilities:

MLOps is described as an interdisciplinary group process [125]. Kreuzberger et al. provide seven role defintions for MLOps processes and describe their responsibilities. The roles they suggeest are business stakeholder, solution architect, data scientist, data engineer, software engineer, DevOps engineer and ml engineer.

Artifacts

Neither Kreuzberger et al. nor Symeonidis et al. provide descriptions of artifacts.

Tools and techniques:

Both Kreuzberger et al. and Symeonidis et al. provide exhaustive lists of tools which can be used for MLOps:

- CI/CD Component: Jenkins, GitHub actions
- Source Code Repository: ensure code storing and versioning, allows developers to commit and merge their code. Bitbucket, GitLab, GitHub, Gitea
- Workflow Orchestration Component: Apache Airflow, Kubeflow Pipelinee, Luigi, AWS, SageMaker Pipelines, Azure Pipelines
- Feature Store System: ensures central storage of commonly used features. Offline feature store and ony feature store. Google Feast, AWS Feature Store, Tecton.ai, Hopswork.ai
- Model Training Infrastructure: provides foundational computation resources. Distributed vs central. Kubernetes, Red Hat OpenShift
- Model Registry: stores the trained ML models with their metadata. MLflow, AWS SageMaker Model Registry, Microsoft Azure ML Registry, Neptune.ai, Microsoft Azure Storage, Google Cloud Storage and Amazon AWS S3
- ML Metadata Stores: stoe metadata of e.g. workflow pipeline tasks, model training jobs, hyperparameters, performance metrics, model lineage (data and code). Kubeflow Pipelines, AWS SageMaker Pipelines, Azure ML, IBM Watson Studio, MLflow

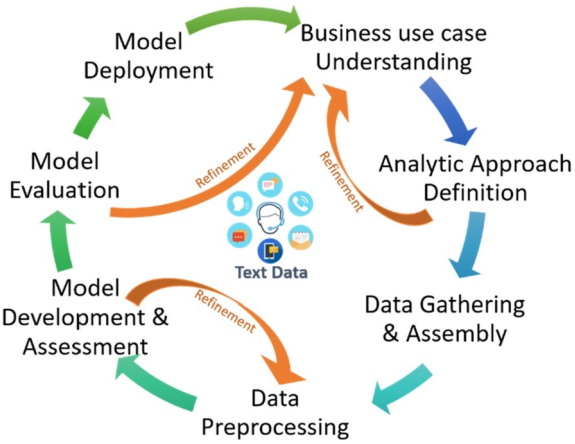

Figure A.22: The modified version of CRISP-DM for text analytics according to [126]

- Model Serving Component: serving for online vs batch inference. Often used with Flask web applications inside of Docker. KServing of Kubeflow, TensorFlow Serving, Seldion.io, Apache Spark, Microsoft Azure ML REST API, AWS SageMaker Endpoints, IBM Watson Studio, Google Vertex AI

- Monitoring Component: continuous monitoring of model serving performance (prediction accuracy), ML infrastructure, CI/CD, orchestration. Prometheus with Grafana, ELK (ElasticSearch, Logstash, Kibana), TensorBoard, Kubeflow, MLflow, AWS Sagemaker

A.1.24 Cross-Industry Process Standardization for Text Analytics

Skarpathioataki and Psannis [126] recognized the need for adapting CRISP-DM to projects which deal with unstructured data and proposed the Cross-Industry Process Standardization for Text Analytics in 2021. They emphasize that text analytics brings a lot of potential to organizations but also comes with distinct challenges due to the nature of the data and the unique methods which are proposed to handle it. They identify challenges which are related to the three V's of big data: textual data comes in large volumes, it has a large variety in languages, formats and length and it often requires to cope with a high velocity.

Life cycle, tasks and processes:

The authors modify the original CRISP-DM pgases to accommodate for specific changes required for textual data. The overall life cycle stays similar to CRISP-DM as can be seen in figure A.22.

Phase 1: Business Use Case Understanding This phase is analogous to the business understanding phase in CRISP-DM. Its goal is to correctly understand and precisely frame the business problem. This entails a detailed requirement analysis which covers both an analysis of the goals but also the generated value. According to the authors it is crucial to involve the business stakeholders in this phase and to jointly define measurable success criteria with the end-users.

Phase 2: Analytic Approach Definition In the second phase an appropriate analytics approach is chosen to tackle the problem identified in the prior phase. The authors propose a serious of analytics

methods which mostly cover traditional statistical models for text analytics and mention deep learning only as a side note.

Phase 3: Data Gathering and Assembly In this phase, the data requirements are elicited with respect to volume, content and format. Dependent on the amount of data available an upper error bound is determined. Based on the data requirements data is collected, if necessary from multiple sources. The data is assembled and first insights are extracted via exploratory analysis.

Phase 4: Data Preprocessing After gathering the data, it is cleaned and further preprocessed. This phase takes up the bulk of the project time. Text specific feature engineering is performed. The authors suggest a couple of feature engineering steps which follow traditional natural language processing methods and are not typically used in the context of deep learning based methods.

Phase 5: Model Development and Assessment Based on the business requirements and the available datasets various machine learning models are trained. If beneficial, models are combined in ensembles.

Phase 6: Model Evaluation After training, the model quality is assessed. Dependent on the task, appropriate metrics are chosen and used to verify the quality of the results. The evaluation also includes an assessment as to how well the model solves the business problem. The scalability of the model is also assessed.

Phase 7: Model Deployment Lastly, the model is deployed into a productive infrastructure. According to the authors, in this phase a decision has to be made as to which criteria are used to determine which model to deploy. There might be trade-offs between interpretability and model accuracy. The choice might also depend on the given infrastructure, where complex cases require deployment on edge devices.

Roles and responsibilities

The methodology does not explicitly describe roles and responsibilities.

Artifacts

The methodology does not mention any artifacts.

Tools and techniques

The authors specify text analytics specific tools and techniques such as ontolearn, part of speech tagging, sentiment analysis, NLP specific machine learning techniques, preprocessing steps such as punctuation removal, tokenization, stopword removal, data stemming, lemmatization, vectorization, feature engineering and finally model ensembling.

A.1.25 Data to Value

In 2022 Leidner introduced the data to value methodology [17]. The methodology is explicitly built around unstructured data, supervised learning, ethical questions, big data challenges and early evaluation.

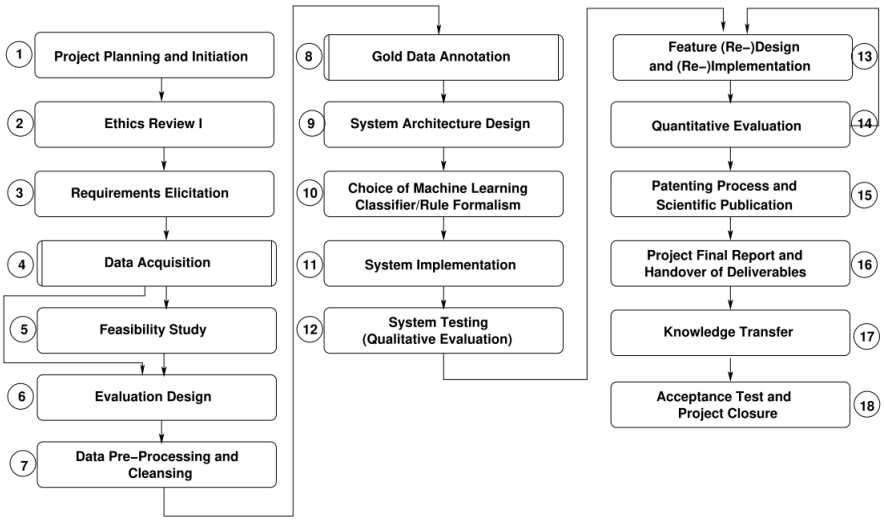

Figure A.23: The process model of the data2value methodology as introduced in [17].

Life Cycle, Tasks and Processes

The methodology consists of 23 phases and brief descriptions of these phases. Some of the phases are unique to this methodology such as the annotation phase and the ethics review. Figure A.23 depicts the complete process model of the methodology with all 23 phases.

Phase 1: Project Planning and Initiation The first phase is used to draft a project charter and launch the project. In this phase the success criteria are specified and evaluation procedures are planned.

Phase 2: Ethics Review I In the beginning of the project the question is answered whether it is ethically right to proceed with the project or whether any output might be morally objectionable.

Phase 3: Requirements Elicitation In this phase functional and non-functional requirements are derived.

Phase 4: Data Acquisition The authorization and access to datasets is obtained.

Phase 5: Feasibility Study Optionally, a feasibility study can be performed, which lowers the risk by giving an estimate of the expected quality on a subsample of the data. This phase can be split into the same phases as the whole project.

Phase 6: Evaluation Design The stakeholders agree on the experimental design and evaluation protocol.

Phase 7: Data Pre-Processing and Cleaning The data is brought into the right format, noise is eliminated, data is translated into a canonical form and different datasets are linked. This might be the most time-consuming phase.

Phase 8: Gold Data Annotation To train supervised classifiers, training data is annotated. This phase is split into multiple subphases. First, a small subset of data is inspected and annotated. Annotation guidelines are built based on the first annotations. Multiple annotators annotate a small dataset and the inter-annotator agreement is calculated. Discrepancies are reviewed and the guidelines are adjusted accordingly.

Phase 9: System Architecture Design The architecture of the system is designed.

Phase 10: Choice of Machine Learning Classifier/Rule Formalism The team decides whether to use rule-based formalisms or machine learning classifiers and which ones.

Phase 11: System Implementation The processing engine is implemented, independent of rules and features.

Phase 12: System Testing The system is tested based on manual and automated tests.

Phase 13: Feature Design and Implementation Brainstorming sessions about possible features are held, the data is studied and models or rules are implemented in implementation sprints. The phase ends either because the desired quality was reached or because the project runs out of budget.

Phase 14: Quantitative Evaluation Features are evaluated based on appropriate quantitative metrics and if necessary new features are tried.

Phase 15: Patenting and Publication The work is documented and patented.

Phase 16: Final Report Authoring Data, software and documentation are handed over.

Phase 17: Knowledge Transfer Findings are communicated to the stakeholders.

Phase 18: Acceptance and Closure The research part of the project ends.

Phase 19: Ethics Review II If the results of the project should be used in production, another ethics review is performed with a focus on the system functions and its emergent properties.

Phase 20: Deployment The system is deployed.

Phase 21: Monitoring The system is monitored in production and relevant events are logged.

Phase 22: Quantitative Evaluation II The performance of the systems is evaluated in the production setting and with respect to changing data.

Phase 23: Model Re-Training If necessary, the models are retrained to account for changes in the data.

Roles and responsibilities

The methodology does not explicitly describe roles and responsibilities.

Artifacts

The methodology mentions a couple of artifacts without further description. It also comes with a catalog of questions, which was only partially released by the author. The artifacts which are mentioned are the project charter, project plan, (annotated) datasets, annotation guidelines, system architecture design and final report.

Tools and techniques

No tools and techniques are explicitly mentioned.

A.2 NLU Case Studies

In this section, we present the remainder of the project case studies, which we used for our analysis of project-relevant characteristics in Chapter 4.

A.2.1 Medical Publication Mining

In this project the goal was to mine data about colorectal cancer from the medical publication library "PubMed" and create an application that helps doctors with the treatment of the disease. The project focused on patients which did not respond to any treatments in the medical guidelines.

Project Setting and Problem Description

The project consortium consisted of six partners with different expertise. One of the partners was a German clinic. The problem setting was open at the project start, i.e. project partners were aware of various problems in colorectal cancer treatment, but it was not clearly defined which problems to approach and how. What was clear from the start was that publications, clinical data and medical documentation should be included in the analysis for the final application. The NLU team focused on the implementation of the literature mining solution. This included setting up crawlers, preprocessing the data, modeling and creating appealing visualizations of the results, which are easily understandable for the medical staff. The NLU team opted for following two approaches: using topic modeling to generate a general structure of the 300.000 relevant publications about colorectal cancer and building an information extraction pipeline using structured resources such as medical ontologies to automatically annotate relevant information regarding biomarkers, treatments and diseases in the PubMed literature.

Roles involved

Due to the large project consortium, we will focus on the roles included on the NLU team side and the actual users, i.e. the staff from the hospital which participated in the project. On NLU side there were two junior data scientists which were responsible for all tasks regarding data processing, modeling and application development. Further, halfway in the project a project lead joined who was responsible for administrative and legal questions, as well as coordination and communication with and among the partners. Both the junior data scientists as well as the project lead worked on gaining a better understanding of the domain problem and how to solve it. The medical team comprised of two doctors. One who was responsible for project leading and one who was responsible for data annotation as well as providing domain expertise.

Project Outcome

The outcome of the project was successful, i.e. a topic model of the publications was created, which was deemed useful by the medical staff and was integrated in the final application. Further, triplets of biomarkers, treatments and outcomes were automatically extracted and provided to the medical staff.

Most important Characteristics

Management/Governance - Team Organization - Stakeholder Involvement - Domain Experts
The availability of domain experts was crucial for the success of the project. They were involved in every phase of the project and gave relevant insights into the domain, the problems they face and the quality of solutions.

Process - Project Ideation - Mutual Understanding In the beginning, before the new project lead joined, the project stagnated. After they joined, a second project iteration started, with the refinement of the understanding of the problem domain as well as the capabilities of NLU methods. With this better understanding the scope definition, data selection and modeling became a lot easier.

Process - Project Management The absence of a project manager in the early phases of the project led to a slow and uncoordinated start. With the assignment of the new project lead, the project achieved a clear schedule, better coordination and communication as well as solution driven direction.

A.2.2 Pathology Report Understanding

In this project the goal was to extract information from German pathology report. The client was a clinic which performed research on colorectal cancer treatment and wanted to automatically extract relevant information about tumors such as the type of tumor, its phase, genetic information and more, from pathology reports.

Project Setting and Problem Description

At the start of the project there were no annotated pathology reports. The client used the NLU team's annotation tool to manually annotate pathology reports based on an annotation scheme which was derived jointly. Goal of the project was to prove that the information extraction process could be automated with a high accuracy. Around 100 pathology reports were pseudonymized and annotated by a medical domain expert in the project and then used for training, validation and testing. A pre-trained German language model was used and adapted for named entity recognition. It was fine-tuned on the small number of labeled samples. During the evaluation and visual inspection of the results in an annotation tool, the team recognized that poor scores were mainly due to noisy labeling, i.e. the model annotated passages, which were missed by the domain experts, but still correct.

Roles involved

The NLU team consisted of a single person being the project lead and the data scientist in the project. The client team consisted of clinicians of which one was also the project lead.

Project Outcome

The project was a success, i.e. the results of the automated extraction were satisfying and the model even recognized annotations which were missed by the domain experts.

Most important Characteristics

Process - Model Development - Model Reuse Given the small number of annotated documents and the number of entity types, it is evident that this project benefited from reusing a pre-trained German language model. This significantly decreased the number of annotated data needed to train a model with high accuracy.

Technology and Infrastructure - Visualization Visualizing the results of the model was a key factor in understanding the performance. Only when the domain experts and the data scientists saw

the model results, they could easily identify that the metrics underestimate the quality of the model, due to noisy/missing annotations in the ground truth.

Management and Governance - Stakeholder Involvement - Domain Experts The involvement of domain experts was especially important for data labeling but also for the inspection of the results. Since the data scientists was no expert in the medical domain, he needed the medical expertise by the clinicians in order to understand the results and to be able to obtain training data.

A.2.3 Opinion Detection in Automobile News

This project was with a client who consults automotive companies. Their goal was to provide car manufacturers with insights into brand reputation by mining news articles. Machine learning models should detect the names of brands, their products and opinions on them, as well as the respective sentiment of the opinion.

Problem Definition

At the beginning of the project no data was available, i.e. news sources had to be selected, crawlers had to be set up and news had to be annotated to train machine learning models to detect the relevant information. A list of car manufacturers and model names was provided by the client which was used to automatically annotate occurrences of them in text. However, the opinions, their sentiment and the links between opinions and manufacturer and brand names had to be annotated manually. To this end, the custom annotation tool of the NLU team was used and the client stakeholders were educated in using it. Further, an annotation guideline was established jointly in annotation workshops. The goal of the project was to deliver a service which processes news, extracts relevant information and then integrates them in a customer-facing front-end.

Hundreds of relevant news articles were selected based on the occurrence of relevant brand and model names and around 100 of them were annotated for information extraction. The problem was split into an information extraction, linking and classification task.

The results of the project should be used as a demonstrator for automotive companies, showing them a dashboard with current news and the extracted information as well as aggregation views.

Roles involved

The NLU team consisted of a project lead, a senior data scientist and a junior data scientist. The project lead was responsible for communication and coordination but also involved in data gathering and preprocessing tasks. The senior data scientist helped with the annotation process and modeling, while the junior data scientist focused solemnly on modeling tasks. The project team on the client side consisted of a project lead and a junior consultant. The project lead was not technically involved and mainly focused on project evaluation and communication. The junior consultant was involved with data annotation, answering domain questions and discussing implementation details.

Project Outcome

The project concluded without a clear evaluation as the client underestimated the effort necessary to finish the modeling tasks. When the expectation were not met after the first annotation and modeling

iteration, the project ended. The first version of the models yielded results which were skewed to negative sentiments and did not provide a lot of insights which were critical for car manufacturers.

Most important Project Characteristics

Process - Project Management - Stakeholder Management - Expectation Management Right at the beginning of the project the expectations with regards to the results and especially the initial results after one modeling iteration should have been communicated more clearly. The expectations were set too high when the client saw the results of a similar project without making the project process transparent.

