Explainable Resource-Aware Representation Learning via Semantic Similarity

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

vorgelegt von

Eduardo Alfredo Brito Chacón
aus
Caracas, Venezuela

Bonn, 2023

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

1. Gutachter: Prof. Dr. Christian Bauckhage

2. Gutachter: Prof. Dr. Stefan Wrobel

Tag der Promotion: 20.10.2023

Erscheinungsjahr: 2023

Abstract

The rapid advancement of artificial intelligence (AI) systems in recent years is largely due to the impressive capabilities of artificial neural networks. Their powerful capabilities in natural language understanding and computer vision have paved the way for the wide adoption of AI solutions. However, these models often demand significant computational resources and operate as "black boxes", limiting their utility in sensitive domains, such as finance and healthcare, where strict personal data protection regulations apply.

This thesis addresses the triadic trade-off between accuracy, explainability, and resource consumption in the context of supervised learning, with an emphasis on representation learning for text applications. It starts presenting three use cases: semantic segmentation for autonomous driving, sentiment analysis via language models, and text summary evaluation. These cases underscore the need for robust evaluation techniques to enhance system trustworthiness but also highlight their limitations, motivating the development of RatVec, an explainable, resource-efficient framework leveraging kernel PCA and k-nearest neighbors, which is presented subsequently. RatVec demonstrates a competitive performance under certain conditions, especially when tasks can be represented as sequence similarity problems, e.g., protein family classification. For situations where RatVec is less suitable, such as text classification, the thesis proposes an analogous pipeline using Transformer-based text representations. This approach, when fine-tuned, approximates the accuracy from pure neural models while maintaining architectural explainability, and enables granular explanations of semantic similarity via a novel technique of pairing contextualized best-matching tokens.

In sum, this thesis advances the pursuit of trustworthy AI systems by introducing RatVec, a resource-efficient, explainable framework optimally suited to settings that are naturally translatable to sequence similarity problems, and proposing an explainable Transformer-based pipeline for text classification tasks. These advancements address some of the challenges of deploying AI in sensitive domains and suggest several promising avenues for future research.

Acknowledgements

My PhD journey coincided with my tenure at Fraunhofer IAIS. I am immensely thankful to my colleagues across different teams whose varied contributions significantly helped in the successful completion of this thesis.

I start thanking my first supervisor Prof. Dr. Christian Bauckhage, whose optimistic and inspiring attitude was key to maintain the motivation necessary to progress in my research. I extend my thanks to Prof. Dr. Stefan Wrobel for serving as a second supervisor. I also want to express my gratitude to Dr. Nico Piatkowski for helping me to draft my PhD proposal, and Dr. Stefan Rüpping for keeping track on my thesis and for taking the time to review parts of it.

I am also very grateful to all my NLU colleagues for our collaborative discussions, mutual support, and shared experiences. I would like to give special thanks to our team lead Sven Giesselbach, whose support enabled me to progress in my PhD amidst a challenging project load. I thank Dr. Jörg Kindermann for mentoring me during my first months in the team. When thinking about my NLU team, I cannot forget to thank Katharina Beckh, Katrin Klug and Nilesh Chakraborty for the sometimes entertaining, sometimes rather relieving (virtual) coffee breaks.

Other Fraunhofer colleagues influenced me notably during this period. I would like to particularly thank Laura von Rüden and Kostadin Cvejoski for the very inspiring and motivating discussions. At certain moments, their input was the key factor that prevented me from abandoning the pursuit of becoming a PhD.

Last but certainly not least, I thank my parents for their unending support and for always cheering me up, especially during the most challenging times.

List of Publications

A large extent of the ideas and results shown in this thesis have been previously elaborated among the following published conference proceedings and journal articles:

(i) Eduardo Brito and Henri Iser (2023). "MaxSimE: Explaining Transformer-based Semantic Similarity via Contextualized Best Matching Token Pairs." In: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '23. Association for Computing Machinery. Taipei, Taiwan. DOI: 10.1145/3539618.3592017.

(ii) Eduardo Brito, Vishwani Gupta, Eric Hahn and Sven Giesselbach (2022). "Assessing the Performance Gain on Retail Article Categorization at the Expense of Explainability and Resource Efficiency." In: KI 2022: Advances in Artificial Intelligence - 45th German Conference on AI. Cham, Springer, pp. 45–52. Trier, Germany (online). DOI: 10.1007/978-3-031-15791-2_5.

(iii) Julia Rosenzweig, Eduardo Brito, Hans-Ulrich Kobialka, Maram Akila, Nico M. Schmidt, Peter Schlicht, Jan David Schneider, Fabian Hüger, Matthias Rottmann, Sebastian Houben and Tim Wirtz (2021). "Validation of simulation-based testing: Bypassing domain shift with label-to-image synthesis." In: 2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops). IEEE, pp. 182–189. Nagoya, Japan (online). DOI: 10.1109/IVWorkshops54471.2021.9669248.

(iv) Marijn Schraagen, Joanna Wall and Eduardo Brito (2020). "The CLIN30 shared task: Have-doubling in historical varieties of Dutch." In: Computational Linguistics in the Netherlands Journal 10, pp. 161–178.

(v) Maren Pielka, Rajkumar Ramamurthy, Anna Ladi, Eduardo Brito, Clayton Chapman, Paul Mayer and Rafet Sifa (2020). "Fraunhofer IAIS at FinCausal 2020, Tasks 1 & 2: Using Ensemble Methods and Sequence Tagging to Detect Causality in Financial Documents." In: Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation. COLING, pp. 64–68. Barcelona, Spain (online).

(vi) David Biesner, Eduardo Brito, Lars Patrick Hillebrand and Rafet Sifa (2020). "Hybrid Ensemble Predictor as Quality Metric for German Text Summarization: Fraunhofer IAIS at GermEval 2020 Task 3." In: Proceedings of the 5th Swiss Text Analytics Conference and the 16th Conference on Natural Language Processing. SwissText/KONVENS 2020. Zurich, Switzerland (online).

(vii) Eduardo Brito, Bogdan Georgiev, Daniel Domingo-Fernández, Charles Tapley Hoyt and Christian Bauckhage (2019). "RatVec: A General Approach for Low-dimensional Distributed Vector Representations via Rational Kernels." In: Proceedings of the

Conference on "Lernen, Wissen, Daten, Analysen", pp. 74-78. Berlin, Germany.

(viii) Eduardo Brito, Rafet Sifa, Christian Bauckhage, Rüdiger Loitz, Uwe Lohmeier and Christin Pünt (2019). "A Hybrid AI Tool to Extract Key Performance Indicators from Financial Reports for Benchmarking." In: Proceedings of the ACM Symposium on Document Engineering 2019. DocEng '19. Association for Computing Machinery. Berlin, Germany. DOI: 10.1145/3342558.3345420.

(ix) Eduardo Brito, Rafet Sifa and Christian Bauckhage (2019). "Two Attempts to Predict Author Gender in Cross-Genre Settings in Dutch." In: Proceedings of the Shared Task on Cross-Genre Gender Prediction in Dutch at CLIN29 (GxG@CLIN29) co-located with the 29th Conference on Computational Linguistics in The Netherlands (CLIN29). Groningen, The Netherlands.

(x) Merijn Beeksma, Maarten Van Gompel, Florian Kunneman, Louis Onrust, Bouke Regnerus, Dennis Vinke, Eduardo Brito, Christian Bauckhage and Rafet Sifa (2018). "Detecting and correcting spelling errors in high-quality Dutch Wikipedia text." In: Computational Linguistics in the Netherlands Journal 8, pp. 122–13.

(xi) Eduardo Brito, Rafet Sifa and Christian Bauckhage (2017). "KPCA Embeddings: an Unsupervised Approach to Learn Vector Representations of Finite Domain Sequences." In: Proceedings of the Conference on "Lernen, Wissen, Daten, Analysen". Rostock, Germany.

(xii) Eduardo Brito, Rafet Sifa, Kostadin Cvejoski, César Ojeda and Christian Bauckhage (2017). "Towards German word embeddings: A use case with predictive sentiment analysis." In: Data Science–Analytics and Applications. Springer Vieweg, Wiesbaden, pp. 59–62. DOI: 10.1007/978-3-658-19287-7_8

Other Contributions

Released Repositories

Some of the listed publications reference to open source repositories, allowing to reproduce and extend the proposed approaches.

(i) MaxSimE: Explaining Transformer-based Semantic Similarity via Contextualized Best Matching Token Pairs. https://github.com/fraunhofer-iais/MaxSimE.
(ii) RatVec: A General Approach for Low-dimensional Distributed Vector Representations via Rational Kernels https://github.com/ratvec/ratvec.

Supervised Work

A Master's thesis started to be supervised in the scope of the work presented in this thesis:

- Henri Iser. "Improved Explainability of Transformer-based Text Classification through Late-Interaction". From April 2023.

Two students were supervised in the scope of the *Lab Development and Application of Data Mining and Learning Systems: Data Science and Big Data* at the University of Bonn:

- Mahnaz Mirhaj. "Fine-tuning Sentence Transformers for Similarity-Based Multi-label Text Classification." Winter semester 2022-2023.
- Henri Iser. "Distilling Explainable Semantic Textual Similarity Functions from Pretrained Transformers." Winter semester 2021-2022.