Process - Data Processing - Data Augmentation The data augmentation process took a lot of resources from the client side, especially in the form of domain experts who had to analyze and annotate the according news articles. This led to the situation that the client wanted to achieve results after an insufficient number of samples was labeled. Further the labels were skewed because of only using news article from a certain timeframe, leading to correlations between brands and sentiments.

Artifact and Asset Management - Software/Code - Reusability A key factor for acquiring this project was the ability to demonstrate that it had worked in a prior project and that the code base for training and inference for the same model type was already established and functioning. This reduced the cost of the project and made it more accessible for the client. However, it also came with the downside that the client expected the project to be easy and with low effort regarding data processing and modeling.

A.2.4 Social Media Analysis

Here we discuss two research projects which were executed simultaneously and very similar in their nature and their results are presented in [25]. The scope of the two research projects was the analysis of data from social media. In one project the goal was to understand the perceived safety of civilians in local communities. The goal of the other project was obtaining information from social media in emergency scenarios such as earthquakes, floods or wildfires.

Project Setting and Problem Description

In both projects the goal was to create a component which analysis social media streams and identifies posts relevant to certain topics to gain insights. Both components should then be integrated into larger systems derived by project partners. Both projects had similar structures with large multinational project consortia, containing computer science/data science partners as well as partners from different fields. In both cases the partners wanted to develop applications jointly, with different teams focusing on different aspects of the application. In both cases, the NLU team was responsible for developing topic models which structure large document collections into topic clusters. These topic clusters should represent domain-specific topics as well as a ”non-relevant topic”. There was no annotated data available and the topics were not known a priori. However, in both cases terminologies with relevant terms were available. These were used as seed terms to filter out relevant social media posts. Different topic modeling approaches were benchmarked and after obtaining a final model, the model was integrated in a new software component which was developed for both projects. The component took social media posts, pre-processed them and then applied the respective topic models

and visualized social media posts on a geographic map based on their meta-data. This way end-users could monitor where people discussed relevant topics and which topics in particular.

Roles involved

As the project consortia consisted of a large number of stakeholders, we focus on the people involved from the NLU team. The project team in both cases consisted of a project lead, a junior data scientist and an MLOps engineer. The project lead was mainly involved with organizational matters, communicating with the leads of the other partners and discussing project process. In both projects the data scientists worked rather independently. When the data scientists realized the similarities between their projects they decided to work on a shared solution and include an MLOps engineer for application development.

Project Outcome

Both projects ended successfully from a machine learning perspective, yielding insightful topic models, a functioning application which could also serve as geospatial demonstrator and a publication of the results. However, in both cases the goal of integrating the NLU component in another application was not achieved.

Most important Characteristics

Process - Project Management - Stakeholder Management - Coordination and Collaboration Both projects suffered severely from the fact that there were many project partners involved but a rather passive coordination of efforts with little to no collaboration among stakeholders. This played an important aspect along the whole projects, starting with little to no conceptualization of a joint application, only minimal coordination of tasks and no joint development framework.

Artifact & Asset Management - Software/Code - Reusability Aiming for reusability in the design of the social media analysis component actually enabled the success of the machine learning part of both projects. Key to this was a concept which did not tackle one domain specifically but was problem-centered and domain-agnostic.

Management/Governance - Team management - Team Organization A vital factor in achieving a success in both projects was the restructuring of the team organization by merging the teams of both projects and letting them successfully collaborate in designing a solution.

A.2.5 Patent and Publication Mining

The goal of this project was to help develop a tool that automatically classifies patents and publications according to pre-defined classes and relevant tags. The public patent classification system ”CPC” was used in conjuncture with automatically detected keywords. As a result, a service should be provided that automatically assigns classes and tags to a new patent or publication.

Project Setting and Problem Definition

In this project multiple partners worked together: The NLU team, a client who wanted to use the analysis of patents and publications for innovation management and a third party, which was the

software provider who should integrate the analysis component in their infrastructure. The data consisted of publicly available publications and a patent dataset. In the beginning of the project, it was unclear what should be built, it was just known that machine learning should be used to improve the innovation management processes. The client wanted to use this dataset to understand topics of interest and to categorize publications and patents into these topics. The team opted for a two-fold approach, first training a classification model that assigns ”CPC” patent classes to the documents and then another approach which first derives keywords and then classifies documents according to a set of the most common keywords. This way the keywords are more treated like tags that were obtained in an unsupervised. In addition to the pure classification system a visualization was developed which showed the links between classes and keywords. The NLU team had strict time constraints, forcing them to come to an end of the project in as little time as possible.

Roles involved

On the NLU team side a project lead and a senior data scientist were involved throughout the project. Later in the project an MLOps engineer joined. On the client side, a project lead and domain experts were involved. The third party involved an IT project lead.

Project Outcome

Despite the strict time constraint, the project had a successful outcome in that a component was delivered which successfully was able to classify both publications and patents according to the ”CPC” categories and a set of keywords. This component was delivered to the third party as a containerized solution.

Most important Characteristics

Technology and Infrastructure - Visualization Tools The senior data scientist came up with an innovative way of visualizing the results of the model which helped both the technical project stakeholders as well as the end users to understand the model better. This was a vital tool in debugging the model, understanding its capabilities in such a scenario where there are no clear benchmarks and also in the communication with the client and the end users.

Management & Governance - Team Management - Team Organization The project team for this project was rather small but benefited largely from a separation of concerns. A dedicated project lead was involved to handle planning, scheduling and similar project management tasks. The senior data scientist involved in the project was very experienced and able to handle the project tasks without a lot of support. Further the senior data scientist could draw from a lot of experience with similar projects and come up with creative solutions.

Process - Project Management - Planning Planning was of the essence in this project as there were only little time and resources available. The project lead thoroughly planned the next steps jointly with the senior data scientist and the other project stakeholders.

A.2.6 Retail Product Classification

In this project the NLU team worked together with a client who was a tech consulting firm for a large German retail chain. The goal of the chain was to modernize their online shop and re-categorize all of

the items available.

Project Setting and Problem Definition

The goal of this project was to annotate a set of millions of items with category labels with as little as possible human effort. The NLU team got access to a set of 200,000 annotated items, labelled in a three-level hierarchy with 599 categories on the lowest level. The goal was to create a service which categorizes product descriptions into categories on each of the three levels. An important requirement was that the model should take in the whole dataset as a batch, i.e. millions of articles had to be classified in one request. The NLU team opted to try various classifiers with different computational complexity to assure a good trade-off between resource demand and performance. Finally, the service was hosted on the premises of the NLU team and exposed as an API. The project had to go from handover to deployment in a couple of weeks as there was a lot of pressure on the client side.

Roles involved

On the side of the NLU team a project lead and a junior data scientist were involved. The project lead handled the communication with the customer and gave technical guidance to the junior data scientist. The junior data scientist explored various models. On the customer side a project lead, a data scientist and a technical lead were involved.

Project Outcome

From a machine learning perspective the project was a success. High f1-scores of about 90% were reached with a classification time of only a few minutes for all articles. However, the client decided to not pursue this route further as the catalog was frequently subject to change and the frequent retraining of models was not deemed a viable solution.

Most important Characteristics

Artifact and Asset Management - Model - Resource Demand The resource demand played an important role in determining which models to use as the model had to have a high inference speed in order to process millions of articles regularly.

Process - Project Ideation - Mutual Understanding The understanding of the goals was flawed on both the NLU team and the client side. The client often adjusted the goals and lacked an understanding of the effort necessary to retrain models. In the end it was even questionable if the problem should be tackled with machine learning at all.

Artifact and Asset Management - Software/Code - Reusability Resuability of classification and deployment code played a major role in being able to deploy the inference service as quickly as necessary.

A.2.7 HR document classification

The client in this project was a scan service provider, i.e. a company which processes documents for other companies and performs tasks such as scanning, separating and classifying documents as well as extracting information from them. The process of the company was tailored around human

annotators and it took months to process large amounts of documents, often in the dimension of more than 100,000 documents. The goal of the project was to explore whether machine learning models could improve the classification process, i.e. if the human annotators annotate roughly 15% to 30% of the dataset, could the rest be handled by a machine learning model reliably.

Project Setting and Problem Description

The project started after the client had successfully finished a project in which they separated and classified 120,000 documents for a customer. The client wanted to benchmark the ability of NLU models to automate the classification with 20,000 annotated documents and with 40,000 annotated documents. The data was available as text extracted from the document by an OCR, i.e. the model had to be robust against spelling mistakes. The classification problem contained 72 classes which were heavily imbalanced in their frequency. Further the customer had strict resource constraints, i.e. the model had to run on CPU in an acceptable timeframe and classify a batch of up to 100,000 documents. Training was performed on premises of the NLU team, which had to take extra precautions for processing the data as it was highly sensitive due to privacy constraints.

Roles involved

On side of the NLU team a project lead and a junior data scientist were involved. On customer side the CEO of the company was involved as well as an IT professional to help with the data transfer.

Project Outcome

From a machine learning perspective the project was a success. Both best performing models, one trained with 20,000 documents as well as one with 40,000 documents, performed on par with human annotators. They reached f1-scores of roughly 95%. However, it took the models only three minutes to compute labels for the 80,000 - 100,000 documents in the test set, as opposed to months in the case of humans. Due to flaws in the scoping and the conceptualization of the project, the service was never deployed on customer side. The client realized that the classification component alone would not suffice, if the model was not able to also separate the documents. Most of the time human annotators would annotate both information in one step. Further, the client realized that they had to train new models for every project in which new categories of document labels had to be assigned.

Most important characteristics

Artifact/Asset Management - Software/Code - Reusability A major benefit for the NLU team in the project was that they could focus on exploring various models without too much programming and data preprocessing as there were available data and model pipelines which they could simply reuse. This allowed the NLU team to execute the model despite a low budget on the client side.

Process - Project Ideation - Mutual Understanding Both parties should have invested more time in understanding the business goals and the desired project outcome. Not understanding that the classification would be nearly worthless without the document separation led to the result of not pursuing further activities or even deploying the classification model.

Artifact/Asset Management - Model - Resource Demand The resource demand played a large role in deciding which model type to use. Since the constraint was that the model should be executable

on a CPU with little processing time, this significantly decreased the number of models which could be considered for the task.

A.2.8 NLU Covid Support

In this project the NLU team worked jointly with a medical client who wanted to examine two ways of supporting the fight against Covid. In the first project stream of the project the goal was to detect misinformation about covid, the second stream was about information extraction from medical reports.

Project Setting and Problem Definition

The project was part of an initiative to provide help during the pandemic. The client was a research institute with close ties to the healthcare domain and to certain clinics. At the start of the project there was no data available and the goal of the two different work streams was not clear. A major task in the beginning of the project was to understand what could be done and which data would be necessary and available. Due to regulatory issues there was no data available for the information detection stream until the end of the project. On the fake news detection side another partner joined the project which maintained a website for rating medical news based on their factual correctness. The aim of this project stream was to detect whether news articles use sensationalist language rather than journalistic language when reporting medical news.

Roles involved

On the side of the NLU team a project lead and three different data scientists were involved. One data scientist was assigned to the information extraction stream, the other two to the fake news detection stream. On the side of the client one project lead and two domain experts with medical background were involved.

Project Outcome

The project was only partially successful. In both work streams there was no real data and the goal was not clearly specified. Intermediate results such as reports were created but no software was deployed at the client side.

Most important Characteristics

Process - Project Ideation The project ideation posed a real problem for the project. At no point was it really clear what should be achieved and hence the requirements were not clear at the start of the project.

Artifact & Asset Management - Data - Sources - Availability For the information extraction use case data availability posed a real problem. Later in the project a clinic offered to gather data for the use case, however it was not possible due to regulatory constraints.

Process - Project Management The project manager left most of the planning tasks to the data scientists who would have clearly benefited from more guidance and a clear roadmap. This way the team had to self-organize which was difficult given the circumstances of the project.

A.2.9 Information Extraction from Invoices with an SME

In this project the client was a small enterprise which built software for document management systems. Goal of the project was the integration of an automated information extraction service which processed invoices and automatically extracted relevant information from them.

Project Setting and Problem Description

This was the clients first project with a machine learning context. They gathered their knowledge about machine learning from the interaction with the NLU team, which took some time to explain the potentials and the risks of machine learning, i.e. for example the uncertainty of predictions and the necessity to have a human reviewer to correct model mistakes. The client had access to around 6,000 unlabeled invoices and a database which contained the respective information to the invoices which should be automatically extracted by the new service. The NLU team modified and reused a framework for the automated weak annotation of the documents based on the structured information. This way no human annotations were required. As machine learning model a pre-trained model from a different project was reused and fine-tuned on the weakly annotated data. The model was integrated in an already existing service for invoice processing, which the team had built as a demonstrator before. This service had to be deployed as a docker container on a windows machine which could not run Linux based containers. This requirement was not communicated to the NLU team a priori leading to significant development overheads.

Roles involved

On the side of the NLU team a project lead, two junior data scientists and an MLOps engineer were involved. On the client side the CEO of the company was involved, a project lead and the lead IT professional.

Project Outcome

The project can only be considered partially successful. Although the model met the expectations communicated by the client, the client struggled to integrate the final component in their system due to difficult hardware and infrastructure constraints. However, the hardware requirements were communicated early on.

Most important Characteristics

Process - Project Ideation - Requirements Engineering In the beginning of the project not enough time was spent on requirements engineering. This led to the problem that it was not clear that the software had to be delivered as a Windows based docker container and had to run on older machines. The project plan was severely impacted by this and the budget and time needed were a lot higher than expected.

Management/Governance - Organization - Maturity and Readiness The client organization clearly lacked experience with integrating machine learning software and had an overall low maturity and readiness in their integration process. This led to many problems in the deployment of the software and to frustration on the client side.

Technology and Infrastructure - Data Tools Using tools which automated the labeling of the documents is the characteristic which enabled the project in the first place and can hence be considered a key success factor.

A.2.10 Classification of Documents in the Mail Inbox

This project was a proof-of-concept with a small company which offered a document management software. Goal of the project was to show the potential of NLU methods for automating the classification of documents which are added to the software.

Project Setting and Problem Description

The project was planned as a proof-of-concept as the client wanted to understand the potential and limitation of NLU models. The client provided the NLU team ten-thousands of documents, labeled according to 93 different classes. The data contained a high imbalance of classes, almost following an exponential curve and there was a lot of noise in the data. Goal of the project was not to deliver software but to present the results of modeling and an exhaustive evaluation of the errors which the best model made.

Roles involved

On the NLU team side a single person was project lead and data scientist. On the side of the client the CEO of the company was involved as well as the lead IT professional.

Project Outcome

Although the machine learning models were able to reach about 90% f1-score, the company decided to not pursue the integration of a NLU service into their system. The main reason they declared was that integrating machine learning models into their existing software was a complicated process requiring rethinking of the user experience and the interaction of the system with the machine learning components.

Most important Characteristics

Process - Application Development - User Experience Design The difficulty of redesigning the user experience of their software to accommodate for the characteristics of machine learning models led the CEO to decide to not pursue the solution at the given time.

Process - Application Design - Integration of AI/ML with traditional software Integrating a machine learning model into the existing software required rethinking existing processes and interfaces of the software and posed a challenge as the software was not originally designed with machine learning in mind. This would have significantly increased the costs of an integration project.

Artifact/Asset Management - Data - Data Quality The data quality was a large challenge for the NLU team as many documents were mislabeled, the class labels were not completely independent and there was a large skew in the frequency distribution of the labels. This influenced the choice of the model and the evaluation.

A.2.11 Detecting Medical Synonyms

In this project the client, a large German clinic chain, wanted a service which provided them with synonyms given a medical term.

Project Setting and Problem Description

Starting point for the analysis was a medical ontology with German terms as well as a set of German medical publications. The NLU team decided to crawl further public datasets with medical texts to enrich the training set for the machine learning models. Goal of the project was to deliver a report of the feasibility and a software which given a term yields synonyms for this term or at least related words.

Roles involved

On the side of the NLU team a project lead as well as a junior data scientist were involved. On the client side a project lead was involved, who was also a domain expert.

Project Outcome

Although the client was satisfied with the project no functioning synonym finder was created in the project. It turned out that the accuracy of the models was simply not good enough.

Most important Characteristics

Process - Project Ideation - Research In the beginning of the project the data scientist spend a lot of time on researching publications which already solved similar problems. The fact that there were only a handful of publications, which only partially solved the problem with a lot more data available, convinced the client that this problem was hard to solve and helped in managing the expectations.

Process - Project Ideation - Concept Design A missed opportunity in the project was tailoring a concept of the solution around the fact that the results would probably not be perfect. Using the solution as a recommendation system for humans rather than as a tool which automatically fills ontologies would have mitigated a lot of the problems with the bad model quality.

Process - Project Management - Stakeholder Management - Communication Clear and transparent communication of the intermediate steps in the project and the project results led to a good climate between the client and the NLU team even when the results were not as good as hoped for. At all times the client understood that the steps which the NLU team took made sense and the team made sure that the client would know useful next steps after the end of the project.

A.2.12 Information Extraction from Engineering News

The goal of this project was to build a service which would automatically extract relevant information about products from engineering news. This service should be integrated in a search engine by a German startup.

Project Setting and Problem Description

Starting point of the project was that the startup had already tried multiple machine learning approaches with other companies but were not satisfied with the results. They had accumulated a large dataset of over one million news articles. Further they had annotated relevant information in a few hundred articles, which should in the future be automatically extracted. It was unclear how the machine learning service should be integrated in the final software and further how users should interact with the system. This led to a very long initial phase in which both the NLU team and the client worked excessively on better understanding the business problem, user wishes and how to align the business problem with NLU concepts. Next, new annotations were required as the problem which was considered most relevant required different annotations than the ones already at hand. During the annotation process, the NLU team regularly trained intermediate models to verify how much more data was needed. The amount of data needed was significantly cut by the fact that the NLU team used already pre-trained models. One use case was even approached without annotation by taking an existing reading comprehension model and posing the problem as a reading comprehension task instead of an information extraction task.