Contents

Chapter 1

Introduction

1.1 Motivation and Research Context

The current wave of artificial intelligence (AI) is overwhelmingly driven by the remarkable success of deep neural networks (DNNs) over the past few years across various fields, including autonomous driving, molecular biology, and text classification. Aiming to foster the widespread adoption and trust in AI systems, machine learning (ML) researchers tend to focus on three different aspects when proposing new approaches:

(i) **Accuracy**: the aim is to beat previous best performing models according to some evaluation metric.
(ii) **Explainability**: understanding why the model produced an specific output.
(iii) **Resource efficiency**: reducing runtime and/or memory requirements.

Unfortunately, these three goals are hard to achieve at the same time and constitute a "tradeoff triangle", graphically depicted in Figure 1.1. For instance, let's consider an autonomous driving system that employs a ML model to identify and classify objects on the road. High performance (accuracy) is essential in this scenario, as errors could potentially lead to accidents. However, should an accident occur, it would be crucial to understand why the model made a particular decision (explainability). In addition, given the embedded nature of such systems, the model must be capable of running on hardware with limited resources (resource efficiency). DNNs have become a standard tool when the best possible predictions are aimed. However, their superiority on many tasks when enough data is available comes with vast computation and memory requirements, which leads to high energy consumption as well. We often need a GPU even if we only need to fine-tune a pretrained DNN, sometimes even for inference. This disables deep learning models in scenarios where hardware resources are limited and when computation in the cloud is not an option, for example in mobile devices with limited or no internet connection. Furthermore, due to the notable energy consumption of these models (García-Martín et al., 2019), the research community is encouraged to embrace computationally efficient hardware and algorithms not only because of environmental reasons but also not to make research feasible only for those who can afford high computational resources (Strubell, Ganesh, and McCallum, 2019). Besides that, DNNs are black box models whose decisions cannot be really explained if the user de-

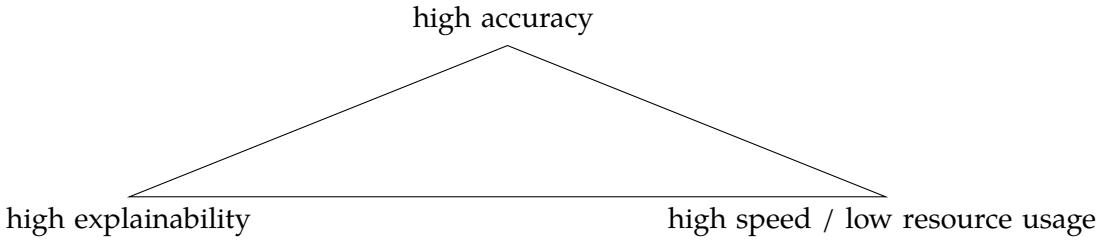


Figure 1.1: The tradeoff triangle: these three aspects are desirable at ML applications but are also hard to obtain simultaneously.

mands it. This inconvenience disqualifies DNNs in areas where an automated decision making algorithm is required to provide explanations about any generated output, including healthcare, finance, and law. Therefore, we aim to find representation learning techniques that find a better balance between accuracy, explainability and resource consumption e.g., achieving more explainable models at the cost of a small performance drop.

Most of the approaches and results presented in this thesis were achieved at research projects at the Lamarr Institute for Machine Learning and Artificial Intelligence (or its predecesor ML2R competence center) and are motivated by projects with industry partners at Fraunhofer IAIS. The resulting publications constitute the core of this dissertation, which can be considered as a unifying outline of their various topics. An example from industry projects motivating this research is presented in Section 3.1. In another one, we developed a system for extracting Key Performance Indicators (KPIs) from financial reports (Brito, Sifa, Bauckhage, et al., 2019). These documents are populated with tables and text boxes that we could detect with Convolutional Neural Networks (CNNs). Following this, we used rule-based models to identify the relevant tables and text boxes. Subsequently, to extract the KPIs from the parsed components, we trained tree-based classifiers on interpretable features. A visualization of the whole process (as displayed in Figure 1.2) added another layer of clarity, enabling us to track the progress of the information extraction and facilitate troubleshooting, while demonstrating the process to stakeholders as well. These aspects were crucial to the success of this project: thanks to them, we could not only increasingly improve the accuracy of the extracted KPIs by understanding the causes of failure, but we could also increase user trust by incorporating their feedback and domain expertise into the models. This use case underscores the necessity of explainability and high accuracy for achieving user acceptance.

1.2 Research Questions

The main goal of this dissertation is to study and propose representation learning techniques, in particular for natural language understanding (NLU) applications, that enable the three aspects of the mentioned tradeoff triangle to a certain extent simultaneously. To this end, and considering the aforementioned research context, the main research question is:

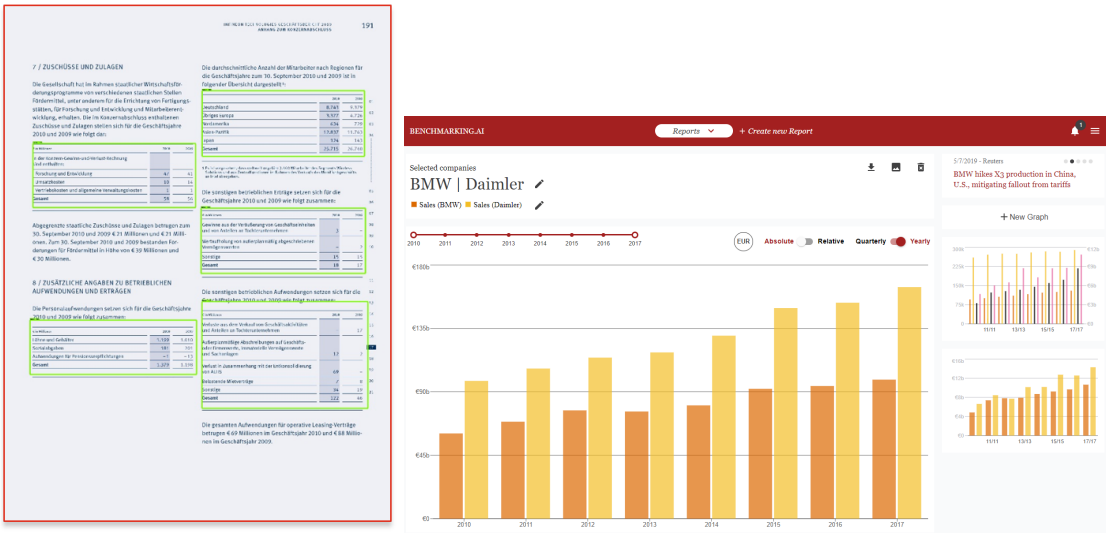


Figure 1.2: Example of table detection using a CNN (left) and visualization of the KPI "turnover" values extracted from yearly business reports of two car manufacturers. Explainability by visualization and feature importance from interpretable features played a role for troubleshooting the underlying models and increasing user trust in the system. Images previously published by Brito, Sifa, Bauckhage, et al., 2019.

- Can we learn competitive ML models that are not only more resource-aware but also explainable? This general question motivates the following more specific research questions.
- Can a sound statistical evaluation provide any guarantee that alleviates the lack of explainability to enable trustworthy systems?
- To what extent can we exploit lightweight interpretable approaches to encode (explainable) similarity among processed entities?
- Can we perform explainable similarity-based classification without a significant drop in performance compared to transformer-based models?
- What are the saved computational resources by using the proposed models?

1.3 Outline

The remaining chapters of this thesis are structured as follows.

Chapter 2 presents background information on ideas and methods that helps to follow the discussions from the following chapters.

Chapter 3 presents three use cases where we used evaluation as a mean to achieve trustworthy systems: semantic segmentation for autonomous driving systems, language modelling for sentiment analysis, and automatic text summarization. Each use case utilizes different evaluation methods to enhance trust in the system and exposes limitations of using black-box models.

Chapter 4 introduces the RatVec framework as an alternative to pure neural methods

for reducing computational requirements and increasing model interpretability. The RatVec framework generates vector representations using rational kernels within kernel PCA, which were later used for classification tasks solved by a k-nearest neighbors (KNN) classifier, making it resource-efficient and explainable in the sense that elements are classified according to similar elements from the training set, where the similarity is given by the applied kernel function.

Chapter 5 focuses on a pipeline that text representations from foundation models and a KNN classifier, which is structurally similar to the RatVec framework. However, instead of generating vector representations using rational kernels, it leverages Transformer-based models, specifically Sentence-BERT (SBERT), to extract text representations. This chapter starts comparing the performance gap between the proposed pipeline and a pure BERT-based classifier, it continues proposing an approach to fine-tune the SBERT model for multi-label text classification to narrow the measured gap, and it concludes proposing an explanation method for the classifications delivered by this pipeline, which does not only show the most similar texts as explanations but also highlights the tokens that contribute most to the measured semantic similarity.

Chapter 6 provides a summary of the thesis, discussing the key insights and suggesting directions for future research.

To sum up, this dissertation addresses the complex tradeoff between accuracy, explainability, and resource requirements in ML models. By exploring and proposing representation learning techniques, such as the RatVec framework and SBERT text representations within a similarity-based classification pipeline, we seek to develop solutions that are more interpretable, resource-efficient, and maintain a high level of accuracy. This thesis provides insights into the benefits of these techniques in a variety of contexts, from protein family classification to multi-label text classification. The potential impact can be significant: more interpretable models mean more transparency and trust in AI systems, and more resource-efficient models open up the use of AI in resource-constrained environments and contribute to more sustainable practices.

Chapter 2

Preliminaries

This chapter introduces a set of concepts and methods that will be utilized across the thesis and may help readers that are less familiar with the involved topics.

2.1 Word Embeddings

ML approaches for NLU generally demand a numeric vector representation for words. We can distinguish any two different words from a fixed vocabulary by assigning them a one-hot vector, where all entries of the vector are zero-valued but in a single position that identifies the word. This is a very sparse representation that encodes no information about the words but their position in the vocabulary.

A more information-rich alternative to one-hot vectors are the so-called *word embeddings*. They are distributed vector representations, which are dense, low-dimensional, real-valued and can capture latent features of the word (Turian, Ratinov, and Bengio, 2010). Based on the *distributional hypothesis* from Harris, 1954 (words that appear in similar contexts have similar meanings), they exploit word co-occurrence so that similar words are mapped close to each other in the word vector space. These word vectors encode syntactical and semantical information so that some NLU tasks can be solved by simple linear vector operations thanks to the distributed nature of the word representations. For example, we can answer analogy questions just with additions and subtractions: if we subtract from a vector representing the word "Madrid" the vector corresponding to "Spain" and we add the vector "France", the resulting vector should be very close to the vector "Paris" (Mikolov, Sutskever, et al., 2013). These vector word representations have also the advantage that they can be trained efficiently by simple (shallow) neural networks such as the continuous Skip-gram model (SG) and the Continuous Bag of Words model (CBOW) (Mikolov, K. Chen, et al., 2013), both popularized after the release of the `word2vec` software. Beyond NLU, word embeddings have also inspired research in other areas, such as graph theory, where similar techniques are used to learn node representations (Grover and Leskovec, 2016).

Although syntax and semantics can be encoded with word2vec embeddings, they do not incorporate morphological information about the word. As a consequence, morphologically similar words may not be nearby in the word vector space. Some ap-

proaches make use of existing linguistic resources so that the word embeddings capture not only contextual information but also morphological information (Cotterell and Schütze, 2015; Jurdzinski et al., 2016). Due to the their dependence on language-specific resources, they will not work for languages whose available linguistic resources are scarce.

The SG network is formally defined to predict nearby words. However, we focus on the the resulting hidden layer weights after the training phase: they constitute dense vector representations of words. Words are presented to the network with one-hot encoding. We can model a projection from the one-hot vector to its embedding by means of an identity transfer function. In contrast to the most common neural networks models which require a non-linear transfer function in the hidden layer, the SG model does not use any non-linear transfer function in the hidden layer but only in the output layer. The number of necessary neurons in the hidden layer is determined by the number of features that we want our word embeddings to have: if V is size of the vocabulary and N the size of our word vectors, we can represent the weights as a $V \times N$ matrix, in which each row i is the embedding of the corresponding word placed in the position i within the vocabulary. The context of a word token is defined by setting a maximum window size m so that for a token sequence w and a token in position t, the context of the token $w(t)$ is made of tokens that are at a maximum distance m' from the central word $w(t)$ (excluding the central word itself), where m' is sampled from the interval $[1, m]$. By choosing the window size stochastically for each example, we ensure that word tokens that appear closer to the central word get more importance, as they are more likely to fit within the context window

$$w(t - m') \cdots w(t - 1)w(t + 1) \cdots w(t + m'). \tag{2.1}$$

The output layer performs multinomial logistic regression (without bias term). Since we do not care about the prediction accuracy but about the quality of the word embeddings, the original formulation using softmax as output layer transfer function is simplified by using *negative sampling* or *hierarchical softmax* (Mikolov, Sutskever, et al., 2013). This is specially important to efficiently train the embeddings since all matrix calculations needed for the softmax activation function are computationally expensive.

The CBOW model works in a similar way but defined for the inverse task: in this case, the context words are the input of the network, which are used to predict the central word they surround. Also, this prediction task is used only to learn the hidden layer weights, which correspond to word embeddings exactly as in the SG model.

The *Paragraph Vector* model (Le and Mikolov, 2014) extends both mentioned models so that also a set of words such as a sentence or a whole document can be represented as a vector (the so-called paragraph vector). Paragraph vectors can be derived using two different architectures: the Distributed Bag of Words (PV-DBOW) and Distributed Memory (PV-DM) of Paragraph Vectors, which are respectively similar to SG and CBOW. They consider each paragraph as another token belonging to the vocabulary that appears in all context windows during the training phase. This approach improves text classification performance compared to using a mean of word embeddings. For more details about these models we refer the reader to Le and Mikolov, 2014.

2.2 Topic Modeling

Topic modeling can be described as a dimensionality reduction technique applied in classification and clustering tasks. Unlike word embeddings, topic modeling provides a more interpretable approach to creating vector representations of text. The goal of topic models is to discover hidden semantic structures that we call *topics*. In the context of topic modeling, documents are viewed as a blend of different topics, with each document possessing a certain probability of belonging to each topic. Despite the prominence of DNNs for classification, topic models hold the advantage of explainability as we can identify the words that are most relevant for each topic.

The most commonly utilized topic models are grounded on the distributional hypothesis. The most popular one is Latent Dirichlet Allocation (LDA) (Blei, Ng, and Jordan, 2003). LDA is a generative statistical model that allows sets of observations to be attributed to unobserved groups. These groups are our targeted topics, and the observed data are the documents and the words within the documents. Latent semantic analysis (LSA) (Dumais, 2004) alternatively leverages Singular Value Decomposition (SVD) on the Term-Document Matrix to identify the relationships between words and the concepts they form in the text data.

More recent alternatives employ non-negative matrix factorization (NMF) (Arora, Ge, and Moitra, 2012; Hillebrand et al., 2021). NMF operates as a method that simultaneously performs dimensionality reduction and clustering. The non-negativity constraint of NMF results in a parts-based representation. This is because NMF decomposes the original data into two non-negative matrix factors, which can be interpreted as the topics and their associated weights. Therefore, each topic is represented by a cluster of words (parts) that frequently occur together in the data.

Other methods like Anchored CorEx (Gallagher et al., 2017) adopt an information-theoretic approach instead of relying on a probabilistic generative model.

In the next section, we will discuss Transformer-based models such as the BERT model, which have also been used to support certain topic modeling techniques (Grootendorst, 2020). These methods utilize the contextual relationships that Transformer models capture to derive topics. However, these models can be computationally intensive and the interpretability of their topics may not be as straightforward as those obtained from more "traditional" non-neural methods.

2.3 Transformer Networks

Inspired by the success of transfer learning techniques in computer vision, which involve pretraining a DNN and then fine-tuning it for a different task, transfer learning has been adapted to NLU in recent years. This adaptation has taken the form of pretrained language models and has led to significant advancements in NLU tasks. A pivotal developments in this domain has been the introduction of the Transformer model Vaswani et al., 2017. The Transformer model, with its attention mechanisms, initiated the current state-of-the-art paradigm for pretrained language models.

In contrast to previous pretrained word embeddings like word2vec (Mikolov, K. Chen, et al., 2013; Mikolov, Sutskever, et al., 2013) that provide static vector representa-

tions for words or text fragments, Transformer models offer context-dependent representations after being pretrained on large corpora. These context-dependent representations, such as Bidirectional Encoder Representations from Transformers BERT (Devlin et al., 2019), have demonstrated superior performance over classical word embeddings in a variety of benchmark NLU tasks. This success has motivated the exploration of Transformer models in other domains, including bioinformatics (Clauwaert and Waegeman, 2019; Rives et al., 2019; Ingraham et al., 2019) and multi-horizon forecasting (Lim et al., 2019).

Among Transformer-based models, SentenceBERT (SBERT) (Reimers and Gurevych, 2019) is a BERT-based model specifically designed to produce contextualized representations for sentences or short texts. It is intended such that the semantic similarity of two sentences correlates with the (cosine) similarity of their respective embeddings. To achieve this, SBERT employs a Siamese network architecture during training, which encodes two separate sentences using two identical copies of the same BERT model.

The wide-scale adoption of these models across various tasks has led to them being referred to as *foundation models*. They are typically first trained on vast amounts of data without supervision and are later fine-tuned on downstream tasks.

2.4 Knowledge Distillation

Knowledge distillation has been proposed as a technique to train smaller models while maintaining performance levels comparable to those of their larger "teacher" models (Buciluǎ, Caruana, and Niculescu-Mizil, 2006; Hinton, Vinyals, and Dean, 2015). For instance, DistilBERT significantly reduces the size of a BERT model by 40%, while retaining 97% of its language understanding capabilities and offering a 60% speed improvement (Sanh et al., 2019). In an effort to extract interpretable insights from deep neural networks (DNNs), several distilled models have been proposed, including decision trees, finite state automata, graphs, and rule-based models (N. Xie et al., 2020). Despite these advances, these distilled models may still be unsuitable for low-resource environments or may not achieve the performance levels of their equivalent DNNs.