Roles involved

On the side of the NLU team a project lead, a senior data scientist and a junior data scientist were involved. The senior data scientist mainly consulted the junior data scientist who took on a lot of responsibility in project execution but also in communication with the client. On the client side, the two founders of the startup were involved as well as an IT professional.

Project Outcome

The project concluded successfully. NLU services were delivered which recognized the required information and provided the team with a certainty score for the results, such that they could decide what to feed into their search engine and which results needed to be overhauled. Nevertheless, the startup decided to start another iteration of business problem definition as they recognized that they still did not understand the needs of their customers sufficiently.

Most important Characteristics

Process - Model Development - Model Reuse Being able to reuse already pre-trained models reduced the amount of labeled data necessary to meet the business needs and even enabled exploring tasks for which there were no labeled data.

Process - Project Ideation - Mutual Understanding Getting an understanding of the problem to be solved was a critical part of the project and took a significant amount of time. This could have been resolved more easily if end users were present in that phase of the project. The startup later decided to interview many potential customers in order to understand the need of them better.

Management/Governance - Organization - Strategy/Vision A major problem of the project was that the startup lacked a strategy of how to build their business and how to benefit from NLU solutions. It was evident that a text understanding model could help them but not in which way exactly.

A.2.13 Invoice Processing Benchmark

In this benchmark a major German software provider wanted to benchmark various state of the art approaches for extracting relevant information from invoices.

Project Setting and Problem Description

The software provider had their own data science department which was tasked with developing and integrating new models in their software. They wanted to use this project as a way to explore novel models based on large language models and to gain insights on how to train them on their tasks. The software provider had a dataset of about 80,000 labeled invoices in multiple languages. Other than working with the supervised dataset alone, multiple alternatives should be explored such as automatically annotating data with weak supervision or extending model knowledge with a type of soft business logic. The project was structured in an agile manner, i.e. there were 3-week long sprints after which results were presented and new tasks were defined. At the end of the project, services for all trained models should be provided, along with data processing and training pipelines. The project faced two personnel changes on the client side, i.e. the project lead changed twice.

Roles involved

On the NLU team side there was one project lead, two junior data scientists as well as one senior data scientist. On the client side there was a technical manager as well as a project lead.

Project Outcome

The outcome of the project was positive. The team managed to deliver end to end models which were on par with specialized models embedded in rule-based system and trained on larger datasets. A downside was that there was a huge delay in deliveries caused by a shift of the project lead on the client side. The new project lead changed some of the requirements shortly before ending the project. Another reason for the delay were issues with quality assurance. The pipelines for data processing and modeling were not sufficiently tested.

Most important Characteristics

Process - Application Development - Application Evaluation - Quality Assurance Quality assurance played a major role in delays of the project results. There were some bugs in the data and modeling pipeline which only became apparent after the project ended and the pipelines were tested on different datasets. During the project, the project lead did not calculate enough time to test and debug the pipelines.

Process - Model Development - Model Reuse Reusing existing pre-trained language models led to a huge benefit in the project, as this significantly increased the accuracy during fine-tuning. The easiness of adapting the pre-trained models enabled the team to quickly experiment with them and try several different models.

Management/Governance - Team Management - Team Organization The change in personnel on the side of the client led to various changes in the project scope and requirements and resulted in delays in the project delivery.

A.2.14 Receipt Classification

In this project the client was a small document management system provider who wanted to enrich their software with NLU components, namely a component which distinguishes different types of receipts.

Project Setting and Problem Description

The client wanted to tackle the problem of differentiating invoices, delivery receipts and order confirmations in a first step, which should serve as an initial proof-of-concept that NLU could help automating document-based processes. The company provided around 30,000 labeled documents. Since the budget of the project was small, the NLU team opted for an easy to implement baseline process in which they experimented with various hyper-parameters of a publicly available model. The model was also resource-friendly, i.e. it could easily run inference on the servers of the client's customers.

Roles involved

On the side of the NLU team a single person filled the roles of project lead and data scientist. On the customer side the CEO of the company was involved as well as the IT lead.

Project Outcome

Although the project results were great, with an f1-score of around 98%, the company decided to not further pursue the integration of the AI component as they did not see a compelling business case.

Most important Characteristics

Management/Governance - Resources - Budget Since the project was on a tight budget, the data scientist opted for an easy to implement and train model, which turned out to be a good decision. In a matter of days, the final model was presented with high f1-scores.

Process - Project Ideation - Acquisition The client and the NLU team should have used more time in the beginning to calculate the business case before engaging in the proof-of-concept. The information necessary for understanding that there was no compelling business case were clearly not dependent on the result of the proof-of-concept. That the client decided to start the proof-of-concept was also due to the low pricing hurdle.

Artifact/Asset Management - Data - Quality The good quality of the data made the project feasible with a low budget as the data scientist was not forced to spend a lot of time on data preprocessing.

A.2.15 Semantic Similarity for Legal Documents

This project was sponsored by a German startup which specialized in software for attorneys and judges. The software helps to structure cases by providing templates for letters and rulings but also by showing semantically relevant content for cases from a database of all documents associated with a case or similar cases. The goal of the NLU team was to develop a component which stored all documents and various segments of them and calculated the semantic similarity of them to a query.

Project Setting and Problem Description

The project was set up in a way that one law firm provided data for training and validating the models integrated in the component. Given the specific requirements in terms of inference speed and APIs, the NLU team decided to build a new component from scratch. The project required the development of models and the component, i.e. an application, simultaneously. The project was supposed to be accompanied by a lawyer to provide feedback on the results and the model quality, but this did not work out due to scheduling issues. Further the project ran into problems when the inference speed requirements could not be met and a developer on the client side developed a more lightweight component, explicitly tailored to their system. Finally, the staff in the NLU team was exchanged and a component was developed which better met the requirements and which was thoroughly tested.

Roles involved

In the beginning of the project, a project lead and two junior data scientists were involved in the project. Later, the project lead was exchanged and the junior data scientists were replaced by a senior data scientist and an MLOps engineer. On the client side the CEO, the technical lead and a lead developer of the startup were involved.

Project Outcome

The NLU team was able to deliver a component which met the requirements with good semantic matching results, however both the project budget as well as the project schedule were overshot significantly. The outcome was negative, since the customer developed their own solution and did not want to use the solution provided by the NLU team anymore.

Most important Characteristics

Team Organization A characteristic that influenced the problems of the project in a major way, was the team staffing. The junior data scientist responsible for the development of the application was still very inexperienced and had little to no guidance in the development process.

Requirements Elicitation The process of requirements elicitation in the beginning of the project was not done thoroughly enough and many requirements only became obvious after starting the project. This significantly influenced the duration of the project and the effort needed to finish it.

Quality Assurance The quality assurance measures for monitoring the whole project as well as the application development were insufficient, leading to severe issues when the first versions of the application were delivered to the customer. An issue which was revealed rather quickly was that the application needed way too many resources and had rather long inference times.

A.2.16 Social Media Mining for Cosmetics

The client in this project was the R&D department of a large cosmetics company. The company wanted to gain insights into the effects of their marketing campaigns as well as into the perception of ingredients by relevant target groups.

Project Setting and Problem Description

The goal of this project was to provide insights from social media such that the cosmetics company could use them for product development as well as their marketing campaigns. The company was particularly interested in how the topic of allergies was discussed in social media and if their marketing campaigns, which explicitly marketed their products as being allergy friendly, made a significant impact on the online discussion of their products. As a starting point, the NLU team was provided with a dataset crawled from different social media sources such as Facebook, Twitter, blogs and forums. The team put a lot of effort into preprocessing and also started to build a terminology with relevant terms considering ingredients, brands and products with the help of the client. The team opted for two approaches. One approach was to structure the millions of documents via topic modeling and the other approach was to detect ingredient mentions via named entity recognition. In order to visualize the results of the topic modeling, the team implemented a visualization tool which allowed the client to interactively explore the detected topics and relevant terms and documents belonging to each topic.

Roles involved

On the side of the NLU team a project lead, a senior data scientist and a junior data scientist were involved. On the client side a single person acted as project lead and domain expert.

Project Outcome

The project had a positive outcome. The client discovered a new ingredient which the community praised for its properties and was not used by any of the competition. Further, the client understood that their marketing campaigns should not focus on allergies as other brands were more successful with marketing campaigns focusing on more positively associated topics such as current trends.

Most important Characteristics

Technology and Infrastructure - Visualization Tools Developing a visualization tool for the topic modeling results significantly increased the understanding of the client. The visualizations served as a center for communicating and discussing results and helped the client to dig deeper into topics of interest.

Process - Project Management- Stakeholder Management - Communication The client and the NLU team had regular meetings for discussing their findings based on the visualizations. This mode of communication helped the client to get insights and to steer the focus of the analysis by the NLU team.

Process - Data Processing - Data Augmentation Augmenting the social media posts with structured data in the form of a terminology enabled the NLU team to automatically label ingredient mentions and train a model on recognizing novel ingredients.

A.2.17 Interview: Contract Analytics

In this project, the client was a large auditing firm, which wanted to build a solution for the automated analysis of lease contracts. A new regulation in Germany forced companies to list lease contracts in their balance sheets.

Project Setting and Problem Description

This project was in the early days of the deep learning hype, so the expectations of deep learning solutions were very high and the requirements and capabilities of these solutions were still widely unknown to most companies. The problem that the client wanted to solve in this project was that many companies had paper-based lease contracts, which they needed to include in their balance sheets. Together, the customers of the client had over 300,000 such documents. There was no existing solution to the problem and not a lot of budget to develop one. The NLU team was supposed to build a document processing service, composed of an OCR and an information extraction pipeline and software developers of the client would build a software which integrates the machine learning services and offers a visualization and user interaction components. The team additionally decided to create a so called business logic layer, i.e. a layer which verifies or corrects results from the machine learning models. At the start of the project, the client had no data available and little knowledge about machine learning. The NLU team spend some time in the beginning to explain how machine learning works in general, to make the requirements of such a project more transparent. On the other hand, the client offered to explain details about the leasing domain so that the data scientists could understand the problem better. A major obstacle was, that the project started without any data, let alone annotated data. Hence, the decision was made to create a dataset. During the course of the project, it became obvious that the labeling of the contracts was a tedious process, so the NLU team opted for a template generation solution: Given labeled templates of lease contracts, the solution would fill the template with values such as company names to create synthetic training data. Based on 300 templates over 3,000 contract samples were generated and used for training.

Roles involved

The project contained multiple parties, two teams worked on the different aspects of the machine learning pipeline, i.e. one team worked on the OCR while the NLU team worked on the information extraction pipeline and the business logic. The NLU team consisted of a project lead, a senior data scientist and two junior data scientists. On the client side a managing partner was involved, a project lead and two domain experts. The domain experts consulted the NLU team on questions and helped with the labeling process.

Project Outcome

The project was only partially successful, i.e. there was a working demonstrator with decent machine learning results. The main benefit came from a business logic, which corrected the machine learning models and produced alternative results. It significantly reduced processing time of contracts. However, the solution did not come into use on the customer side.

Most important Characteristics

The interviewee stated that the application architecture design and especially the separation of concerns were positive characteristics of the project. However, a lot of characteristics affected the project negatively, such as the difficult stakeholder management and expectation management, lacking skills on the team side, the maturity and readiness of the team, the lack of data, the overall team organization and the tedious data augmentation processes.

A.2.18 Interview: Analysis of Court Rulings on Theft Offenses

The client in this project was the legal department of a university, which wanted to establish a comparability between court rulings. To achieve this, relevant information should be extracted from court rulings and then submitted to a database, such that they can be queried and compared easily.

Project Setting and Problem Description

There is no database Germany, which contains structured meta-data of all legal cases. Hence, it is very difficult to find cases with similar verdicts and offenses. Further, it appears that there is a discrepancy in the verdict when comparing similar cases in northern and southern Germany. In order to quantify the difference between northern and southern Germany and to start establishing a legal database, the client wanted to perform a proof-of-concept. The goal was to see if this could be automated with NLU methods. The client had access to a list of relevant court rulings and the initial scope was limited to court rulings on theft offenses from a limited number of courts in Germany. The NLU team set up their annotation tool such that the domain experts could annotate the relevant information in the court rulings. Due to time constraints only slightly more than 100 court rulings were annotated by the domain experts. Pre-trained language models were used for classification of the documents as well as information extraction. The models were extended with a trainable logical layer which was fed with expert knowledge and served as a post-processing step to improve the model results. Finally, the results of the analysis were integrated in a visual demonstrator.

Roles involved

On the side of the NLU team, a project lead, who also served as senior data scientist, as well as a junior data scientist were engaged in the project. On the client side a project lead as well as domain experts worked on the project.

Project Outcome

The project ended successfully with good modeling results as well as successful showcases of the demonstrator.

Most important Characteristics

The interviewee stated that the characteristics which influenced the project positively were the involvement of domain experts, the availability of an annotation tool, the reuse of pre-trained models and the creation of an visualization tool. The project was negatively affected by the effort and time necessary for data augmentation.

A.2.19 Interview: Information Extraction from Medical Reports

In this project, the client was a large German hospital. The goal of the project was to extract information from various report types in a structured manner and then, use the structured information to generate discharge summaries. As of writing, only one of the use cases has ended. This use case has a focus on information extraction from pathology reports.

Project Setting and Problem Description

The project started with no annotated data, hence the NLU team set up their annotation tool. In the beginning, a large amount of time was spent on defining annotation guidelines and iteratively improving them, while educating medical domain experts on how to use the annotation tool. Along with the annotation work, a process was established to generate bug reports and feature requests to improve the overall quality of the annotation tool and the annotation process itself. Pathology reports for 300 patients were completely anonymized and then annotated with named entity recognition mentions for dozens of entity types and relations between the entities. The NLU team tried various pre-trained named entity recognition models. Lastly, the model which performed best was a model which was specifically pre-trained on the medical domain, highlighting the benefit of training language models on domain specific data. For both the named entity recognition and the relation extraction, not only the pre-trained models were reused, but also the respective model pipelines. The resulting models were integrated in the annotation tool to guide the annotation of the next documents.

Roles involved

On the side of the NLU team a project lead and a junior data scientist were involved in the project, as well as the product owner of the annotation tool and an MLOps engineer who worked on the annotation tool. On the client side a project lead was involved as well as a data scientist and a team of domain experts who were responsible for the annotations.

Project Outcome

The use case was concluded successfully and a version of the annotation tool with the embedded models was delivered. The models reached sufficiently high f1-scores to help the annotators in the process of annotating.

Most important Characteristics

The interviewee stated, that there were multiple characteristics which served as success factors of the project: a good mutual understanding and data understanding in terms of availability, volume, security and other legal aspects. Regular communication, clear team organizations, good expectation management, the reuse of models and software, clear goal and milestone definitions, a good work culture and the involvement of stakeholders such as domain experts and users. Particular challenges were the privacy concerns regarding the data and restaffing of the project. Other characteristics which were important were the availability of an annotation tool, a visualization tool, the integration of solutions in the system of the client and the data augmentation process.

A.2.20 Interview: Classification of Repair Reports

In this project, the client was a company responsible for repairs of wind turbines. Their goal was to automatically classify repair reports into categories describing the damages.

Project Setting and Problem Description

The client provided the NLU team with 7,000 documents and their labels. The problem in this project was a classification problem and there were hundreds of classes. There was a strong imbalance in the data and many classes were underrepresented. The NLU team tried various methods, however classic machine learning models, i.e. an SVM, performed better than state-of-the-art approaches. One important aspect in modeling was, that the client wanted to get information about the uncertainty of predictions. This influenced the modeling decisions. The best-performing models were integrated in services so that the client could test their performance. Unfortunately, end users were not directly involved in the project. A challenge in the project was that the NLU team had strict time constraints and had to end the project at an earlier date than anticipated in the beginning.

Roles involved

On the side of the NLU team a project lead and a data scientist were involved in the project. On the client side a project lead was involved.

Project Outcome

The use case was concluded successfully and a version of the service was deployed for usage. The client was happy with the results although they wished for a longer project with more iterations.

Most important Characteristics

The interviewee listed multiple relevant characteristics, which influenced the project. The project management, the communication and collaboration within the project as well as including skilled senior members in the project team were named positive characteristics. The project was negatively impacted by the overall team organization, including changing staff, tight time constraints which led to difficult planning and scheduling, a scarcity of resources and an unclear mutual understanding of goals and deliverables between the client and the team.