2.5 Kernel PCA

Kernel PCA (KPCA) (Bernhard Schölkopf, Alexander Smola, and Müller, 1998) is a kernel method designed to identify latent structures or *principal components* within a dataset, which can then be used for dimensionality reduction (Bernhard Schölkopf, Alexander Smola, and Müller, 1998). In contrast to (linear) principal component analysis (PCA), the principal components are not extracted from the input space but in the *feature space*, an arbitrary reproducing kernel Hilbert space with higher dimensionality than the input data. This involves solving the eigendecomposition of the Gram matrix in the feature space instead of the input space. Using the "kernel trick", dot products in the feature space can be replaced by a kernel function defined over the input space. This opens up the possibility of selecting any kernel function that suitably measures similarity for the task at hand. This freedom for the kernel choice enables performing KPCA

on non-numeric data, for example by applying string kernels on text pairs.

Defining our input data as a zero-mean matrix $\mathbf{X}$ containing n m-dimensional data points, and $\mathbf{C} = \frac{\mathbf{1}}{\mathbf{n}}\mathbf{X}\mathbf{X}^\top$ as its covariance matrix, PCA solves

$$\mathbf{C}\mathbf{v} = \lambda\mathbf{v}, \tag{2.2}$$

for the eigenvalues λ and eigenvectors $\mathbf{v}$. By expanding (2.2), we can express every eigenvector $\mathbf{v}$ as a linear combination of the data points of the data matrix $\mathbf{X}$

$$\frac{1}{n\lambda}\mathbf{X}\mathbf{X}^\top\mathbf{v} = \mathbf{X}\beta = \mathbf{v}, \tag{2.3}$$

where $\beta \in R^m$. Substituting this back into (2.2), we obtain

$$\frac{1}{n}\mathbf{X}\mathbf{X}^\top\mathbf{X}\beta = \lambda\mathbf{X}\beta \tag{2.4}$$

and left-multiplying $\mathbf{X}^\top$ on both sides of (2.4) results in

$$\frac{1}{n}\mathbf{X}^\top\mathbf{X}\mathbf{X}^\top\mathbf{X}\beta = \lambda\mathbf{X}^\top\mathbf{X}\beta. \tag{2.5}$$

Replacing the Gramian $\mathbf{X}^\top\mathbf{X}$ by a kernel matrix $\mathbf{K}$, where

$$\mathbf{K_{ij}} = \mathbf{k}(\mathbf{x_i}, \mathbf{x_k}),\ \forall \mathbf{x_i}, \mathbf{x_k} \in \mathbf{X} \tag{2.6}$$

for a selected kernel function k, one obtains

$$\frac{1}{n}\mathbf{K}\mathbf{K}\beta = \gamma\mathbf{K}\beta, \tag{2.7}$$

which, upon multiplication by $\mathbf{K^{-1}}$, results in a kernelized representation of the conventional PCA

$$\frac{1}{n}\mathbf{K}\beta = \gamma\beta, \tag{2.8}$$

After diagonalizing $\mathbf{K}$, we select the d eigenvectors $\mathbf{v_1}, \ldots, \mathbf{v_d}$, where $d \leq n$ belonging to the largest eigenvalues and divide them by their respective eigenvalues $\lambda_1, \ldots, \lambda_d$ to construct a projection matrix $\mathbf{P}$:

$$\mathbf{P} = \left[\frac{\mathbf{v_1}}{\lambda_\mathbf{1}}, \ldots, \frac{\mathbf{v_d}}{\lambda_\mathbf{d}}\right]. \tag{2.9}$$

To extract d principal components for a data point t, we evaluate the kernel function on t against all the data points in $\mathbf{X}$ to obtain a kernelized distance vector $\mathbf{r_t}$. The product of $\mathbf{r_t}$ with the projection matrix $\mathbf{P}$ constitutes the d-dimensional vector representation $\mathbf{u_t}$ of t in the feature space.

$$\mathbf{r_t} = \mathbf{k}(\mathbf{w_t}, \mathbf{X}) \qquad \text{and} \qquad \mathbf{u_t} = \mathbf{P}^\top\mathbf{r_t}. \tag{2.10}$$

2.6 Weighted Finite-State Transducers and Rational Kernels

Weighted finite-state transducers, in essence, are classical finite-state automata enhanced with an additional weight structure. Specifically:

Definition (*Cortes, Haffner, and Mohri, 2004*). A weighted finite-state transducer T over a semiring $\mathbb{K}$ is an 8-tuple

$$T = (\Sigma, \Delta, Q, I, F, E, \lambda, \rho), \tag{2.11}$$

where Σ is the finite input alphabet of the transducer; Δ is the finite output alphabet; Q is a finite set of states; $I \subseteq Q$ the set of initial states; $F \subseteq Q$ the set of final states; $E \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times (\Delta \cup \{\varepsilon\}) \times \mathbb{K} \times Q$ a finite set of transitions; $\lambda : I \to \mathbb{K}$ the initial weight function; and $\rho : F \to \mathbb{K}$ the final weight function mapping F to $\mathbb{K}$. ■

Definition. Given a transition $e \in E$ we denote its previous, next states and weight respectively by $p(e), n(e), w(e)$. Given a path $\pi = e_1 e_2 \dots e_k \in E^*, n(e_{i-1}) = p(e_i), i = 2, \dots, k$ of length k, we define

$$p(\pi) := p(e_1), \quad n(\pi) := n(e_k), \quad w(\pi) := \bigotimes_{i=1}^{k} w(e_i). \tag{2.12}$$

■

Definition. A transducer T is called regulated if the sum

$$[[T]](x, y) := \bigoplus_{\pi \in P(I,x,y,F)} \lambda(p(\pi)) \otimes w(\pi) \otimes \rho(n(\pi)), \quad \forall (x, y) \in \Sigma^* \times \Delta^* \tag{2.13}$$

converges. Here $P(I, x, y, F)$ denotes the set of paths starting at an initial state $i \in I$ and ending at a final state $f \in F$, which are labeled by an input label $x \in \Sigma^*$ and output label $y \in \Delta^*$. ■

When appropriately regulated, weighted transducers enjoy properties such as being closed under summation, products, Kleene-star operations, among others (Cortes, Haffner, and Mohri, 2004). One can also intuitively think of a transducer as a matrix over a countable set $\Sigma^* \times \Delta^*$. Thus, a composition of transducer should be analogous to matrix multiplication (Cortes, Haffner, and Mohri, 2004).

Proposition. Suppose the weighted transducers T_1, T_2 are defined over a commutative semiring $\mathbb{K}$. Provided it converges, the composition of T_1, T_2 is defined by

$$[[T_1 \circ T_2]](x, y) := \bigoplus_{z \in \Delta^*} [[T_1]](x, z) \otimes [[T_2]](z, y). \tag{2.14}$$

■

For further details on transducer composition we refer to Cortes, Haffner, and Mohri, 2004.

Definition. A kernel function

$$K : \Sigma^* \times \Delta^* \to \mathbb{R} \tag{2.15}$$

is called a rational kernel, if there exists a weighted transducer T and a function $\psi : \mathbb{K} \to \mathbb{R}$ such that

$$K(x, y) = \psi(T(x, y)), \quad \forall (x, y) \in \Sigma^* \times \Delta^*. \tag{2.16}$$

■

In many cases of learning algorithms one moreover needs the following property:

Definition. Let X be a non-empty set. A kernel function $K : X \times X \to \mathbb{R}$ is said to be a positive definite symmetric (PDS) kernel if

$$K(x_1, x_2) = K(x_2, x_1), \quad \forall (x_1, x_2) \in X \times X, \tag{2.17}$$

$$\sum_{i,j=1}^{n} c_i c_j K(x_i, x_j) \geq 0, \quad \forall n \geq 1, \forall \{x_1, \dots x_n\} \subseteq X, \forall \{c_1, \dots, c_n\} \subseteq \mathbb{R}. \tag{2.18}$$

■

Proposition (*Berg, Christensen, and Ressel, 1984*). The class of PDS kernels is closed under summation, product, tensor products, pointwise limits and power series. ■

Proposition (*Cortes, Haffner, and Mohri, 2004*). (Informal) The class of rational PDS kernels is closed under summation, products and Kleene-closures provided the mapping ψ has some morphism properties. ■

However, rational PDS kernels are not closed under composition.

2.7 Edit-Distance Families and Indefinite Learning Techniques

Edit-distances measure the similarity between two strings by evaluating the minimum cost of a sequence of edit operations such as substitutions, deletions, and insertions (Levenshtein, 1966). These techniques are integral to various tasks as they can optimize alignments.

All classical edit-distance measures can be articulated as weighted transducers, thereby allowing the use of rational kernels (Cortes, Haffner, and Mohri, 2004). This not only permits the implementation of efficient algorithms but also provides theoretical tools for constructing more task-relevant edit-distances. However, it should be noted that edit-distance kernels are not guaranteed to be PDS (Cortes, Haffner, and Mohri, 2004). Nonetheless, indefinite kernel methods can be viewed within the context of an appropriate pseudo-Euclidean space (X. Huang et al., 2016). For instance, the KPCA algorithm is perceived as an optimal variance problem in this setting. This perspective aids in theoretically explaining our empirical results, considering that our kernels are not intrinsically definite. We will refrain from introducing suitable positive definite approximations, given that our experiments using specific indefinite kernels demonstrate competitive performance (later shown in Chapter 4).

Proposition (*Cortes, Haffner, and Mohri, 2004*). Let Σ be a finite alphabet. For each $x, y \in \Sigma^*$ let the edit-distance $d_e(x, y)$ be defined as the minimal number of deletions, substitutions and insertions needed to transform x into y (that is, all three elementary operations have a fixed cost of 1). Then, d_e is not positive-definite and, moreover, d_e is negative definite if and only if Σ consists of exactly one symbol. ■

The proof proceeds via a direct check, where one uses the following central result:

Theorem (*Berg, Christensen, and Ressel, 1984*). A symmetric kernel $\kappa : X \times X \to \mathbb{R}$ is negative definite if and only if $\exp(-t\kappa)$ is positive definite for all positive numbers t. ■

Several machine learning algorithms, such as SVMs, make assumptions that necessitate some definiteness to ensure convergence. Additionally, from the interpretive viewpoint, the kernel cannot be considered as a scalar product in an appropriate Hilbert space (i.e., the kernel trick). A method to mitigate these issues is as follows:

Theorem (*B. Schölkopf and AJ. Smola, 2002*). Suppose the data points $x_1, \ldots, x_n \in X$ and the kernel function $\kappa : X \times X \to \mathbb{R}$ are such that the matrix

$$K_{ij} := \kappa(x_i, x_j), \quad 1 \leq i, j \leq n, \tag{2.19}$$

is positive-definite. Then there exists a Hilbert space H and a map $F : X \to H$ such that

$$\kappa(x_i, x_j) = \langle F(x_i), F(x_j) \rangle \tag{2.20}$$

Conversely, for a map F into some Hilbert space H, the matrix $K_{ij} := \langle F(x_i), F(x_j) \rangle$ is positive. ■

In other words, we only need to ensure that the kernel (similarity) function has an appropriate behavior when restricted to our dataset. Note that no special structure is assumed on X - in particular, one can work with non-numeric/non-symbolic entities, offering substantial flexibility.

Nonetheless, when dealing with indefinite kernel/similarity functions in learning algorithms is unavoidable, various indefinite-kernel techniques and modifications have been proposed (X. Huang et al., 2016). For instance, indefinite machine learning algorithms such as SVM and KPCA have been interpreted through distance optimization in specific pseudo-Euclidean spaces (Haasdonk, 2005) i.e., spaces equipped with a metric tensor, whose eigenvalues are not necessarily positive (a guiding intuition is given by computing distances between events in a 4-dimensional Minkowski spacetime equipped with the metric $g_M := dt^2 - dx^2 - dy^2 - dz^2$).

2.8 Explainability Methods in Classification Tasks

Explainability in machine learning refers to the transparency of the decision-making process of a model. This clarity is essential in sensitive fields like healthcare or finance where comprehending the reasoning behind a prediction is as important as the prediction itself. Explainability methods aim to elucidate how a model makes predictions,

thereby enhancing transparency, trust, and fairness, and reducing bias. These methods also facilitate model debugging and validation. We can categorize these methods into four key criteria:

- **Ante-hoc vs. Post-hoc**: Ante-hoc methods incorporate interpretability into the model design before training. These include decision tree classifiers, inherently interpretable due to their simple if-then rule structure, and attention mechanisms used extensively in Transformer-based models, which highlight the parts of the input the model focuses on during prediction. Conversely, post-hoc methods interpret models post-training, such as LIME (Local Interpretable Model-Agnostic Explanations) (Ribeiro, S. Singh, and Guestrin, 2016) and SHAP (SHapley Additive exPlanations) (Lundberg and S.-I. Lee, 2017).
- **Instance/Local vs. Model/Global**: Instance or local interpretability methods focus on explaining individual predictions. LIME is an example of a local explanation method. It explains an individual prediction by approximating the model locally with an interpretable model. SHAP also falls in the same category providing a unified measure of feature importance that allocates the contribution of each feature to the prediction of each instance. Model or global methods aim to provide an overall understanding of the decision-making process. Counterfactual explanations (Wachter, Mittelstadt, and Russell, 2017), which provide an alternative instance to a given input instance that would lead to a different model prediction, are a further example. In contrast to the latter methods, feature importance scores provided by tree-based models like random forest (Breiman, 2001) can serve as an example of global explanation. This can be done by measuring how much the output changes when a the value of a feature is varied. The feature importance is often calculated based on the number of splits a feature is involved in and the improvement to the model from each split. Also CAVs (Concept Activation Vectors) (Kim et al., 2018) provide global explanations. CAVs are directions in the representation space of the model that correspond to human-interpretable concepts. They are used to quantify the extent to which a given input activates a certain concept.
- **Specific vs. Agnostic**: Specific methods interpret particular types of models, leveraging their unique structures and features. For example, DeepLIFT (Deep Learning Important FeaTures) (Shrikumar, Greenside, and Kundaje, 2017) and visualization of attention mechanisms are methods specific to neural networks. Agnostic methods, in contrast, aim to interpret any model, regardless of its structure. LIME and SHAP, mentioned earlier, are examples of model-agnostic methods.
- **Data-Dependent vs. Data-Independent**: Data-dependent methods need access to the training data or data distribution to generate explanations. Partial Dependence Plots (PDPs) (Friedman, 2001) and Individual Conditional Expectation (ICE) plots (Goldstein et al., 2015) fall into this category as they provide insights into the relationship between feature values and the prediction by varying feature values and observing the corresponding changes in predictions. On the other hand, data-independent methods generate explanations based on the internal parameters or structure of the model, without requiring access to the original training data. The visualization of a decision tree is an example of a data-independent method.

For a more comprehensive overview of explainability methods, we refer to Burkart and Huber, 2021; Molnar, 2020.

Chapter 3

Use Cases for Trustworthy Applications

The increasing prevalence of ML models across various industrial sectors, especially in safety-critical domains such as autonomous driving (AD), medicine, and finance, has amplified the demand for trustworthy ML-powered systems. This demand is driven primarily by the necessity to verify, safeguard, and certify these models (Braiek and Khomh, 2020; J. M. Zhang et al., 2020). Many deep learning models require vast amounts of labeled and representative data for training, and even more so for systematic testing and validation. Single datasets often do not offer sufficient data diversity, making statistical assessments of how a model performs in all relevant situations impossible (e.g., near-accidents in real street scenes).

In this chapter, we explore several use cases where evaluating a black-box model becomes necessary to increase trust in the ML model. We begin with the AD case, proposing a simulation-based testing approach to validate classification models on street scenes. Our second use case focuses on evaluating German word embeddings intrinsically and determining if the top-performing models are also the best in a sentiment analysis task. We conclude this chapter by proposing an evaluation metric for German text summarization models that outperforms standard evaluation metrics frequently used in the field of automatic text summarization.

While this chapter does not directly address explainable or resource-aware representation learning methods (as in the subsequent chapters), it does highlight use cases where such methods could potentially offer user-demanded explanations.

3.1 Validation of Simulation-based Testing: Bypassing Domain Shift with Label-to-Image Synthesis

This section is an adaptation of the work presented by Rosenzweig et al., 2021, where the author of this dissertation is co-first author. As such, the author was responsible for the synthetic data generation and for conceiving the presented framework, which was jointly designed with the other co-first author Julia Rosenzweig.

3.1.1 Introduction

Testing on real data can be prohibitively expensive or even unfeasible: Some classification tasks on video for AD involve recording and manually labeling hours of street scenes or finding rare critical situations in actual driving data. This motivates training and testing models with data from simulations (Paranjape et al., 2020; Wagner et al., 2019; Dosovitskiy et al., 2017), which can be seen as a particular approach within informed ML (Rueden, Mayer, Beckh, et al., 2020; Rueden, Mayer, Sifa, et al., 2020).

Synthetic data generation essentially comes "for free": Virtual environments allow for fast and controllable generation of scenes and thereby enable higher (test) coverage and systematic statistical testing e.g. against occlusions and corner cases, which is unfeasible in real scenes. However, using synthetic data introduces a domain shift with respect to the real data that the model was trained on. In this section, we focus on the question of reliable and realistic testing of semantic segmentation models with synthetic data, namely:

To what extent can we transfer testing results from synthetic data to real data?

In particular, we investigate if testing on synthetic data uncovers exactly the same failure modes that would occur in a real environment.

We propose to tackle this issue with a modular two-stage framework that allows for an in-depth investigation beyond aggregated performance scores. It combines label-to-image synthesis with controllable simulations enabling the generation of test cases, e.g. a child jumping on the street or any situation stated in safety requirements of regulatory entities. In the first step, a paired set of semantically equivalent real and synthetic scenes allowing a direct comparison (e.g. model performance) is used to assess the transferability of testing results. If it is satisfactory,[1] in the second step test cases can be generated independently from the previous set using, e.g. a simulator to obtain labels for scene synthesis. In this way, the testing process "bypasses" the domain shift. With the perspective of avoiding repeated costly tests on real data, various participants involved in testing ML models can benefit from this workflow, be it as a developer or auditor in certification bodies.