A.3 Additional Information about the Evaluation of Methodologies

A.4 Additional Information about the Methodology

A.4.1 Comparing FM-DSM to Other Methodologies

We show the phases of a selection of methodologies from Chapter 3 in Figure A.25. We excluded the methodologies which do not explicitly define project phases. We align the project phases of each methodology with the de facto standard methodology CRISP-DM, which is highlighted in green. In blue we highlight phases which are not part of CRISP-DM at all. Such phases might overlap with CRISP-DM phases or extend them. Note that this form of comparison does not necessarily indicate the completeness of a methodology, especially since many methodologies define phases on different levels. This becomes especially apparent when considering our evaluation of the Process characteristics in Chapter 4. Microsoft TDSP has 6 phases, just like CRISP-DM, but provides a more complete picture of the process of modern data science projects, while Data2Value has 23 phases and lacks detail in

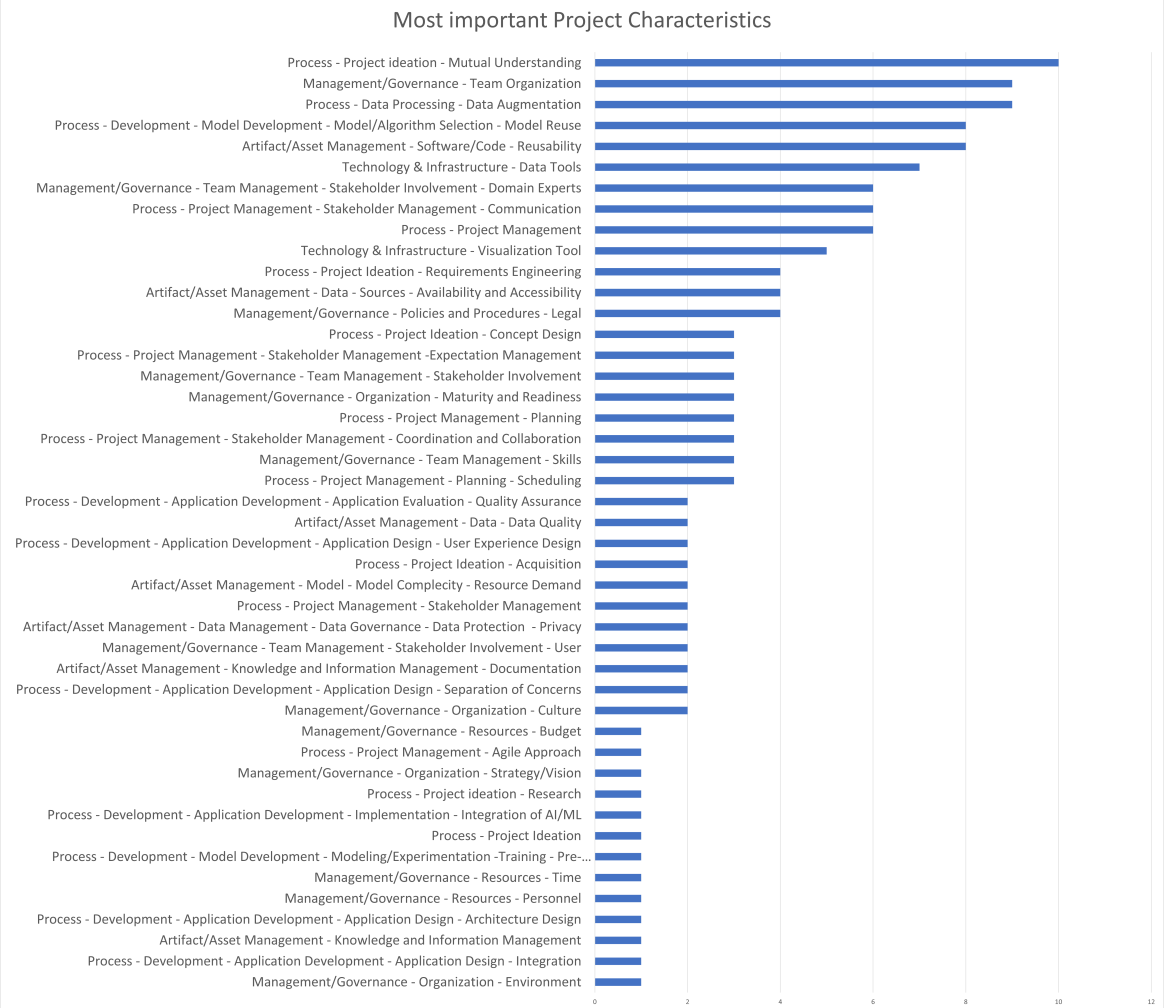

Figure A.24: The number of projects in which characteristics were mentioned as being most important to the project process and outcome.

many of the phases, i.e. many of its so called phases are rather on the granularity of tasks in the CRISP-DM methodology.

A.5 Detailed Description of Artifacts, Roles and Technologies in the Phases of Our Methodology

In this section we provide a detailed overview of the roles which are active in each phases, their activities, the artifacts as well as the tools and techniques which are helpful in each phase.

A.5.1 Mutual Understanding Phase

Here, we provide detailed descriptions of the roles, artifacts as well as tools and techniques in the Mutual Understanding phase.

Active Roles in the Mutual Understanding Phase

Following, we list the roles which are active in the Mutual Understanding phase and their tasks:

- **Project Lead**: estimates time and effort for the contract, obtains business understanding, identifies key stakeholders, assembles team, creates a project plan, schedules meetings, establishes quality assurance processes
- **Business Stakeholder**: defines business goals, participates in use case workshops, provides business understanding, helps defining KPIs and goals
- **Business Development**: engages with customer, introduces NLU solutions and demos to customer, helps identifying business goals, analyzes customer maturity and readiness
- **Data Scientist**: performs initial assessment of use cases, participates in use case workshops, gets initial insights into data, helps estimating time and effort, obtains initial business understanding, creates NLU terminology, researches existing solutions
- **Solution Architect**: performs initial assessment of use cases, participates in use case workshops, helps estimating time and effort, researches existing solutions, maps business problems to NLU tasks, designs a solution and integration concept
- **ML Engineer**: participates in use case workshops, helps estimating time and effort
- **User Experience Designer**: participates in use case workshop, interviews user and domain experts, creates a user experience design concept
- **Domain Expert**: participate in use case workshop, provide business understanding, give insights into business processes, create business/domain terminology
- **End User**: participate in use case workshop, provide business understanding, give insights into business processes
- **Client IT Expert**: participate in use case workshops

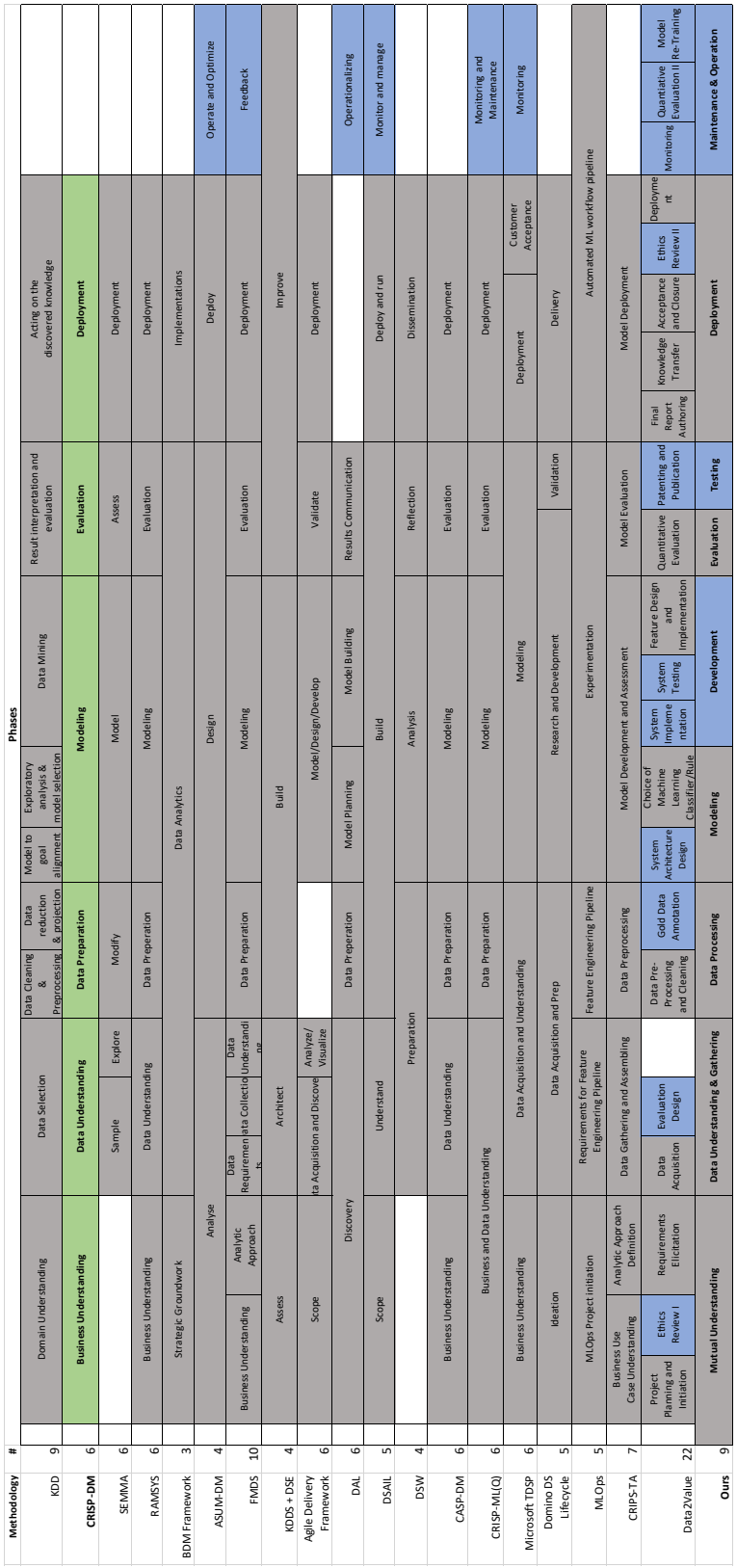

Figure A.25: Comparison of the phases of different methodologies.

- **Team IT Expert**: participate in use case workshops, analyze security requirements
- **Auditors**: perform ethics review, review regulatory requirements, review security requirements
- **Legal Expert**: review legal requirements, approve contract
- **Requirements Engineer**: participate in use case workshops, define functional and non-functional requirements, prioritize requirements, document requirements
- **Team Lead**: approve project and contract, assign resources

Artifacts of the Mutual Understanding Phase

The following artifacts are produced or used in the Mutual Understanding phase.

1. **Contract**: The signed contracts containing all relevant information regarding deliverables, the desired project outcome, available budget, time and rights. It might include NDAs, order data processing and other legal information.
2. **Charter Document**: The charter document is updated throughout the whole project. It contains information on the project and business background, the scope, the value proposition, the agreed upon metrics and KPIs, the project organization and the communication and meeting plan.
3. **Inventory of Resources**: The inventory of resources lists available personnel, data, computing resources and software. Here the team also specifies which components and models they will be reusing.
4. **Use Case Descriptions**: Detailed descriptions of use cases, potentially with UML-like graphics for easy understanding [119].
5. **Requirements Document**: Contains all requirements, whether functional or non-functional. Requirements include the quality of results, security, legal issues, rights to use data and many more.
6. **Solution Design**: This document will be updated with insights from the later phases. At this phase it contains an initial design for a solution architecture including model hypothesis, the software architecture and interfaces. It also contains links to solutions from previous projects as well as to reusable models, software components and datasets. The document should highlight the mapping from the business problems and use cases to the architecture.
7. **Project Plan**: The project plan schemes all project phases, their duration and schedule, the required resources, required inputs and dependencies and produced outputs. It should make iterations visible, analyze dependencies between time schedule and risks and contingency plans. It also contains milestones and review points.
8. **Domain Knowledge Document**: This document contains domain knowledge, important business and NLU terminologies, process and business knowledge and FAQs from the team and the client.

9. **Initial Insights**: This document contains initial insights from data analysis and potentially even modeling if the customer supplied data before the project start.
10. **Data Sample**: In the initial phases of the project the client might already transfer data samples for preliminary data understanding, assessment and analysis.

Tools and Techniques for the Mutual Understanding Phase

The following tools and techniques are useful for the Mutual Understanding phase. They consider project management activities and cloud platforms to set up projects on.

1. **Tools for Project Management:** PowerPoint, Email, Confluence, JIRA, Workshops, Interviews, Word
2. **Cloud platforms:** AWS, Azure, IBM Cloud, Google Cloud

A.5.2 Data Gathering and Understanding Phase

Here, we provide detailed descriptions of the roles, artifacts as well as tools and techniques in the Data Gathering and Understanding phase.

Active Roles in the Data Gathering and Understanding Phase

Following, we list the roles which are active in the Data Gathering and Understanding phase and their tasks:

- **Project Lead**: revise planning, communicate insights, communicate plan adjustments
- **Business Stakeholder**: provide feedback about data insights
- **Data Scientist**: analyze data, create statistics, inspect data samples, assess data quality, generate hypothesis about the data
- **Solution Architect**: revise concept design
- **Data Engineer**: create data connectors and crawlers, create database design
- **Data Master**: help identifying data sources, define access and usage policies, setup data hub
- **Domain Expert**: determine relevant data sources, provide insights into data, provide feedback about data insights
- **End User**: provide insights into data
- **Client IT Expert**: help identifying local data sources
- **Team IT Expert**: set up infrastructure for data processing and storing
- **Requirements Engineer**: refine data requirements
- **Legal Expert**: check data licenses and ownership

Artifacts of the Data Gathering and Understanding Phase

The following artifacts are produced or used in the Data Gathering and Understanding phase.

1. **Data Collection Report**: Contains information about the data sources, the storage locations, the acquired datasets, the methods used to acquire them with links to the code
2. **Data Dictionary**: A collection of domain and business knowledge about data, data-specific terminology and FAQs about the data.
3. **Data Description**: Information about the data structure, formats, quantities, ownership, sensitivity and authorship.
4. **Data Exploration Report**: Describes the methods used to explore the data, links to the code and contains findings and hypothesis.
5. **Data Quality Report**: Presents the results of the data assessment with regards to data integrity, fairness, and so on.
6. **Updated Project Plan**: The updated project plan, considering all the findings from the data collection, exploration and assessment.
7. **Updated Solution Design**: Update of the solution design considering all changes which have to be made based on the available data and novel hypothesis.
8. **Data**: The collected data, including all meta-data, stored in its raw form. The data is linked to the reports and the software used to obtain it.
9. **Data Gathering and Exploration Software**: All crawlers, exploration and assessment code is shared in the Software Hub to ensure reproducibility and reuse.

Tools and Techniques for the Data Gathering and Understanding Phase

The following tools and techniques are useful for the Data Gathering and Understanding phase. They consider data processing, exploration and visualization activities.

- **Tools for data processing:** Hadoop, Spark, Kafka
- **Tools for data exploration and visualization:** pyLDAvis, Wordcloud, pyplot, seaborn, BERTViz, Huggingface Spaces, jupyter notebooks
- **Techniques for data exploration:** LDA, BERTopic

A.5.3 Data Cleaning and Preprocessing Phase

Here, we provide detailed descriptions of the roles, artifacts as well as tools and techniques in the Data Gathering and Understanding phase.

Active Roles in the Data Cleaning and Preprocessing Phase

Following, we list the roles which are active in the Data Cleaning and Preprocessing phase and their tasks:

- **Project Lead**: communicate results
- **Data Scientist**: define annotation scheme, create data splits, engineer features
- **Data Engineer**: create data pipelines, transform data, establish data scorers, clean data, define annotation scheme, commit data, document data processing steps, preprocess data, review code, commit code
- **Software Engineer**: review code
- **Domain Expert**: help identifying noise in data, annotate data

Artifacts of the Data Cleaning and Preprocessing Phase

The following artifacts are produced or used in the Data Cleaning and Preprocessing phase.

1. **Data Transformation Report**: Describes decisions regarding the format, the transformations and integration of datasets.
2. **Data Cleaning Report**: Describes the decisions regarding cleaning and filtering of the dataset.
3. **Data Annotation Guidelines**: Serves as guideline for annotators and describes the annotation scheme with detailed explanations how to annotate and annotation samples. Further explains choices of annotation tools and configurations, rules and models for automated annotations.
4. **Data Preprocessing Report**: Describes preprocessing steps and tools, dataset reduction choices and methods, dataset augmentation methods and model-specific transformations.
5. **Preprocessed Data**: The preprocessed dataset after cleaning, annotation and preprocessing.
6. **Dataset Splits**: Documents the splits of the dataset for training, validation and testing.
7. **Features**: Derived attributes, newly created attributes, vectorized text, etc.
8. **Data Preprocessing Software**: The scripts and pipelines created for data preprocessing are committed to the Software hub for reuse.

Tools and Techniques for the Data Cleaning and Preprocessing Phase

The following tools and techniques are useful for the Data Cleaning and Preprocessing phase. They consider data preprocessing and annotation activities as well as common data formats.

- **Tools for data preprocessing and cleaning:** NLTK, SpaCy, TextBlob, Pattern
- **Tools for data annotation:** Prodigy, Snorkel, doccano, inception, Argilla
- **Common Data Formats:** Apache UIMA CAS, Apache Parquet, JSON

A.5.4 Modeling Phase

Here, we provide detailed descriptions of the roles, artifacts as well as tools and techniques in the Data Gathering and Understanding phase.

Active Roles in the Modeling Phase

Following, we list the roles which are active in the Modeling phase and their tasks:

- **Project Lead**: communicate modeling decisions, communicate test strategy
- **Data Scientist**: research state-of-the-art, select models, design model tests, implement rules, evaluate models, perform code reviews
- **Solution Architect**: map business problems to modeling techniques
- **Software Engineer**: perform code reviews
- **ML Engineer**: implement training pipeline, set up training infrastructure, review code, commit code
- **Domain Expert**: assess model performance, inspect model results
- **End User**: assess model performance, inspect model results
- **Legal Expert**: check licenses of reused models

Artifacts of the Modeling Phase

The following artifacts are produced or used in the Modeling phase.

1. **Modeling Decisions**: Describes the alignment and choice of models and rules with the respective tasks and business problems.
2. **Experiment Documentation**: Documents all experiments, choices of parameters and results.
3. **Updated Solution Design**: An updated solution design based on the modeling choices and results.
4. **Model Cards**: Model cards as suggested in [190] to describe the models and their properties.
5. **Updated Project Plan**: The project plan is updated based on the modeling results.
6. **Models and Rules**: The models and rules are committed to the Model Hub.
7. **Modeling Software**: Training and inference pipelines are committed to the Software Hub.

Tools and Techniques for the Modeling Phase

The following tools and techniques are useful for the Modeling phase. They consider modeling activities as well as public model APIs.

- **Tools for ML:** Scikit-Learn, Tensorflow, Torch, JAX, Azure ML Studio, Keras, Spark, MLLib, XGBoost
- **Tools for modeling**: NLTK, Huggingface Transformers, SpaCy, TextBlob, gensim, CoreNLP, PyNLPI, AllenNLP, Polyglot, Stanza
- **Tools for workflow orchestration:** Apache Airflow, Kubeflow, Luigi, SageMaker Pipelines, Azure Pipelines, Apache UIMA, Flyte, Prefect
- **APIs for public LLMs:** OpenAI GPT 3/4 API, Aleph Alpha Luminous API, Google PaLM API

A.5.5 Application Development Phase

Here, we provide detailed descriptions of the roles, artifacts as well as tools and techniques in the Application Development phase.

Active Roles in the Application Development Phase

Following, we list the roles which are active in the Application Development phase and their tasks:

- **Project lead**: revise planning, communicate plan
- **Business Stakeholder**: confirm revised project plan
- **Data Scientist**: support model integration
- **Solution Architecture**: revise design, define interfaces
- **Software Engineer**: research solutions, develop components, customize components, test software, package software, commit software, implement business logic and safeguards, test business logic and safeguards
- **ML Engineer**: support model pipeline integration
- **Product Owner**: understand product requirements, translate requirements to product changes, schedule implementation
- **Data Engineer**: support data pipeline integration
- **User Experience Designer**: revise user experience design
- **Domain Expert**: support business logic and safeguard development
- **End User**: support business logic and safeguard development

- **IT Experts**: define interfaces
- **Legal Experts**: analyze software licenses
- **Requirements Engineer**: refine requirements

Artifacts of the Application Development Phase

The following artifacts are produced or used in the Application Development phase.

1. **Updated Solution Architecture**: Updated solution architecture after revisiting requirements and taking into account results from other phases. Containing models, rules, data and model pipelines as well as interface descriptions.
2. **Business Logic Documentation**: Describes the business logic and the domain expertise behind it.
3. **Documentation**: Documentation for any new software that is designed, the business logic as well as for the complete application.
4. **Licenses**: A document containing all the licenses of third-party software.
5. **Test Design and Results**: Describes the design and the results of the various tests.
6. **Application Software**: The application itself and all the code written for implementation, building, testing and deployment is committed to the Software Hub. Also, the compiled and containerized artifacts are committed together with dependency files.

Tools and Techniques for the Application Development

The following tools and techniques are useful for the Application Development phase. They consider programming languages, environments and container orchestration.

- **Tools for container orchestration:** Kubernetes, Docker Swarm, Apache Mesos, Docker Compose
- **Programming Languages:** Programming Languages: Python, R, Java, Julia, Scala, SQL, C, C++, GO
- **Programming Environments:** R Studio, VSCode, Jupyter Notebook, PyCharm, Eclipse

A.5.6 Application Testing Phase

Here, we provide detailed descriptions of the roles, artifacts as well as tools and techniques in the Application Testing phase.

Active Roles in the Application Testing Phase

Following, we list the roles which are active in the Application Testing phase and their tasks:

- **Project Lead**: plan application updates, communicate update plan
- **Business Stakeholder**: verify update plan
- **Solution Architect**: assess test results, derive measures
- **Software Engineer**: perform application tests, integrate application in customer system, perform integration tests, improve application
- **End User**: perform application tests
- **Client IT Experts**: support application integration, support integration tests
- **Team IT Experts**: set up test infrastructure

Artifacts of the Application Testing Phase

The following artifacts are produced or used in the Application Testing phase.

1. **Deployment Documentation**: Documentation of the test deployment to derive a plan for the actual deployment.
2. **Test Design Descriptions**: Description of the application, integration and user tests.
3. **Test Results**: Documentation of the test results.
4. **User Feedback**: Documentation of the user feedback.
5. **Updated Project Plan**: The project plan is updated based on the results of testing and user acceptance.
6. **Updated Software**: The updated software is committed.

Tools and Techniques for the Application Testing Phase

The following tools and techniques are useful for the Application Testing phase. They consider testing activities.

- **Tools for Software Testing:** Selenium, Appium, Eggplant, Watir, Tosca

A.5.7 Evaluation Phase

Here, we provide detailed descriptions of the roles, artifacts as well as tools and techniques in the Evaluation phase.

Active Roles in the Evaluation Phase

Following, we list the roles which are active in the Evaluation phase and their tasks:

- **Project Lead**: update project plan, communicate project plan
- **Business Stakeholder**: make deployment decision, approve updated project plan
- **Data Scientist**: evaluate application, investigate model behavior in application
- **ML Engineer**: investigate model pipeline behavior
- **Data Engineer**: investigate data pipeline behavior
- **Domain Expert**: perform safeguard tests, evaluate application
- **End User**: perform safeguard tests, evaluate application

Artifacts of the Evaluation Phase

The following artifacts are produced or used in the Evaluation phase.

1. **Application Evaluation**: The evaluation design and results with regards to the selected performance measures and business KPIs.
2. **Updated Project Plan**: The project plan is updated based on the evaluation results and the decision whether to deploy or to continue development.

Tools and Techniques for the Evaluation Phase

The following tools and techniques are useful for the Application Testing phase. They correspond to the experiment tracking tools of Subsection 5.2.2.

- **Experiment tracking solutions:** Neptune.ai, Weights & Biases, Comet, Sacred, MLflow, TensorBoard

Active Roles in the Deployment Phase

Following, we list the roles which are active in the Deployment phase and their tasks:

1. **Project Lead**: schedule deployment, perform closing workshop, handover artifacts, perform lessons learned workshop
2. **Business Stakeholder**: participate in closing workshop, approve project end
3. **Business Development**: prepare marketing material
4. **Data Scientist**: participate in lessons learned workshop, clean up artifacts
5. **Solution Architect**: participate in lessons learned workshop

6. **Software Engineer**: educate end users, hand over application, participate in lessons learned
7. **ML Engineer**: schedule monitoring and maintenance, participate in lessons learned workshop, clean up artifacts
8. **Data Engineer**: participate in lessons learned workshop, clean up artifacts
9. **Data Master**: clean up artifacts
10. **End User**: participate in user training
11. **IT Experts**: schedule deployment, set up target system, perform integration tests, take over application
12. **Requirements Engineer**: participate in lessons learned workshop
13. **Team Lead**: participate in lessons learned workshop

A.5.8 Deployment Phase

Here, we provide detailed descriptions of the roles, artifacts as well as tools and techniques in the Deployment phase.

Artifacts of the Deployment Phase

The following artifacts are produced or used in the Deployment phase.

1. **Deployment Plan**: The plan for application deployment and roll-out.
2. **Test Results**: Test results after deploying the solution on the production system.
3. **Monitoring and Maintenance Plan**: Monitoring and maintenance plan for the time after the system goes live.
4. **Final Reports and Presentations**: The final reports and presentations containing an overview of the application, insights, modeling decisions and results.
5. **Lessons Learned Documentation**: Internal document to summarize the project, its pitfalls, best practices and the links to reusable artifacts.

Tools and Techniques for the Deployment Phase

The following tools are useful for the Deployment phase.

- **Tools for CI/CD:** Bamboo, Jenkins

A.5.9 Maintenance and Operation Phase

Here, we provide detailed descriptions of the roles, artifacts as well as tools and techniques in the Maintenance and Operation phase.

Active Roles in the Maintenance and Operation Phase

Following, we list the roles which are active in the Maintenance and Operation phase and their tasks:

- **Project Lead**: plan updates
- **Data Scientist**: investigate faulty outputs, analyze model behavior, retrain models
- **Software Engineer**: analyze user behavior, improve application
- **Domain Expert**: investigate faulty outputs, correct model outputs
- **End User**: investigate faulty outputs, correct model outputs
- **Client IT Experts**: monitor application, support operation, roll out updates

Artifacts of the Maintenance and Operation Phase

The following artifacts are produced or used in the Maintenance and Operation phase.

1. **Support Documents**: Documents to support users and IT experts when using the application, including FAQs.
2. **Monitoring Logs**: Logs from monitoring the application output and the usage.
3. **Evaluation Reports**: Reports which describe the results of the evaluation of the application in productive use.
4. **Production Data**: Data which is gathered during the usage of the application for logging and retraining purposes.
5. **Updated Software**: Updated version of the code and the application.
6. **Updated Models**: Updated version of the models in usage.

Tools and Techniques for the Maintenance and Operation Phase

The following tools are useful for the Maintenance and Operation phase.

- **Tools for monitoring:** Prometheus with grafana, ELK, TensorBoard, Kubeflow, MLflow, AWS Sagemaker

Bibliography

[1] *Data Analytics Market Size - Global Industry, Share, Analysis, Trends and Forecast 2022 - 2030*, https://www.acumenresearchandconsulting.com/data-analytics-market, Accessed: 2023-08-03 (cit. on p. 1).

[2] R. Wirth and J. Hipp, "Crisp-dm: towards a standard process modell for data mining", 2000 (cit. on pp. 1, 12, 38–40, 49, 56, 96, 97, 134–136, 203, 204).

[3] J. Charvat, *The Project management methodologies: Selecting, implementing and supporting methodologies and processes for projects*, Hoboken, NJ: John Wiley & Sons Inc., 2003 (cit. on pp. 1, 31, 38, 39).

[4] J. Saltz, N. Hotz, D. Wild and K. Stirling, *Exploring project management methodologies used within data science teams*, Americas Conference on Information Systems 2018: Digital Disruption, AMCIS 2018 (2018) 1 (cit. on p. 1).

[5] M. Das, R. Cui, D. Campbell, G. Agrawal and R. Ramnath, "Towards methods for systematic research on big data", 2015 2072 (cit. on pp. 2, 51, 226, 228).

[6] R. Jurney, *Agile Data Science 2.0*, O'Reilly Media, Inc., 2017, ISBN: 9781491960110, arXiv: arXiv:1011.1669v3 (cit. on pp. 2, 37, 51, 106, 127, 133–136, 146, 166, 231, 233).

[7] Microsoft, *What is the Team Data Science Process?*, https://docs.microsoft.com/en-us/azure/architecture/data-science-process/overview, Accessed: 2022-06-22 (cit. on pp. 2, 51, 110, 125, 134, 243, 244).

[8] G. Symeonidis, E. Nerantzis, A. Kazakis and G. A. Papakostas, *MLOps - Definitions, Tools and Challenges*, 2022 IEEE 12th Annual Computing and Communication Workshop and Conference, CCWC 2022 (2022) 453, arXiv: 2201.00162 (cit. on pp. 2, 114, 261).

[9] *This Week In AI Stats: Up To 50% Failure Rate In 25% Of Enterprises Deploying AI*, https://www.forbes.com/sites/gilpress/2019/07/19/this-week-in-ai-stats-up-to-50-failure-rate-in-25-of-enterprises-deploying-ai/?sh=6cc6c3c872ce, Accessed: 2023-06-17 (cit. on p. 2).

[10] C. Haskins, *Survey: 96% of Enterprises Encounter Training Data Quality and Labeling Challenges in Machine Learning Projects*, https://www.businesswire.com/news/home/20190523005183/en/Survey-96-of-Enterprises-Encounter-Training-Data-Quality-and-Labeling-Challenges-in-Machine-Learning-Projects, Accessed: 2023-06-17, 2019 (cit. on p. 2).

[11] J. S. Saltz, "The need for new processes, methodologies and tools to support big data teams and improve big data project effectiveness",
2015 IEEE International Conference on Big Data (Big Data), 2015 2066
(cit. on pp. 2, 37, 39, 50).

[12] G. Paass and S. Giesselbach, *Foundation Models for Natural Language Processing: Pre-trained Language Models Integrating Media*,
Foundation Models for Natural Language Processing (2023)
(cit. on pp. 2, 7, 19, 21, 26, 28, 57).

[13] OpenAI, *GPT-4 Technical Report*, ArXiv **abs/2303.08774** (2023) (cit. on pp. 2, 29).

[14] Y. Li et al., *Competition-level code generation with AlphaCode*, Science **378** (2022) 1092,
eprint: https://www.science.org/doi/pdf/10.1126/science.abq1158,
URL: https://www.science.org/doi/abs/10.1126/science.abq1158 (cit. on p. 2).

[15] *ChatGPT sets record for fastest-growing user base - analyst note*,
https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/l, Accessed: 2023-06-12 (cit. on p. 2).

[16] F. Martínez-Plumed et al.,
CRISP-DM Twenty Years Later: From Data Mining Processes to Data Science Trajectories,
IEEE Transactions on Knowledge and Data Engineering **33** (2021) 3048 (cit. on pp. 3, 133).

[17] J. L. Leidner,
Data-to-Value: An Evaluation-First Methodology for Natural Language Projects, (2022) 1,
arXiv: 2201.07725, URL: http://arxiv.org/abs/2201.07725
(cit. on pp. 3, 52, 78, 116, 140, 159, 166, 265, 266).

[18] J. S. Saltz, I. Shamshurin and K. Crowston,
Comparing data science project management methodologies via a controlled experiment,
Proceedings of the Annual Hawaii International Conference on System Sciences
2017-January (2017) 1013, ISSN: 15301605 (cit. on pp. 3, 36, 37, 200).

[19] L. Adilova, S. Giesselbach and S. Rüping,
Making Efficient Use of a Domain Expert's Time in Relation Extraction,
ArXiv **abs/1807.04687** (2018) (cit. on pp. 6, 191).

[20] S. Giesselbach, I. Reid and S. Rüping,
"Social media analytics and community policing: challenges and solutions",
Changing Communities, Changing Policing, Neuer Wissenschaftlicher Verlag, 2018 107
(cit. on pp. 6, 191).

[21] L. von Rueden et al., *Informed Machine Learning - A Taxonomy and Survey of Integrating Prior Knowledge into Learning Systems*,
IEEE Transactions on Knowledge and Data Engineering (2021) 1, ISSN: 2326-3865,
URL: http://dx.doi.org/10.1109/TKDE.2021.3079836 (cit. on pp. 6, 190).

[22] V. Gupta, K. Beckh, S. Giesselbach, D. Wegener and T. Wirtz, “Supporting verification of news articles with automated search for semantically similar articles”, *Proceedings of the workshop Reducing Online Misinformation Through Credible Information Retrieval (ROMCIR 2021) co-located with 43rd European Conference on Information Retrieval (ECIR 2021), Lucca, Italy (On-line), April 1, 2021*, ed. by F. Saracco and M. Viviani, vol. 2838, CEUR Workshop Proceedings, CEUR-WS.org, 2021 47, URL: http://ceur-ws.org/Vol-2838/paper5.pdf (cit. on pp. 7, 29).

[23] B. Kirsch et al., “Using Probabilistic Soft Logic to Improve Information Extraction in the Legal Domain.”, *LWDA*, 2020 76 (cit. on pp. 7, 81, 192).

[24] D. Wegener, S. Giesselbach, N. Doll and H. Horstmann, “Introducing the NLU Showroom: A NLU Demonstrator for the German Language”, *3rd Conference on Language, Data and Knowledge (LDK 2021)*, ed. by D. Gromann et al., vol. 93, Open Access Series in Informatics (OASIcs), Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021 28:1, ISBN: 978-3-95977-199-3, URL: https://drops.dagstuhl.de/opus/volltexte/2021/14564 (cit. on pp. 7, 193, 194).

[25] B. Kirsch, S. Giesselbach, D. Knodt and S. Rüping, *Robust End-User-Driven Social Media Monitoring for Law Enforcement and Emergency Monitoring*, Community-Oriented Policing and Technological Innovations (2018) 29 (cit. on pp. 7, 194, 195, 271).

[26] A. Vaswani et al., “Attention is All you Need”, *Advances in Neural Information Processing Systems*, ed. by I. Guyon et al., vol. 30, Curran Associates, Inc., 2017, URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf (cit. on pp. 10, 18, 20).

[27] D. E. Rumelhart, G. E. Hinton and R. J. Williams, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, Cambridge, MA (cit. on p. 10).

[28] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, Neural computation **9** (1997) 1735 (cit. on pp. 10, 17).

[29] C. J. Matheus, P. K. Chan and G. Piatetsky-Shapiro, *Systems for Knowledge Discovery in Databases*, IEEE Transactions on Knowledge and Data Engineering **5** (1993) (cit. on pp. 11, 38, 49, 97, 201).

[30] U. Fayyad, G. Piatetsky-Shapiro and P. Smyth, “From Data Mining to Knowledge Discovery in Databases”, *AI Magazine*, vol. 17, 1996 37, URL: http://www.aaai.org/ojs/index.php/aimagazine/article/view/1230%5Cnhttp://www.rbc.lsie.unb.br/index.php/rbc/article/viewFile/594/577%5Cnhttp://portal.acm.org/citation.cfm?doid=1900008.1900067%5Cnhttp://www.i

nf.ufg.br/sites/default/files/uploads/relatorios-
(cit. on pp. 11, 12, 49, 96, 97, 134, 135, 201, 202).

[31] N. W. Grady, J. A. Payne and H. Parker, *Agile Big Data Analytics*,
2017 Ieee International Conference on Big Data (Big Data) (2017) 2331, ISSN: 2639-1589,
URL: http://www.midp-info.org/uploads/1/0/6/5/10650753/s02208_2693.pdf
(cit. on pp. 11, 51, 76, 79, 83, 105, 162, 229).

[32] I. H. Witten, E. Frank and M. A. Hall,
Data Mining: Practical Machine Learning Tools and Techniques, 3rd ed.,
Morgan Kaufmann Series in Data Management Systems,
Amsterdam: Morgan Kaufmann, 2011, ISBN: 978-0-12-374856-0,
URL: http://www.sciencedirect.com/science/book/9780123748560 (cit. on p. 11).

[33] *Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2025*,
https://www.statista.com/statistics/871513/worldwide-data-created/,
Accessed: 2022-07-10 (cit. on p. 12).