This work is organized as follows: Section 3.1.2 outlines some topics related to our work. Section 3.1.3 is dedicated to the description of our conceptual framework, including the data generation procedure as well as the validation measures. We validate our

[1] Results on synthetic data do not need to be identical to those on real data to be useful. However, a strong correlation in, e.g. failure modes or performances for investigated classes is desirable. Due to the pairwise correspondence such quantities are directly measurable.

framework in Section 3.1.4 for the use case of semantic segmentation for AD as a proof of concept. Finally, we conclude in Section 3.1.5 discussing some open questions and giving an outlook on future work.

3.1.2 Related Work

We outline some directions of related work. One concerns the testing of ML models, others are domain adaptation and synthetic data generation.

Testing of Machine Learning Models

Our approach falls into the broader category of offline testing methods (in contrast to online testing methods, which are applied at deployment time) (Haq et al., 2020), and aims to find and test weak spots. By doing so, it opens up possibilities for simulation-based *safety argumentations* (Rudolph, Voget, and Mottok, 2018; Schwalbe and Schels, 2020; Willers et al., 2020). Conventional software-testing approaches are often not directly applicable for testing ML-based systems or need to be strongly adapted to be applicable. Hence, new *methods, measures, and evaluation techniques* are needed to argue for the safety of ML models (Braiek and Khomh, 2020; J. M. Zhang et al., 2020). In *statistical model checking* (Barbier et al., 2019), ML models are validated using KPIs of interest in combination with statistics and modeling the ML components as probabilistic systems. In this paper and in many other testing approaches, *simulation-based testing* is deployed in which simulators are used to create testing data (Dosovitskiy et al., 2017; Paranjape et al., 2020; Wagner et al., 2019). However, the frameworks proposed in this context in part or entirely neglect validating their results in real-world situations. Our approach addresses exactly this shortcoming by proposing a framework to assess to what degree testing results obtained on synthetic data are realistic. Closest to our contribution is the work by Wagner et al., 2019. The authors propose an approach to locally verify the use of simulation data for testing. Our approach differs significantly in two main aspects: First, we do not apply any formalism or scene description language to produce corresponding synthetic data from the real data. Instead, we use the real labels as input to a generative model to produce a paired synthetic dataset (see Subssection 3.1.3). Additionally, due to the use of the generative model (also on simulation labels, see Section 3.1.3), we can extend the input space coverage not only locally and allow for completely new (controllable) scenarios to be generated and tested against. This is in line with the literature about *scenario-based testing* in the context of AD (Neurohr et al., 2020; Bussler et al., 2020).

Domain Adaptation and Synthetic Data Generation

While simulations may help with the aforementioned challenges, synthesizing data for testing creates a domain mismatch w.r.t. the data the system will process when deployed in real-world settings. Most *domain adaptation* approaches aim to measure and minimize the domain gap between a source and the desired target domain during the training phase so that the system generalizes well on the target domain (Kouw and Loog, 2019; Mei and Deng, 2018). In contrast, we aim to validate transferability of testing results on

synthetic data to real-world data regardless of the concrete magnitude of the domain gap. To this end, we apply a generative adversarial label-to-image synthesis model. In the particular case of video data employed for our proof of concept, *video-to-video synthesis* (D. Chen et al., 2017; T.-C. Wang et al., 2018) can serve as a generative model to synthesize photo-realistic time-consistent videos given an input video. This task can be defined as a distribution matching problem that can be solved by conditional generative adversarial models. It can also be understood as an *image-to-image translation* problem, (Isola et al., 2017; T. Wang et al., 2018), additionally introducing the necessary temporal dynamics so that the synthesized image sequences are temporally coherent.

Other approaches try to directly learn generative models for synthesizing realistic scenes that have a low domain gap compared to the real data (Kar et al., 2019; Devaranjan, Kar, and Fidler, 2020), or learn models that improve the realism of given simulated images (Shrivastava et al., 2017). Close to the idea for our data processing procedure is (Gaidon et al., 2016), where the authors create a simulated world that is cloned from the corresponding real-world dataset Kitty (Geiger, Lenz, and Urtasun, 2012).

3.1.3 Approach

Our ultimate goal is to assess whether the output, which the system under test produces on synthetic data is the same as the one produced on corresponding real-life data. Here the correspondence is defined through representations of the same abstract events in the real and the synthetic domain. For instance, the abstract scenario of driving near a crosswalk with pedestrians has representations in both the real, e.g. an image recorded directly on the street, and synthetic domain. Without domain gap, testing the network on both representations should ideally lead to exactly the same output. However, as domain gaps can occur in practice, we propose a qualitative framework to validate the transferability of the testing results by exploring this gap under controlled circumstances. Although we detail it for semantic segmentation, it is also applicable for other multi-class classification tasks with slight modifications.

Our framework consists of the following basic components:

(i) An ML model (system under test) trained for a specific semantic segmentation task on real data featuring the target appearance[2],
(ii) a labeled real-world testing dataset featuring the target appearance,
(iii) a generative label-to-image transfer model, which synthesizes data featuring the target appearance from segmentation masks (or from classification labels with sufficient semantic information e.g., from meta data)
(iv) a controllable simulation engine that allows us to create labeled synthetic data of interest, and
(v) a set of (interpretable) testing measures to evaluate transferability of testing results on synthetic data to corresponding real data.

We initially apply testing measures to validate that the model behavior on synthesized data obtained from the generative model is indicative of its behavior on real data, thereby

[2]We use the expression "target appearance" to describe the characteristics of the real-world data in distinction to its synthetic counterpart.

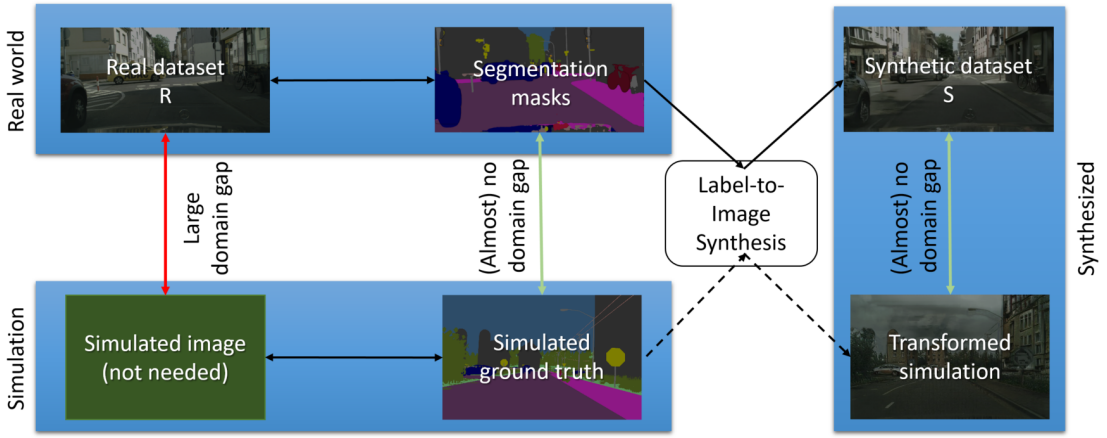


Figure 3.1: Label-to-image-based synthesis: Generation of the synthetic counterpart to the real set (upper part) and transformation of the simulated ground truth masks (lower part).

exploring the domain gap. If our testing measures yield a robust correlation of results between synthetic and real data, this substantiates the transferability of results and reliability of testing[3]. In this case, testing on simulated scenes of interest, processed by the generative model, can be expected to yield valid results. The intended validation of transferability (described in more detail in Section 3.1.3) requires a specific data generation procedure (described in Section 3.1.3).

Data Generation

First, we describe how to apply the generative model to produce pairs of real and synthesized elements to validate transferability. Secondly, assuming the transferability of results from synthetic to real data is valid, we detail the extension of the input data coverage by generating additional data with our controllable simulation engine involving the same generative transfer procedure. Here, we have the following underlying assumption (shown in Figure 3.1): Focusing on the ground truths as input for the generative model implicitly makes use of the advantage that the per-element domain gap between the ground truth's of simulated and real data is smaller than between their respective input data. Thus, the synthesized elements resulting from simulation masks and those resulting from real masks exhibit a smaller domain shift enabling argumentation of transferability. This way, we shift the questions concerning the relevance of the domain gap from the simulation to a controllable comparison.

Real-Synthetic Paired Dataset

We first need a paired dataset to validate transferability. Let R be our initial, labeled dataset extracted directly from the real-world data that features the target appearance. We generate a corresponding synthetic dataset S in the following way: For each element $r_i \in R$, we synthesize an equivalent element $s_i \in S$, whose label is the same as r_i, by

[3]Note that w.r.t. testing this can only be seen as a qualitative indicator of transferability.

applying the generative model to the segmentation label of the real element with the intention of mimicking the target appearance of R. The procedure is depicted in the upper part of Figure 3.1. In case the labeled elements from R do not suffice to evaluate transferability and additional unlabeled elements featuring the target appearance are available, R can be extended with pseudo-labels and S with the respective synthesized elements.