[34] M. Kaufmann, *Big data management canvas: A reference model for value creation from data*,
Big Data and Cognitive Computing **3** (2019) 1, ISSN: 25042289 (cit. on pp. 12, 50, 102, 218).

[35] D. Laney, *3D Data Management: Controlling Data Volume, Velocity, and Variety*, tech. rep.,
META Group, 2001,
URL: http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf
(cit. on p. 12).

[36] M. Schroeck, R. Shockley, J. Smart, D. Romero-Morales and P. Tufano,
Analytics: The Real-World Use of Big Data,
IBM Institute for Business Value - Executive Report, IBM Institute for Business Value, 2012
(cit. on p. 12).

[37] Y. Demchenko, P. Grosso, C. de Laat and P. Membrey,
"Addressing big data issues in Scientific Data Infrastructure",
2013 International Conference on Collaboration Technologies and Systems (CTS), 2013 48
(cit. on p. 12).

[38] J. Andreu-Perez, C. Poon, R. Merrifield, S. Wong and G.-Z. Yang, *Big Data for Health*,
IEEE journal of biomedical and health informatics **19** (2015) (cit. on p. 12).

[39] M. A.-d. Khan, M. F. Uddin and N. Gupta,
"Seven V's of Big Data understanding Big Data to extract value", *Proceedings of the 2014 Zone 1 Conference of the American Society for Engineering Education*, 2014 1 (cit. on p. 12).

[40] W. Chang and N. Grady,
NIST Big Data Interoperability Framework: Volume 1, Definitions, en, 2019
(cit. on pp. 12, 218).

[41] J. Dutcher, *What is Big Data?*,
https://ischoolonline.berkeley.edu/blog/what-is-big-data/,
Accessed: 2023-06-11, 2019 (cit. on p. 12).

[42] M. Zaharia et al., *Apache spark: a unified engine for big data processing*, Communications of the ACM **59** (2016) 56 (cit. on p. 12).

[43] N. Garg, *Apache kafka*, Packt Publishing Birmingham, UK, 2013 (cit. on p. 12).

[44] T. White, *Hadoop: The definitive guide*, " O'Reilly Media, Inc.", 2012 (cit. on p. 12).

[45] V. Dhar, *Data Science and Prediction*, Commun. ACM **56** (2013) 64, ISSN: 0001-0782, URL: https://doi.org/10.1145/2500499 (cit. on p. 12).

[46] F. Provost and T. Fawcett, *Data Science for Business: What you need to know about data mining and data-analytic thinking*, " O'Reilly Media, Inc.", 2013 (cit. on p. 12).

[47] J. J. T. Thomas, W. Roberts and P. Nathan, *Operationalizing AI*, First Edit, O'Reilly, 2021 1, ISBN: 9781098101299, URL: https://www.imwuc.org/HigherLogic/System/DownloadDocumentFile.ashx?DocumentFileKey=19e3a670-03a7-1abb-72c0-51196c4d11d1 (cit. on pp. 13, 52, 113, 130, 257–259).

[48] S. Studer et al., *Towards CRISP-ML (Q): a machine learning process model with quality assurance methodology*, Machine learning and knowledge extraction **3** (2021) 392 (cit. on pp. 13, 70–72, 77, 81, 114, 149, 160, 162).

[49] T. M. Mitchell et al., *Machine learning*, vol. 1, McGraw-hill New York, 2007 (cit. on p. 13).

[50] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*, Cambridge university press, 2014 (cit. on p. 13).

[51] S. Girgis, E. Amer and M. Gadallah, "Deep Learning Algorithms for Detecting Fake News in Online Text", 2018 (cit. on p. 16).

[52] G. Bebis and M. Georgiopoulos, *Feed-forward neural networks*, Ieee Potentials **13** (1994) 27 (cit. on p. 16).

[53] D. E. Rumelhart, G. E. Hinton and R. J. Williams, *Learning representations by back-propagating errors*, Nature **323** (1986) 533 (cit. on p. 16).

[54] D. Bahdanau, K. Cho and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, arXiv preprint arXiv:1409.0473 (2014) (cit. on p. 18).

[55] D. M. Blei, A. Y. Ng and M. I. Jordan, *Latent dirichlet allocation*, J. Mach. Learn. Res. **3** (2003) 993, ISSN: 1532-4435, URL: http://portal.acm.org/citation.cfm?id=944937 (cit. on pp. 23, 192).

[56] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado and J. Dean, *Distributed representations of words and phrases and their compositionality*, Advances in neural information processing systems **26** (2013) (cit. on pp. 24, 27).

[57] Z. S. Harris, *Distributional Structure*, ¡i¿WORD¡/i¿ **10** (1954) 146, eprint: https://doi.org/10.1080/00437956.1954.11659520, URL: https://doi.org/10.1080/00437956.1954.11659520 (cit. on p. 24).

[58] J. R. Firth, *A synopsis of linguistic theory 1930-55.*, **1952-59** (1957) 1 (cit. on p. 24).

[59] Y. Kim, "Convolutional Neural Networks for Sentence Classification", *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, 2014 1746, URL: https://aclanthology.org/D14-1181 (cit. on p. 24).

[60] J. P. Chiu and E. Nichols, *Named Entity Recognition with Bidirectional LSTM-CNNs*, Transactions of the Association for Computational Linguistics **4** (2016) 357, ISSN: 2307-387X, eprint: https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00104/1567392/tacl_a_00104.pdf, URL: https://doi.org/10.1162/tacl%5C_a%5C_00104 (cit. on p. 24).

[61] Meta, *Word vectors for 157 languages*, https://fasttext.cc/docs/en/crawl-vectors.html, Accessed: 2023-06-11 (cit. on p. 25).

[62] P. Bojanowski, E. Grave, A. Joulin and T. Mikolov, *Enriching word vectors with subword information*, Transactions of the association for computational linguistics **5** (2017) 135 (cit. on p. 25).

[63] Q. Le and T. Mikolov, "Distributed representations of sentences and documents", *International conference on machine learning*, PMLR, 2014 1188 (cit. on p. 25).

[64] K. Orkphol and W. Yang, *Word sense disambiguation using cosine similarity collaborates with Word2vec and WordNet*, Future Internet **11** (2019) 114 (cit. on p. 25).

[65] M. E. Peters et al., "Deep Contextualized Word Representations", *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana: Association for Computational Linguistics, 2018 2227, URL: https://aclanthology.org/N18-1202 (cit. on pp. 25, 27).

[66] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, arXiv preprint arXiv:1810.04805 (2018) (cit. on pp. 25, 27, 70, 72, 79, 80).

[67] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever et al., *Improving language understanding by generative pre-training*, (2018) (cit. on p. 26).

[68] M. Hearst, S. Dumais, E. Osuna, J. Platt and B. Scholkopf, *Support vector machines*, IEEE Intelligent Systems and their Applications **13** (1998) 18 (cit. on p. 26).

[69] T. B. Brown et al., *Language Models are Few-Shot Learners*, 2020, URL: https://arxiv.org/abs/2005.14165 (cit. on pp. 27, 43, 70–72, 75, 80, 131).

[70] Z. Jun et al., "An Exploration of Prompt-Based Zero-Shot Relation Extraction Method", English, *Proceedings of the 21st Chinese National Conference on Computational Linguistics*, Nanchang, China: Chinese Information Processing Society of China, 2022 786, URL: https://aclanthology.org/2022.ccl-1.70 (cit. on p. 27).

[71] J. Hoffmann et al., *Training Compute-Optimal Large Language Models*, 2022, arXiv: 2203.15556 [cs.CL] (cit. on pp. 27, 149).

[72] L. Ouyang et al., *Training language models to follow instructions with human feedback*, Advances in Neural Information Processing Systems **35** (2022) 27730 (cit. on pp. 27, 29, 183, 190).

[73] Y. Gu et al., *Domain-specific language model pretraining for biomedical natural language processing*, ACM Transactions on Computing for Healthcare (HEALTH) **3** (2021) 1 (cit. on pp. 28, 80, 131).

[74] Z. Yang et al., "XLNet: Generalized Autoregressive Pretraining for Language Understanding", *Advances in Neural Information Processing Systems*, ed. by H. Wallach et al., vol. 32, Curran Associates, Inc., 2019, URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/dc6a7e655d7e5840e66733e9ee67cc69-Paper.pdf (cit. on p. 28).

[75] Z. Yang, S. Wang, B. P. S. Rawat, A. Mitra and H. Yu, "Knowledge Injected Prompt Based Fine-tuning for Multi-label Few-shot ICD Coding", *Findings of the Association for Computational Linguistics: EMNLP 2022*, Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, 2022 1767, URL: https://aclanthology.org/2022.findings-emnlp.127 (cit. on p. 28).

[76] E. Barba, L. Procopio and R. Navigli, "ConSeC: Word Sense Disambiguation as Continuous Sense Comprehension", *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021 1492, URL: https://aclanthology.org/2021.emnlp-main.112 (cit. on p. 28).

[77] Q. Peng et al., *ERNIE-Layout: Layout Knowledge Enhanced Pre-training for Visually-rich Document Understanding*, arXiv preprint arXiv:2210.06155 (2022) (cit. on p. 28).

[78] S. Wadhwa, S. Amir and B. C. Wallace, *Revisiting relation extraction in the era of large language models*, arXiv preprint arXiv:2305.05003 (2023) (cit. on p. 28).

[79] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks", *Conference on Empirical Methods in Natural Language Processing*, 2019 (cit. on p. 29).

[80] S. Borgeaud et al., "Improving language models by retrieving from trillions of tokens", *International conference on machine learning*, PMLR, 2022 2206 (cit. on p. 29).

[81] S. Takase and S. Kiyono, *Lessons on Parameter Sharing across Layers in Transformers*, ArXiv **abs/2104.06022** (2021) (cit. on p. 29).

[82] R. Anil et al., *PaLM 2 Technical Report*, 2023, arXiv: 2305.10403 [cs.CL] (cit. on p. 29).

[83] A. Kedia, S. C. Chinthakindi and W. Ryu, "Beyond Reptile: Meta-Learned Dot-Product Maximization between Gradients for Improved Single-Task Regularization", *Findings of the Association for Computational Linguistics: EMNLP 2021*, Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021 407, URL: https://aclanthology.org/2021.findings-emnlp.37 (cit. on p. 29).

[84] T. Wolf et al., *Huggingface's transformers: State-of-the-art natural language processing*, arXiv preprint arXiv:1910.03771 (2019)
(cit. on pp. 29, 47, 72, 73, 77, 130, 143, 149, 159, 163).

[85] PMI, ed., *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*, 5th ed., Newtown Square, PA: Project Management Institute, 2013, ISBN: 978-1-935589-67-9
(cit. on pp. 30, 31, 38, 39).

[86] O. Marbán, J. Segovia, E. Menasalvas and C. Fernández-Baizán,
Toward Data Mining Engineering: A Software Engineering Approach, Inf. Syst. **34** (2009) 87, ISSN: 0306-4379, URL: https://doi.org/10.1016/j.is.2008.04.003
(cit. on pp. 31, 32, 50, 98, 209).

[87] J. Saltz and N. Hotz, "Factors that Influence the Selection of a Data Science Process Management Methodology: An Exploratory Study", 2021 (cit. on pp. 32, 56, 133).

[88] I. Martinez, E. Viles and I. G. Olaizola,
Data Science Methodologies: Current Challenges and Future Approaches,
Big Data Research **24** (2021), ISSN: 22145796, arXiv: 2106.07287 (cit. on pp. 32, 39, 47, 51, 56, 70, 73, 75, 77, 79, 83, 95, 99, 101, 102, 104–107, 109, 110, 113, 124, 132, 145).

[89] BAWiki, *THE MYTH OF THE 'WATERFALL' SDLC*,
http://www.bawiki.com/wiki/Waterfall.html, Accessed: 2023-06-12 (cit. on p. 32).

[90] H. D. Benington, *Production of Large Computer Programs*,
Annals of the History of Computing **5** (1983) 350 (cit. on p. 32).

[91] W. W. Royce,
"Managing the Development of Large Software Systems: Concepts and Techniques",
Proceedings of the 9th International Conference on Software Engineering, ICSE '87,
Monterey, California, USA: IEEE Computer Society Press, 1987 328, ISBN: 0897912160
(cit. on pp. 32, 33).

[92] B. Boehm, *A Spiral Model of Software Development and Enhancement*,
SIGSOFT Softw. Eng. Notes **11** (1986) 14, ISSN: 0163-5948,
URL: https://doi.org/10.1145/12944.12948 (cit. on pp. 33, 34).

[93] K. Beck et al., *Manifesto for Agile Software Development*, http://agilemanifesto.org,
Accessed: 2023-06-18, 2001 (cit. on p. 33).

[94] J. Saltz, K. Crowston et al.,
Comparing data science project management methodologies via a controlled experiment,
(2017) (cit. on pp. 34, 109, 241).

[95] K. Schwaber, *SCRUM Development Process*,
Business Object Design and Implementation (1987) 117 (cit. on p. 34).

[96] N. Ovesen, K. Eriksen and C. Tollestrup,
Agile attitude: Review of agile methods for use in design education,
DS 69: Proceedings of E and PDE 2011, the 13th International Conference on Engineering and Product Design Education (2011) (cit. on p. 35).

[97] W. Zayat and O. Senvar,
Framework Study for Agile Software Development Via Scrum and Kanban,
International Journal of Innovation and Technology Management **17** (2020) (cit. on p. 36).

[98] K. Beck, *Extreme Programming Explained: Embrace Change*, An Alan R. Apt Book Series,
Addison-Wesley, 2000, ISBN: 9780201616415,
URL: https://books.google.de/books?id=G8EL4H4vf7UC (cit. on p. 36).

[99] D. Sculley et al., *Hidden technical debt in machine learning systems*,
Advances in Neural Information Processing Systems **2015-January** (2015) 2503,
ISSN: 10495258 (cit. on pp. 37, 72, 144).

[100] *CRISP-DM is Still the Most Popular Framework for Executing Data Science Projects*,
https://www.datascience-pm.com/crisp-dm-still-most-popular/,
Accessed: 2022-06-15 (cit. on pp. 38, 39, 97, 203).

[101] V. Plotnikova, M. Dumas and F. Milani,
Adaptations of data mining methodologies: A systematic literature review,
PeerJ Computer Science **6** (2020) 1, ISSN: 23765992 (cit. on pp. 38, 50).

[102] P. Chapman et al., *CRISP-DM 1.0 Step-by-step Data Mining Guide*, SPSS inc **78** (2000) 1,
ISSN: 1098-6596, arXiv: arXiv:1011.1669v3,
URL: https://www.semanticscholar.org/paper/CRISP-DM-1.0%5C%3A-Step-by-step-data-mining-guide-Chapman-Clinton/54bad20bbc7938991bf34f86dde0babfbd2d5a72%5C%0Ahttp://www.crisp-dm.org/CRISPWP-0800.pdf
(cit. on pp. 40, 50, 97, 158, 203).

[103] S. García, J. Luengo and F. Herrera, *Data Preprocessing in Data Mining*, ()
(cit. on pp. 41, 205).

[104] S. Angée, S. I. Lozano-Argel, E. N. Montoya-Munera, J. D. Ospina-Arango and
M. S. Tabares-Betancur, *Towards an improved ASUM-DM process methodology for cross-disciplinary multi-organization big data & analytics projects*,
Communications in Computer and Information Science **877** (2018) 613, ISSN: 18650929
(cit. on pp. 46, 56, 103, 121, 223, 224).

[105] M. Soukaina, H. Anoun, M. Ridouani and L. Hassouni,
"A study of the factors and methodologies to drive successfully a big data project",
2019 Third International Conference on Intelligent Computing in Data Sciences (ICDS), 2019
1 (cit. on pp. 46, 61, 73–79, 83, 145).

[106] C. Schröer, F. Kruse and J. M. Gómez,
A systematic literature review on applying CRISP-DM process model,
Procedia Computer Science **181** (2021) 526, ISSN: 18770509,
URL: https://doi.org/10.1016/j.procs.2021.01.199 (cit. on pp. 46, 77, 79, 82).

[107] R. Ishaq, *How much did GPT-3 cost?*, https://www.pcguide.com/apps/gpt-3-cost/,
Accessed: 2023-08-07 (cit. on pp. 47, 145).

[108] S. Gururangan et al.,
"Don't Stop Pretraining: Adapt Language Models to Domains and Tasks",
Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics,
Online: Association for Computational Linguistics, 2020 8342,
URL: https://aclanthology.org/2020.acl-main.740 (cit. on pp. 47, 143, 149).

[109] M. Zhang, O. Press, W. Merrill, A. Liu and N. A. Smith,
How language model hallucinations can snowball, arXiv preprint arXiv:2305.13534 (2023)
(cit. on p. 47).

[110] J. S. Saltz and I. Krasteva,
Current approaches for executing big data science projects—a systematic literature review,
PeerJ Computer Science **8** (2022) e862, ISSN: 23765992
(cit. on pp. 47, 50, 73, 74, 76–79, 81–83).

[111] S. Moyle and A. Jorge,
RAMSYS - A methodology for supporting rapid remote collaborative data mining projects,
Proceedings of the ECML/PKDD'01 Workshop on Integrating Aspects of Data Mining,
Decision Support and Meta-Learning (2001) 20,
URL: http://www.niaad.liacc.up.pt/$%5Csim$amjorge
(cit. on pp. 50, 96, 98, 125, 129, 131, 132, 135, 136, 152, 206).

[112] F. Martínez-Plumed et al., *CASP-DM: Context Aware Standard Process for Data Mining*,
ArXiv **abs/1709.09003** (2017) (cit. on pp. 50, 80, 96, 136).

[113] D. Dutta and I. Bose, *Managing a big data project: The case of Ramco cements limited*,
International Journal of Production Economics **165** (2015) 293, ISSN: 09255273,
URL: http://dx.doi.org/10.1016/j.ijpe.2014.12.032
(cit. on pp. 50, 100, 134–136, 200, 212, 213).