Extending Input Coverage: Transformed Simulated Dataset

Corner cases and rare events do not appear frequently enough in many datasets directly extracted from the real world to allow for extensive testing. However, many use cases including safety-critical ones require sufficient test coverage and a controllable framework can help to extend the testing. For this purpose, the simulation engine synthesizes labels containing cases of interest[4]. We then apply the same generative model as in Section 3.1.3 so that the synthesized data has the same style as R, see the lower part of Figure 3.1. Note that here the labels have to be in the same format as used for the paired dataset.

Validation of Transferability

Correlation and Performance Analysis

To assess transferability, we measure the performance of the model on all elements r_i of the real dataset R and on all corresponding elements s_i of the synthesized, paired synthetic set S (see Section 3.1.3). We obtain two sequences $(\text{perf}_{r_i})_{i=1}^{|R|}$ and $(\text{perf}_{s_i})_{i=1}^{|R|}$ of performance scores of same length. As a first step, we compute the sample correlation coefficient of $(\text{perf}_{r_i})_{i=1}^{|R|}$ and $(\text{perf}_{s_i})_{i=1}^{|R|}$. The higher the correlation coefficient, the stronger the evidence for transferability of overall qualitative model behavior from the synthetic to the real-world dataset. A high correlation coefficient on the paired set and the fact that the transformed extended input scenes have (almost) no domain gap w.r.t. the synthetic paired set justifies further testing with the transformed simulated dataset (see Section 3.1.3). In a second step, we assess the model performance on it using the same performance measure, perf, as in step one. We consider the aggregation of the per-image scores into one global score as a first proxy for potential performance on corresponding real data.

However, prediction errors can cancel out in case of non-binary performance scores, like mean-intersection-over-union (mIoU) , or add up to the same global score even if they are of different nature (e.g. in different regions of an image to be segmented). Hence, global model performance measures, such as the mentioned aggregated score, cannot reflect the complete model behavior to inspect our main questions. Therefore, we need a more differentiated analysis considering error distributions.

Error Distribution Analysis

So far, the approach was application-agnostic. For simplicity of notation, we here assume a semantic segmentation task. With slight modifications, the below approach can be adjusted to, e.g. other multi-class classification-related tasks. We first introduce some notation to formalize our approach for the error distribution analysis. Let the se-

[4]Note that for this procedure, we don't need the actual simulated elements but only their labels.

mantic segmentation problem have $C \in N$ classes, let $o(x_i)$ be the prediction mask of the segmentation network to the input $x_i \in \{r_i, s_i\},\ r_i \in R,\ s_i \in S$, and denote by y_i the corresponding segmentation ground truth mask (the mask is the same for both paired elements). Further, let $Y_c(i) \coloneqq \{y_i = c\}$ be the subset of y_i that has class c, for $c \in \{1, \dots C\}$ (i.e., pixels of an image that belong to that class). Let $O_{c,k}(i) \coloneqq \{e \in Y_c(i) : o(x_i) = k\}$ be the subset of $Y_c(i)$ in which the model outputs class $k \in \{1, \dots C\}$. Now, in a second part of our transferability assessment, we analyze the error distribution on the real dataset R compared to its corresponding synthesized set S. We construct a confusion matrix per class of interest. That is, per fixed class $c \in \{1, \dots, C\}$ and element x_i, we save the true positives as well as false negatives - distinguishing w.r.t. all other classes - and normalize these values so that the resulting values add up to one: $\text{TP}_\text{c}(i) = \frac{|O_{c,c}(i)|}{|Y_c(i)|}$ and $\text{FN}_\text{c,k}(i) = \frac{|O_{c,k}(i)|}{|Y_c(i)|}$ for $k \neq c$. Finally, we average over all elements of the respective dataset, resulting in one relative mean true positive score $\text{TPS}_\text{c} \coloneqq \frac{1}{|R|} \sum_{i=1}^{|R|} \text{TP}_\text{c}(i)$ and $C - 1$ false negative scores $\text{FNS}_\text{c,k} \coloneqq \frac{1}{|R|} \sum_{i=1}^{|R|} \text{FN}_\text{c,k}(i)$ for $k \neq c$ w.r.t. the ground truth class of choice c. Note that a TPS_c score of one is ideal. This procedure is repeated for all classes c. The resulting quantities TPS_c and $\text{FNS}_\text{c,k}$, $c \in \{1, \dots, C\}$, now are compared per class across the real and synthetic datasets resulting in a detailed analysis about whether the same misclassifications are made. This in turn provides evidence as to how far qualitative mistakes and semantic failures identified in one of the datasets also constitute errors on the other dataset. Comparing these findings from the paired set with the error distributions on the transformed simulated dataset provides an additional plausibility check. It aims to substantiate that testing on the transformed simulated scenes is justified and a corresponding real scene would lead to comparable model behavior. However, since the extended set might contain intentionally challenging elements, the error distribution may deviate. This points to semantic concepts that could lead to failure modes in corresponding real data.

We propose to visualize the findings with radar plots, one for each class c, since they allow for a systematic visual comparison and readable overview of errors on the different datasets. The axes in the plots correspond to the classes and to guide the eye lines are connected, the values being TPS_c for the ground truth class c and $\text{FNS}_\text{c,k}$ for the other axes $k \neq c$, see Figure 3.7 for an example. In addition, we propose boxplots of the distributions of the errors across the elements of the datasets, where each boxplot has the perspective of one ground truth class and shows the distributions of the $\text{TP}_\text{c}(i)$ and $\text{FN}_\text{c,k}(i)$ scores w.r.t. c.

Discriminating Model Outputs and Errors

While it provides a more comprehensive analysis than solely comparing aggregated performance scores, our error distribution analysis still lacks detail about where exactly these errors happen: we cannot assess whether the model behavior (especially regarding the kinds of committed errors as e.g. errors in different regions of a segmentation prediction) on the synthetic dataset is indistinguishable from the behavior on the real-world dataset. By training a discriminator to distinguish between model output on real data and synthesized data (or model errors, respectively) we target exactly this question. The underlying assumption is that if a discriminator cannot distinguish between model outputs/errors to real and synthetic input, they are sufficiently close in the sense

that we can consider the model behavior to be "the same" (up to discriminator precision) and, thus, synthetic testing is realistic. By choosing an interpretable discriminator setup, we can enhance the interpretability of the differences in model behavior on the real and synthetic sets.

3.1.4 Experimental Setting

We validate our approach described in Section 3.1.3 for an AD scenario by assessing transferability of tests on simulation videos. Our setup consists of the following components:

- The HRNetV2-W48 neural network (J. Wang et al., 2020)[5] for semantic segmentation with 19 classes trained on the Cityscapes training set (Cordts et al., 2016). We use without further modification the pretrained version available for download[6],
- the Cityscapes high resolution (2048 x 1024) validation set serving as the testing dataset,
- the generative adversarial video-to-video synthesis tool vid2vid (T.-C. Wang et al., 2018)[7], in particular, without further modifications the variant pretrained on Cityscapes,
- the controllable CARLA simulation engine (Dosovitskiy et al., 2017), and
- as the testing measures, we employ the score mIoU as the performance measure for the correlation analysis, an error distribution analysis for each of the 19 Cityscapes classes and SkopeRules as rule learner based on the feature engineering from MetaSeg (M. Rottmann et al., 2020)[8] as discriminator of model outputs and errors.

The technical implementation is detailed further in the following subsections.

Generating Synthetic data

The video-to-video synthesis tool vid2vid generates a video sequence using both a sequence of instance and pixel-level semantic segmentation masks. Following our approach described in Section 3.1.3, we generated two different datasets via vid2vid:

Real-Synthetic Paired Dataset: Cityscapes - Synthetic Cityscapes

We transform the 500 Cityscapes validation segmentation masks (corresponding to three different video sequences) with fine annotations (containing 30 classes) to three synthetic video sequences via vid2vid in Cityscapes style. We call the resulting video sequences with the corresponding labels *synthetic Cityscapes A*, in contrast to the original Cityscapes video sequences and labels, which we call *real Cityscapes A*. Note that both datasets share the same segmentation labels and differ only in the corresponding image frames. Vid2vid requires the labeled images to be time-consistent to generate high-

[5] https://github.com/HRNet/HRNet-Semantic-Segmentation/tree/pytorch-v1.1

[6] https://onedrive.live.com/?authkey=%21AErsWO7%2DxcLEVS0&cid=F7FD0B7F26543CEB&id=F7FD0B7F26543CEB%21169&parId=F7FD0B7F26543CEB%21166&action=locate

[7] https://github.com/NVIDIA/vid2vid

[8] We use an internal code base provided by the authors of MetaSeg (M. Rottmann et al., 2020), which is an extension to their existing repository under https://github.com/mrottmann/MetaSeg