[114] M. Vanauer, C. Bohle and B. Hellingrath,
Guiding the introduction of big data in organizations: A methodology with business- and data-driven ideation and enterprise architecture management-based implementation,
Proceedings of the Annual Hawaii International Conference on System Sciences **2015-March**
(2015) 908, ISSN: 15301605 (cit. on pp. 50, 101, 134, 214, 215).

[115] J. B. Rollins, *Foundational Methodology for Data Science*, IBM analytics (2015) 1,
URL: https://www.ibm.com/downloads/cas/WKK9DX51
(cit. on pp. 50, 102, 134, 136, 149, 223, 225).

[116] *IBM: Analytics Solutions Unified Method*,
http://gforge.icesi.edu.co/ASUM-DM_External/index.htm#cognos.external.asum-DM_Teaser/deliveryprocesses/ASUM-DM_8A5C87D5.html,
Accessed: 2022-09-18 (cit. on pp. 50, 103, 133–136, 153, 220, 222).

[117] D. Larson and V. Chang,
A review and future direction of agile, business intelligence, analytics and data science,
International Journal of Information Management **36** (2016) 700, ISSN: 02684012,
URL: http://dx.doi.org/10.1016/j.ijinfomgt.2016.04.013
(cit. on pp. 51, 105, 135, 136, 233).

[118] J. S. Saltz, R. Heckman, K. Crowston and Y. Hegde, *Midst: An enhanced development environment that improves the maintainability of a data science analysis*, International Journal of Information Systems and Project Management **8** (2020) 5, ISSN: 21827788 (cit. on pp. 51, 106, 235).

[119] A. Kühn et al., "Analytics Canvas – A Framework for the Design and Specification of Data Analytics Projects", *28th CIRP Design Conference 2018*, Nantes, 2018 (cit. on pp. 51, 107, 134, 135, 236, 238, 239, 292).

[120] *Enterprise Miner SEMMA*, https://s2.smu.edu/tfomby/eco5385_eco6380/data/SPSS/SAS%20_%20SEMMA.pdf, Accessed: 2022-06-19 (cit. on pp. 51, 241).

[121] *Data Science Workflow: Overview and Challenges*, https://cacm.acm.org/blogs/blog-cacm/169199-data-science-workflow-overview-and-challenges/fulltext, Accessed: 2022-09-17 (cit. on pp. 51, 108, 237, 240).

[122] A. D. Dietrich, M. A. Us, D. Dietrich and M. A. Us, *(12) United States Patent*, **1** (2016) (cit. on pp. 51, 109, 246).

[123] D. D. Labs, *The Practical Guide to Managing Data Science at Scale*, https://www.dominodatalab.com/resources/managing-data-science/, Accessed: 2022-09-19, 2017 (cit. on pp. 52, 112, 132, 134, 135, 141, 149, 162, 254, 255).

[124] C. Byrne, *Development Work ows for Data Scientists*, O'Reilly Media, Inc., 2017, ISBN: 9781491983324 (cit. on pp. 52, 112, 129, 135, 136, 249, 252, 253).

[125] D. Kreuzberger, N. Kühl and S. Hirschl, *Machine Learning Operations (MLOps): Overview, Definition, and Architecture*, (2022), arXiv: 2205.02302, URL: http://arxiv.org/abs/2205.02302 (cit. on pp. 52, 114, 134, 135, 166, 261, 263).

[126] C. G. Skarpathiotaki and K. E. Psannis, *Cross-Industry Process Standardization for Text Analytics*, Big Data Research **27** (2022) 100274, ISSN: 2214-5796, URL: https://www.sciencedirect.com/science/article/pii/S2214579621000915 (cit. on pp. 52, 115, 136, 264).

[127] B. Kitchenham et al., *Systematic literature reviews in software engineering – A systematic literature review*, Information and Software Technology **51** (2009) 7, Special Section - Most Cited Articles in 2002 and Regular Research Papers, ISSN: 0950-5849, URL: https://www.sciencedirect.com/science/article/pii/S0950584908001390 (cit. on p. 53).

[128] G. Kalus and M. Kuhrmann, *Criteria for software process tailoring: A systematic review*, ACM International Conference Proceeding Series (2013) 171 (cit. on pp. 53, 55).

[129] P. Runeson and M. Höst, *Guidelines for conducting and reporting case study research in software engineering*, Empirical Software Engineering **14** (2 2009) 131, ISSN: 1382-3256, URL: http://portal.acm.org/citation.cfm?id=1519313.1519324 (cit. on p. 53).

[130] K. Eckert and P. V. Britos, "Data science methodologies selection with hierarchical analytical process and personal construction theory",
XXV Congreso Argentino de Ciencias de la Computación (CACIC)(Universidad Nacional de Río Cuarto, Córdoba, 14 al 18 de octubre de 2019), 2019 (cit. on p. 56).

[131] V. Sharma, A. Stranieri, J. Ugon, P. Vamplew and L. Martin,
"An Agile Group Aware Process beyond CRISP-DM: A Hospital Data Mining Case Study",
Proceedings of the International Conference on Compute and Data Analysis, ICCDA '17,
Lakeland, FL, USA: Association for Computing Machinery, 2017 109, ISBN: 9781450352413,
URL: https://doi.org/10.1145/3093241.3093273 (cit. on pp. 56, 79).

[132] H. Wiemer, L. Drowatzky and S. Ihlenfeldt, *Data Mining Methodology for Engineering Applications (DMME)—A Holistic Extension to the CRISP-DM Model*,
Applied Sciences **9** (2019), ISSN: 2076-3417,
URL: https://www.mdpi.com/2076-3417/9/12/2407 (cit. on p. 56).

[133] E. Kristoffersen, O. Aremu, F. Blomsma, P. Mikalef and J. Li, *Exploring the Relationship Between Data Science and Circular Economy: an Enhanced CRISP-DM Process Model*, 2019 (cit. on p. 56).

[134] A. Vogelsang and M. Borg,
"Requirements engineering for machine learning: Perspectives from data scientists",
2019 IEEE 27th International Requirements Engineering Conference Workshops (REW),
IEEE, 2019 245 (cit. on pp. 56, 70, 73, 75, 78, 82).

[135] J. S. Saltz and I. Shamshurin, *Big data team process methodologies: A literature review and the identification of key factors for a project's success*,
Proceedings - 2016 IEEE International Conference on Big Data, Big Data 2016 (2016) 2872 (cit. on p. 56).

[136] H. H. Altarturi, K.-Y. Ng, M. I. H. Ninggal, A. S. A. Nazri and A. A. A. Ghani,
"A requirement engineering model for big data software",
2017 IEEE Conference on Big Data and Analytics (ICBDA), 2017 111 (cit. on pp. 56, 81).

[137] H. Belani, M. Vukovic and Z. Car,
"Requirements engineering challenges in building ai-based complex systems", *Proceedings - 2019 IEEE 27th International Requirements Engineering Conference Workshops, REW 2019*,
Institute of Electrical and Electronics Engineers Inc., 2019 252, ISBN: 9781728151656
(cit. on pp. 56, 71–75, 77, 79–83, 144, 189).

[138] G. J. Miller,
Artificial Intelligence Project Success Factors: Moral Decision-Making with Algorithms,
Proceedings of the 16th Conference on Computer Science and Intelligence Systems, FedCSIS 2021 **25** (2021) 379 (cit. on p. 57).

[139] G. J. Miller, "Artificial Intelligence Project Success Factors—Beyond the Ethical Principles",
Information Technology for Management: Business and Social Issues,
ed. by E. Ziemba and W. Chmielarz, Cham: Springer International Publishing, 2022 65,
ISBN: 978-3-030-98997-2 (cit. on pp. 57, 65, 70, 72, 73, 75, 79, 80, 82, 83, 140).

[140] H. Villamizar, M. Kalinowski and H. lopes,
A Catalogue of Concerns for Specifying Machine Learning-Enabled Systems, 2022,
URL: https://arxiv.org/abs/2204.07662 (cit. on pp. 61, 70–72, 76, 77, 80, 82, 83).

[141] I. Martinez, E. Viles and I. G. Olaizola,
A survey study of success factors in data science projects,
Proceedings - 2021 IEEE International Conference on Big Data, Big Data 2021 (2021) 2313,
arXiv: 2201.06310 (cit. on pp. 63, 70, 73–81, 83, 132, 145).

[142] L. E. Lwakatare, A. Raj, J. Bosch, H. H. Olsson and I. Crnkovic, "A Taxonomy of Software Engineering Challenges for Machine Learning Systems: An Empirical Investigation",
Agile Processes in Software Engineering and Extreme Programming,
ed. by P. Kruchten, S. Fraser and F. Coallier, Cham: Springer International Publishing, 2019 227, ISBN: 978-3-030-19034-7 (cit. on pp. 64, 76, 79–82).

[143] A. Paleyes, R.-G. Urma and N. D. Lawrence,
Challenges in Deploying Machine Learning: A Survey of Case Studies,
ACM Comput. Surv. **55** (2022), ISSN: 0360-0300,
URL: https://doi.org/10.1145/3533378 (cit. on pp. 66, 72–75, 79–83).

[144] Z. A. Al-Sai, R. Abdullah and M. H. Husin,
Critical Success Factors for Big Data: A Systematic Literature Review,
IEEE Access **8** (2020) 118940 (cit. on pp. 67, 70, 73–77, 79, 81, 83, 145).

[145] S. Martínez-Fernández et al., *Software engineering for AI-based systems: a survey*,
ACM Transactions on Software Engineering and Methodology (TOSEM) **31** (2022) 1
(cit. on pp. 67, 70–72, 75–83, 143, 144, 189).

[146] I. Krasteva and S. Ilieva, "Adopting Agile Software Development Methodologies in Big Data Projects – a Systematic Literature Review of Experience Reports",
2020 IEEE International Conference on Big Data (Big Data), 2020 2028
(cit. on pp. 70, 72, 75, 82).

[147] T. K. Lwin and A. V. Bogdanov, "Definition and Scope of Big Data Problem",
Proceedings of the 2018 2nd International Conference on Cloud and Big Data Computing,
ICCBDC'18, Barcelona, Spain: Association for Computing Machinery, 2018 38,
ISBN: 9781450364744, URL: https://doi.org/10.1145/3264560.3264571
(cit. on pp. 70, 73, 75).

[148] S. R. Kourla, E. Putti and M. Maleki,
REBD: A Conceptual Framework for Big Data Requirements Engineering,
arXiv preprint arXiv:2006.11195 (2020) (cit. on p. 70).

[149] M. M. Ben Aissa, L. Sfaxi and R. Robbana,
"DECIDE: A New Decisional Big Data Methodology for a Better Data Governance",
Information Systems, ed. by M. Themistocleous, M. Papadaki and M. M. Kamal,
Cham: Springer International Publishing, 2020 63, ISBN: 978-3-030-63396-7
(cit. on pp. 70, 74, 79, 81, 82).

[150] D. Arruda and N. H. Madhavji,
"State of requirements engineering research in the context of big data applications",
Requirements Engineering: Foundation for Software Quality: 24th International Working Conference, REFSQ 2018, Utrecht, The Netherlands, March 19-22, 2018, Proceedings 24,
Springer, 2018 307 (cit. on pp. 70, 72, 75, 78, 81, 82).

[151] G. J. Miller, "Comparative Analysis of Big Data Analytics and BI Projects",
2018 Federated Conference on Computer Science and Information Systems (FedCSIS), 2018
701 (cit. on pp. 70, 72–76, 82, 83).

[152] H. Villamizar, T. Escovedo and M. Kalinowski,
"Requirements engineering for machine learning: A systematic mapping study", *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*,
IEEE, 2021 29 (cit. on pp. 70, 71, 75, 82).

[153] N. Kraut and F. Transchel, "On the Application of SCRUM in Data Science Projects",
2022 7th International Conference on Big Data Analytics (ICBDA), 2022 1
(cit. on pp. 70, 79, 143).

[154] R. Hobbs, "Integrating Ethically Align Design into Agile and CRISP-DM",
SoutheastCon 2021, 2021 1 (cit. on pp. 70, 72, 75, 77, 82).

[155] S. Amershi et al., "Software engineering for machine learning: A case study",
2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), IEEE, 2019 291
(cit. on pp. 70, 72, 74–76, 79–83, 144, 189).

[156] A. Radford et al., "Language Models are Unsupervised Multitask Learners", 2019
(cit. on p. 70).

[157] C. Lu et al., *An Overview and Case Study of the Clinical AI Model Development Life Cycle for Healthcare Systems*, 2020, URL: https://arxiv.org/abs/2003.07678
(cit. on pp. 70, 73, 74, 79, 132, 189).

[158] M. M. John, H. H. Olsson and J. Bosch, "Developing ML/DL Models: A Design Framework",
Proceedings of the International Conference on Software and System Processes, ICSSP '20,
Seoul, Republic of Korea: Association for Computing Machinery, 2020 1,
ISBN: 9781450375122, URL: https://doi.org/10.1145/3379177.3388892
(cit. on pp. 70–72, 74, 75, 78–80, 82, 83, 189, 190).

[159] R. Thoppilan et al., *LaMDA: Language Models for Dialog Applications*, 2022,
URL: https://arxiv.org/abs/2201.08239 (cit. on pp. 70, 75, 80).

[160] Y. Liu et al., *RoBERTa: A Robustly Optimized BERT Pretraining Approach*, 2019,
URL: https://arxiv.org/abs/1907.11692 (cit. on p. 70).

[161] H. Touvron et al., *LLaMA: Open and Efficient Foundation Language Models*, 2023,
URL: https://arxiv.org/abs/2302.13971 (cit. on pp. 70, 131).

[162] I. Beltagy, M. E. Peters and A. Cohan, *Longformer: The long-document transformer*,
arXiv preprint arXiv:2004.05150 (2020) (cit. on p. 70).

[163] S. Wang, B. Z. Li, M. Khabsa, H. Fang and H. Ma,
Linformer: Self-attention with linear complexity, arXiv preprint arXiv:2006.04768 (2020) (cit. on p. 70).

[164] M. Zaheer et al., *Big bird: Transformers for longer sequences*,
Advances in neural information processing systems **33** (2020) 17283 (cit. on p. 70).

[165] M. E. Peters et al., *Knowledge Enhanced Contextual Word Representations*, 2019,
URL: https://arxiv.org/abs/1909.04164 (cit. on pp. 70, 80).

[166] S. Appalaraju, B. Jasani, B. U. Kota, Y. Xie and R. Manmatha,
"Docformer: End-to-end transformer for document understanding",
Proceedings of the IEEE/CVF international conference on computer vision, 2021 993
(cit. on pp. 71, 80).

[167] Y. Xu et al., "Layoutlm: Pre-training of text and layout for document image understanding",
Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020 1192 (cit. on p. 71).

[168] A. Conneau et al., *Unsupervised cross-lingual representation learning at scale*,
arXiv preprint arXiv:1911.02116 (2019) (cit. on p. 71).

[169] K. M. Habibullah and J. Horkoff, "Non-functional requirements for machine learning:
understanding current use and challenges in industry",
2021 IEEE 29th International Requirements Engineering Conference (RE), IEEE, 2021 13
(cit. on pp. 71, 72, 75, 78, 80–82).

[170] J. Horkoff,
"Non-Functional Requirements for Machine Learning: Challenges and New Directions",
2019 IEEE 27th International Requirements Engineering Conference (RE), 2019 386
(cit. on pp. 71, 72, 75, 81, 82).

[171] S. Idowu, D. Strüber and T. Berger, "Asset Management in Machine Learning: A Survey",
Proceedings of the 43rd International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP '21, Virtual Event, Spain: IEEE Press, 2021 51,
ISBN: 9780738146690,
URL: https://doi.org/10.1109/ICSE-SEIP52600.2021.00014 (cit. on pp. 71–73).

[172] *Incredibly fast bloom inference with DeepSpeed and Accelerate*,
URL: https://huggingface.co/blog/bloom-inference-pytorch-scripts
(cit. on pp. 72, 75).

[173] V. Sanh, L. Debut, J. Chaumond and T. Wolf,
DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter,
arXiv preprint arXiv:1910.01108 (2019) (cit. on p. 72).

[174] J. Vig, "BertViz: A tool for visualizing multihead self-attention in the BERT model",
ICLR workshop: Debugging machine learning models, 2019 (cit. on pp. 72, 77).

[175] Z. Qi, S. Khorram and F. Li,
"Visualizing Deep Networks by Optimizing with Integrated Gradients.", *CVPR Workshops*,
vol. 2, 2019 1 (cit. on p. 72).

[176] J. Wei et al., *Chain of thought prompting elicits reasoning in large language models*, arXiv preprint arXiv:2201.11903 (2022) (cit. on p. 72).

[177] G. Mialon et al., *Augmented Language Models: a Survey*, arXiv preprint arXiv:2302.07842 (2023) (cit. on p. 72).

[178] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger and Y. Artzi, *Bertscore: Evaluating text generation with bert*, arXiv preprint arXiv:1904.09675 (2019) (cit. on pp. 72, 162).

[179] H.-M. Heyn et al., "Requirement engineering challenges for ai-intense systems development", *2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN)*, IEEE, 2021 89 (cit. on pp. 72, 75, 80, 82).

[180] B. Lester, R. Al-Rfou and N. Constant, *The power of scale for parameter-efficient prompt tuning*, arXiv preprint arXiv:2104.08691 (2021) (cit. on p. 72).

[181] M. Schulz et al., *Introducing DASC-PM: a data science process model*, (2020) (cit. on pp. 72, 73, 77, 79, 81, 145).

[182] Q. Chen et al., *Large language models in biomedical natural language processing: benchmarks, baselines, and recommendations*, arXiv preprint arXiv:2305.16326 (2023) (cit. on p. 72).

[183] M. Hesenius, N. Schwenzfeier, O. Meyer, W. Koop and V. Gruhn, "Towards a software engineering process for developing data-driven applications", *2019 IEEE/ACM 7th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE)*, IEEE, 2019 35 (cit. on pp. 72, 82, 144).

[184] I. Noorwali, D. Arruda and N. H. Madhavji, "Understanding quality requirements in the context of big data systems", *Proceedings of the 2nd International Workshop on BIG Data Software Engineering*, 2016 76 (cit. on pp. 72, 75, 81, 82).

[185] H. Liu, S. Eksmo, J. Risberg and R. Hebig, "Emerging and Changing Tasks in the Development Process for Machine Learning Systems", *Proceedings of the International Conference on Software and System Processes*, ICSSP '20, Seoul, Republic of Korea: Association for Computing Machinery, 2020 125, ISBN: 9781450375122, URL: https://doi.org/10.1145/3379177.3388905 (cit. on pp. 72, 73, 77–83, 143, 144).

[186] I. A. Ajah and H. F. Nweke, *Big Data and Business Analytics: Trends, Platforms, Success Factors and Applications*, Big Data and Cognitive Computing **3** (2019), ISSN: 2504-2289, URL: https://www.mdpi.com/2504-2289/3/2/32 (cit. on pp. 72, 75, 77, 82, 83).

[187] F. Ishikawa and N. Yoshioka, "How Do Engineers Perceive Difficulties in Engineering of Machine-Learning Systems? - Questionnaire Survey", *2019 IEEE/ACM Joint 7th International Workshop on Conducting Empirical Studies in Industry (CESI) and 6th International Workshop on Software Engineering Research and Industrial Practice (SER&IP)*, 2019 2 (cit. on pp. 72, 73, 77, 79–83).

[188] R. Philipp, A. Mladenow, C. Strauss and A. Völz,
"Machine Learning as a Service: Challenges in Research and Applications",
Proceedings of the 22nd International Conference on Information Integration and Web-Based Applications & Services, iiWAS '20,
Chiang Mai, Thailand: Association for Computing Machinery, 2021 396,
ISBN: 9781450389228, URL: https://doi.org/10.1145/3428757.3429152
(cit. on pp. 72–75, 78).

[189] I. Stoica et al., *A Berkeley View of Systems Challenges for AI*,
tech. rep. UCB/EECS-2017-159, EECS Department, University of California, Berkeley, 2017,
URL:
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-159.html
(cit. on pp. 72, 80, 81).

[190] M. Mitchell et al., "Model cards for model reporting",
Proceedings of the conference on fairness, accountability, and transparency, 2019 220
(cit. on pp. 73, 130, 166, 296).

[191] C. J. Costa and J. T. Aparicio, "POST-DS: A Methodology to Boost Data Science",
2020 15th Iberian Conference on Information Systems and Technologies (CISTI), 2020 1
(cit. on pp. 73, 145).

[192] G. J. Miller, "The influence of big data competencies, team structures, and data scientists on project success",
2019 IEEE Technology & Engineering Management Conference (TEMSCON), IEEE, 2019 1
(cit. on pp. 73, 145).

[193] A. Gupte, *Determining critical success factors for big data projects*,
PhD thesis: Purdue University, 2018 (cit. on pp. 73–79, 82, 83, 145).

[194] J. Baijens, R. Helms and D. Iren, "Applying Scrum in Data Science Projects",
2020 IEEE 22nd Conference on Business Informatics (CBI), vol. 1, 2020 30
(cit. on pp. 73, 83).

[195] J. S. Saltz and I. Shamshurin,
"Achieving Agile Big Data Science: The Evolution of a Team's Agile Process Methodology",
2019 IEEE International Conference on Big Data (Big Data), 2019 3477
(cit. on pp. 73, 78, 79, 83, 143).

[196] J. Saltz and A. Suthrland, "SKI: An Agile Framework for Data Science",
2019 IEEE International Conference on Big Data (Big Data), 2019 3468 (cit. on pp. 73, 83).

[197] A. Schmitz, M. Akila, D. Hecker, M. Poretschkin and S. Wrobel,
The why and how of trustworthy AI: An approach for systematic quality assurance when working with ML components, at-Automatisierungstechnik **70** (2022) 793 (cit. on p. 75).

[198] A. Cremers et al., *Trustworthy use of artificial intelligence-priorities from a philosophical, ethical, legal, and technological viewpoint as a basis for certification of artificial intelligence*,
Fraunhofer Institute for Intelligent Analysis and Information Systems (IAIS) (2019)
(cit. on p. 75).

[199] M. Poretschkin et al.,
Guideline for Trustworthy Artificial Intelligence–AI Assessment Catalog,
arXiv preprint arXiv:2307.03681 (2023) (cit. on p. 75).

[200] H. Washizaki et al.,
"Software engineering patterns for machine learning applications (sep4mla) part 2",
Proceedings of the 27th conference on pattern languages of programs, 2020 1
(cit. on pp. 76, 80–82, 144).

[201] H. Washizaki, H. Uchida, F. Khomh and Y.-G. Guéhéneuc,
"Studying software engineering patterns for designing machine learning systems",
2019 10th International Workshop on Empirical Software Engineering in Practice (IWESEP),
IEEE, 2019 49 (cit. on pp. 76, 80–82, 144).

[202] J. Alammar,
"Ecco: An open source library for the explainability of transformer language models",
Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations, 2021 249 (cit. on pp. 77, 145).

[203] Cdpierse, *Cdpierse/transformers-interpret: Model explainability that works seamlessly with transformers. explain your transformers model in just 2 lines of code.* Accessed: 2023-03-12,
URL: `https://github.com/cdpierse/transformers-interpret` (cit. on p. 77).

[204] N. Kokhlikyan et al.,
Captum: A unified and generic model interpretability library for PyTorch, 2020,
URL: `https://arxiv.org/abs/2009.07896` (cit. on p. 77).

[205] Z. Wan, X. Xia, D. Lo and G. C. Murphy,
How does Machine Learning Change Software Development Practices?,
IEEE Transactions on Software Engineering **47** (2021) 1857
(cit. on pp. 77–79, 81–83, 143, 144, 189, 192).

[206] *Papers with code - the latest in machine learning*, Accessed: 2023-03-13,
URL: `https://paperswithcode.com/` (cit. on pp. 77, 160).

[207] S. Passi and S. Barocas, "Problem formulation and fairness",
Proceedings of the conference on fairness, accountability, and transparency, 2019 39
(cit. on pp. 78, 143).

[208] E. de Souza Nascimento et al.,
"Understanding development process of machine learning systems: Challenges and solutions",
2019 acm/ieee international symposium on empirical software engineering and measurement (esem), IEEE, 2019 1 (cit. on pp. 78, 81).

[209] K. Singla, J. Bose and C. Naik,
"Analysis of Software Engineering for Agile Machine Learning Projects",
2018 15th IEEE India Council International Conference (INDICON), 2018 1
(cit. on pp. 79, 83).

[210] R. Ranawana and A. S. Karunananda, “An Agile Software Development Life Cycle Model for Machine Learning Application Development”,
2021 5th SLAAI International Conference on Artificial Intelligence (SLAAI-ICAI), 2021 1 (cit. on pp. 79–81, 83, 144).

[211] J. Pustejovsky and A. Stubbs, *Natural Language Annotation for Machine Learning: A guide to corpus-building for applications*, ” O’Reilly Media, Inc.”, 2012 (cit. on p. 79).

[212] M. Mintz, S. Bills, R. Snow and D. Jurafsky,
“Distant supervision for relation extraction without labeled data”,
Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2009 1003 (cit. on pp. 79, 191).

[213] P. Törnberg, *Chatgpt-4 outperforms experts and crowd workers in annotating political twitter messages with zero-shot learning*, arXiv preprint arXiv:2304.06588 (2023) (cit. on p. 79).

[214] T. Kudo and J. Richardson, *Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing*, arXiv preprint arXiv:1808.06226 (2018) (cit. on p. 79).

[215] I. Solaiman and C. Dennison,
Process for adapting language models to society (palms) with values-targeted datasets,
Advances in Neural Information Processing Systems **34** (2021) 5861 (cit. on p. 80).

[216] S. Lapuschkin, A. Binder, G. Montavon, K.-R. Muller and W. Samek,
“Analyzing classifiers: Fisher vectors and deep neural networks”,
Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016 2912 (cit. on p. 81).

[217] P. Lertvittayakumjorn and F. Toni,
Explanation-based human debugging of nlp models: A survey,
Transactions of the Association for Computational Linguistics **9** (2021) 1508
(cit. on pp. 81, 165).

[218] M. T. Ribeiro, T. Wu, C. Guestrin and S. Singh,
Beyond accuracy: Behavioral testing of NLP models with CheckList,
arXiv preprint arXiv:2005.04118 (2020) (cit. on pp. 81, 162).

[219] A. Serban, K. van der Blom, H. Hoos and J. Visser,
“Adoption and Effects of Software Engineering Best Practices in Machine Learning”,
Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), ESEM ’20,
Bari, Italy: Association for Computing Machinery, 2020, ISBN: 9781450375801,
URL: https://doi.org/10.1145/3382494.3410681 (cit. on pp. 81, 82).

[220] A. Arpteg, B. Brinne, L. Crnkovic-Friis and J. Bosch,
“Software engineering challenges of deep learning”,
2018 44th euromicro conference on software engineering and advanced applications (SEAA),
IEEE, 2018 50 (cit. on pp. 81–83).

[221] Z. U. Islam, "Software Engineering Methods for Responsible Artificial Intelligence", *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, 2021 1814 (cit. on p. 82).

[222] E. Breck, S. Cai, E. Nielsen, M. Salib and D. Sculley, "The ML test score: A rubric for ML production readiness and technical debt reduction", *2017 IEEE International Conference on Big Data (Big Data)*, IEEE, 2017 1123 (cit. on pp. 82, 83).

[223] K. Steenstrup, R. Sallam, L. Eriksen and S. F. Jacobson, *Industrial Analytics Revolutionizes Big Data in the Digital Business*, https://www.gartner.com/en/documents/2826118, 2014 (cit. on pp. 107, 236).

[224] P. J. Guo, *Software tools to facilitate research programming*, Stanford University, 2012 (cit. on pp. 108, 237).

[225] *Introduction to SEMMA*, https://documentation.sas.com/doc/en/emref/14.3/n061bzurmej4j3n1jnj8bbjjm1a2.htm, Accessed: 2022-06-19 (cit. on pp. 109, 241, 242).

[226] H. J. G. Palacios, R. A. J. Toledo, G. A. H. Pantoja and Á. A. M. Navarro, *A comparative between CRISP-DM and SEMMA through the construction of a MODIS repository for studies of land use and cover change*, Adv. Sci. Technol. Eng. Syst. J **2** (2017) 598 (cit. on pp. 109, 241).

[227] A. Azevedo and M. F. Santos, *KDD, SEMMA and CRISP-DM: a parallel overview*, IADS-DM (2008) (cit. on pp. 109, 241).

[228] A. Dåderman and S. Rosander, *Evaluating frameworks for implementing machine learning in signal processing: A comparative study of CRISP-DM, semma and kdd*, 2018 (cit. on pp. 109, 241).

[229] J. Thomas, *Operationalizing AI — Managing the End-to-End Lifecycle of AI*, https://medium.com/inside-machine-learning/ai-ops-managing-the-end-to-end-lifecycle-of-ai-3606a59591b0, Accessed: 2022-09-21 (cit. on pp. 134–136, 257).

[230] *How much does GPT-4 cost?*, https://help.openai.com/en/articles/7127956-how-much-does-gpt-4-cost, Accessed: 2023-08-11 (cit. on pp. 145, 160).

[231] J. Zhao, *LLMDataHub: Awesome Datasets for LLM Training*, https://github.com/Zjh-819/LLMDataHub, Accessed: 2023-05-23 (cit. on p. 149).

[232] J. Gao, A. Koronios and S. Selle, *Towards a process view on critical success factors in Big Data analytics projects*, 2015 Americas Conference on Information Systems, AMCIS 2015 (2015) 1 (cit. on p. 150).

[233] TII, *Falcon-40B*, URL: https://huggingface.co/tiiuae/falcon-40b (cit. on p. 152).

[234] S. Brown, *Why it's time for 'data-centric artificial intelligence'*, https://mitsloan.mit.edu/ideas-made-to-matter/why-its-time-data-centric-artificial-intelligence, Accessed: 2023-068-11 (cit. on p. 153).

[235] K. Lee et al., *Deduplicating Training Data Makes Language Models Better*, 2022, arXiv: 2107.06499 [cs.CL] (cit. on p. 153).

[236] R. Smith, J. A. Fries, B. Hancock and S. H. Bach,
Language models in the loop: Incorporating prompting into weak supervision,
arXiv preprint arXiv:2205.02318 (2022) (cit. on p. 156).

[237] M. Sabou, K. Bontcheva, L. Derczynski and A. Scharl,
"Corpus annotation through crowdsourcing: Towards best practice guidelines.", *LREC*,
Citeseer, 2014 859 (cit. on p. 157).

[238] E. Ma, *NLP Augmentation*, https://github.com/makcedward/nlpaug, Accessed: 2023-08-11,
2019 (cit. on p. 158).

[239] J. Pfeiffer et al., "AdapterHub: A Framework for Adapting Transformers",
Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 2020 46 (cit. on p. 159).

[240] Google, *Google Cloud - Vertex AI - Get Embeddings*,
https://cloud.google.com/vertex-ai/docs/generative-ai/embeddings/get-text-embeddings,
Accessed: 2023-08-11, 2023 (cit. on p. 159).

[241] Google, *Google Cloud - Vertex AI - SentimentDetection*,
https://cloud.google.com/vertex-ai/docs/text-data/sentiment-analysis/prepare-data?hl=de,
Accessed: 2023-08-11, 2023 (cit. on p. 159).

[242] B. P. Majumder et al.,
"Representation learning for information extraction from form-like documents",
proceedings of the 58th annual meeting of the Association for Computational Linguistics,
2020 6495 (cit. on p. 163).

[243] R. Rodrigues, H. Costa and M. Rocha,
"Development of a machine learning framework for biomedical text mining", *International Conference on Practical Applications of Computational Biology & Bioinformatics*,
Springer, 2016 41 (cit. on p. 163).

[244] J. Rasley, S. Rajbhandari, O. Ruwase and Y. He, "Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters", *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020 3505 (cit. on p. 163).

[245] J. Bergstra and Y. Bengio, *Random search for hyper-parameter optimization.*,
Journal of machine learning research **13** (2012) (cit. on p. 164).

[246] A. Klein, S. Falkner, S. Bartels, P. Hennig and F. Hutter,
"Fast bayesian optimization of machine learning hyperparameters on large datasets",
Artificial intelligence and statistics, PMLR, 2017 528 (cit. on p. 164).

[247] T. Elsken, J. H. Metzen and F. Hutter, *Neural architecture search: A survey*,
The Journal of Machine Learning Research **20** (2019) 1997 (cit. on p. 164).

[248] K. Nguyen and B. O'Connor,
Posterior calibration and exploratory analysis for natural language processing models,
arXiv preprint arXiv:1508.05154 (2015) (cit. on p. 164).

[249] C. Xu and J. McAuley,
"A survey on model compression and acceleration for pretrained language models",
Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, 9, 2023 10566
(cit. on p. 164).

[250] S. Pitis, M. R. Zhang, A. Wang and J. Ba,
Boosted Prompt Ensembles for Large Language Models,
arXiv preprint arXiv:2304.05970 (2023) (cit. on p. 164).

[251] C. Harrison, *LangChain*, `https://github.com/hwchase17/langchain`,
Accessed: 2023-08-11 (cit. on p. 170).

[252] Nvidia, *NeMo Guardrails*, `https://github.com/NVIDIA/NeMo-Guardrails`,
Accessed: 2023-06-30, 2023 (cit. on p. 171).

[253] B. Dehant,
I made a flappy bird clone with GPT4 and Midjourney in under an hour and you can do it too!,
`https://bootcamp.uxdesign.cc/i-made-a-flappy-bird-clone-with-gpt4-and-midjourney-in-under-an-hour-and-you-can-do-it-too-7847bc509431`,
Accessed: 2023-08-11 (cit. on p. 200).

[254] T. Schick et al., *Toolformer: Language models can teach themselves to use tools*,
arXiv preprint arXiv:2302.04761 (2023) (cit. on p. 200).

[255] H. Chase, *langchain*, `https://github.com/hwchase17/langchain`,
Accessed: 2023-06-30, 2022 (cit. on p. 200).

[256] H. Yang, S. Yue and Y. He,
Auto-GPT for Online Decision Making: Benchmarks and Additional Opinions,
arXiv preprint arXiv:2306.02224 (2023) (cit. on p. 200).

[257] S. Hong et al., *MetaGPT: Meta Programming for Multi-Agent Collaborative Framework*,
2023, arXiv: `2308.00352 [cs.AI]` (cit. on p. 200).

[258] OECD, *Data-Driven Innovation*, 2015 456,
URL: `https://www.oecd-ilibrary.org/content/publication/9789264229358-en`
(cit. on p. 218).

[259] C. Lim et al., *From data to value: A nine-factor framework for data-based value creation in information-intensive services*,
International Journal of Information Management **39** (2018) 121, ISSN: 0268-4012, URL:
`https://www.sciencedirect.com/science/article/pii/S0268401217300816`
(cit. on p. 218).

[260] K. Collier,
Agile Analytics: A Value-driven Approach to Business Intelligence and Data Warehousing,
Agile software development series, Addison-Wesley, 2012, ISBN: 9780321504814,
URL: `https://books.google.de/books?id=YMR3HynGQjUC` (cit. on p. 227).

List of Figures

List of Tables