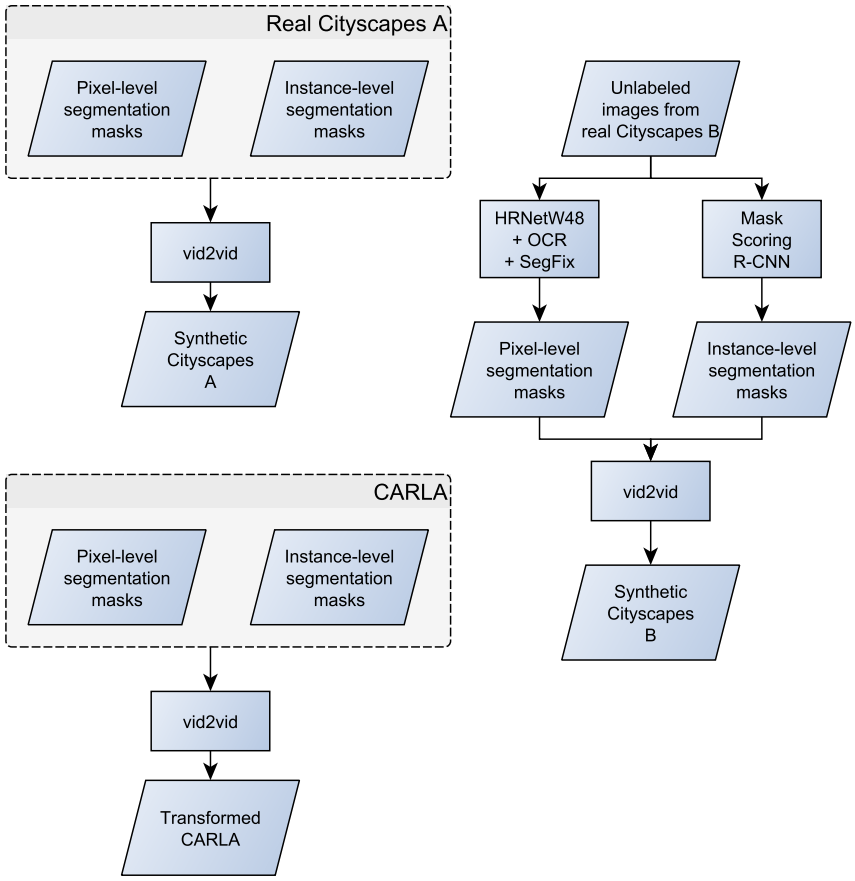


Figure 3.2: Depiction of the performed synthetic generation processes with the involved datasets.

quality sequences – a condition that the aforementioned dataset does not completely fulfill due to its frequent temporal discontinuities. Thus we follow the suggestion of T.-C. Wang et al., 2018 and generate pseudo-labels for the unlabeled Cityscapes validation images, which are more time-consistent, with HRNetW48 + OCR + SegFix (Yuan and Jingdong Wang, 2018; L. Huang et al., 2019; Yuan, X. Chen, and Jingdong Wang, 2020; Yuan, J. Xie, et al., 2020)[9] (we use without modifications the variant pretrained on Cityscapes) and Mask Scoring R-CNN for instance segmentation (He et al., 2017) (pretrained on MS COCO (T.-Y. Lin et al., 2014) and further fine-tuned on a proprietary dataset) that are also synthesized with the pretrained vid2vid model[10]. This results in 15,000 new images that, together with the pseudo-labels, we call *synthetic Cityscapes B* in contrast to the pseudo-labels and original video sequences, which we call *real Cityscapes B*[11]. The schematic processing of datasets is depicted in Figure 3.2. The transformation

[9] https://github.com/openseg-group/openseg.pytorch
[10] This setup leads to an improved segmentation compared to the HRNetW-48 under test.
[11] Note that the real Cityscapes A images are contained in real Cityscapes B. However, for consistency,

(**a**) Share of Cityscapes B (**b**) Pseudo-label

(**c**) Paired synthetic Cityscapes B image

Figure 3.3: Example from the paired dataset generation. We pseudo-label an unlabeled Cityscapes image (a) with HRNetW48 + OCR + SegFix, resulting in a segmentation mask (b), and vid2vid transforms it to a paired synthetic image (c).

process from an unlabeled Cityscapes image (from real Cityscapes B) to its synthetic counterpart (synthetic Cityscapes B) is displayed in the example in Figure 3.3. Notice the quality of the generated image despite the pseudo-labels.

Extending Input Coverage: Transforming CARLA to Cityscapes Style

We generate 100 random video scenes (i.e., random city, weather, lighting, etc.) of about 140 images per sequence with a proprietary variant of the CARLA simulator (Dosovitskiy et al., 2017) together with labels, calling this *original CARLA* dataset. The corresponding segmentation masks are mapped to the 30 Cityscapes classes and processed by vid2vid. This, together with the labels, now constitutes the *transformed CARLA* dataset. For the schematic process see Figure 3.2. An example image from transformed CARLA is displayed in Figure 3.4.

Validation of transferability

Correlation and Performance Analysis

We conduct the performance analysis on the paired set, synthetic and real Cityscapes A, described in Section 3.1.4 using mIoU as the performance metric. We use the reduced set of 19 Cityscapes classes and distinguish between them to gather IoUs per class as well as mIoU scores per image. We average these quantities over all images of the respective dataset and compute the sample correlation coefficients as described in Section 3.1.3.

we use the pseudo-labels as ground truth here.

Figure 3.4: Example of a vid2vid-generated image from a CARLA segmentation mask. Notice how the artifact on the right almost has stop sign shape.

We repeat this procedure for paired Cityscapes B. The obtained results can be seen in Figure 3.5 and Figure 3.6. Figure 3.5 depicts class-wise IoU and class-wise correlation coefficients on both paired sets, in which one can observe a relatively high class-wise correlation coefficient when the network performs well (i.e., it has a rather good IoU for that class), e.g. for the classes road, building and vegetation as well as the particularly safety-relevant classes person and car. Note that despite a high IoU for both synthetic and real sets for the class sky, the correlation coefficient is relatively low.

Plotting the mIoU on paired real and synthetic Cityscapes A in the upper part of Figure 3.6 shows how they correlate (the sample correlation coefficient is approximately 0.403). The lower part of Figure 3.6 shows an analogous plot for the B sets, in which the sample correlation coefficient is slightly higher with 0.457. The displayed peaks and dips of the image-wise mIoU scores on the synthetic set matching those on the real set qualitatively suggest a rather good transferability. From a testing perspective, especially the negative correspondence, i.e., low performance on synthetic data coinciding with lower real data performance, are important as they might help reveal failure modes. Evaluation of the function on the extended scenes of transformed CARLA yields a mIoU of 0.196, constituting a decrease of around 0.07 compared to synthetic Cityscapes A. However, the performance on the transformed CARLA seems to depend on the choice of the particular sequence of videos. This might provide a first insight that semantic concepts in these sequences might constitute failure modes.

Error Distribution Analysis

Conducting the error distribution analysis described in Section 3.1.3, we find that $FNS_{c,k}$ values differ for the datasets, i.e., the error distribution across the datasets differs, providing evidence that errors are not always transferable (in this work we only show one example radar plot, omitting the rest). This effect is in particular visible for transformed CARLA errors. On paired Cityscapes B, we see that errors are rather comparable, especially for the larger classes such as road, sidewalk, building, wall, and sky. We observe that the better quality of the real B set, relative to the requirements of vid2vid,

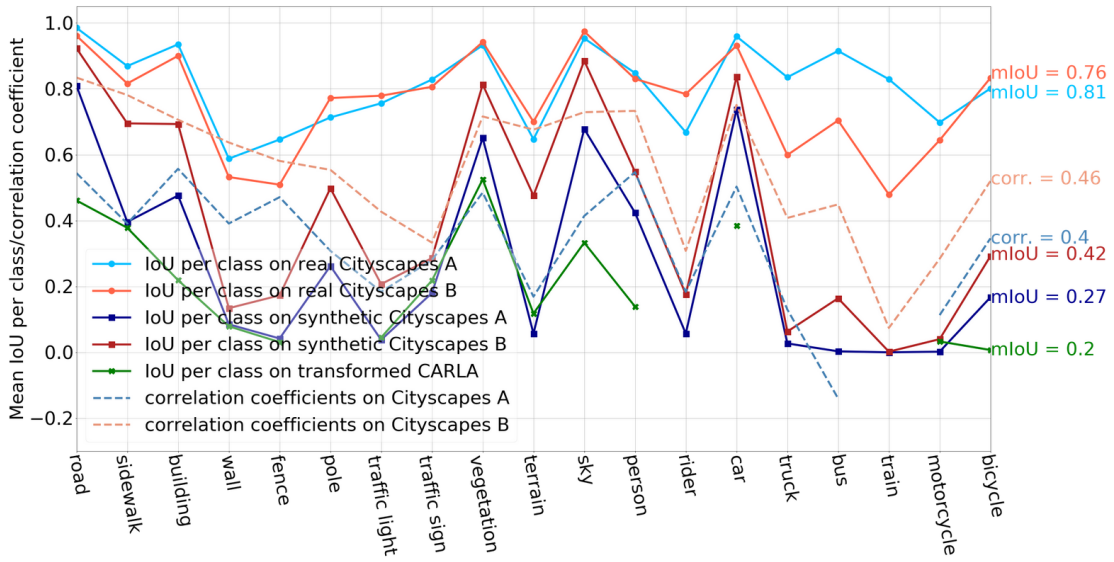


Figure 3.5: Correlation and performance analysis results on paired Cityscapes A and B as well as transformed CARLA classwise IoUs. Note that there is no correlation coefficient for class 'train' on Cityscapes A since there are not enough samples from the synthetic and real sets. Classes in CARLA that have no IoU value are not available from simulation.

enhances the comparability of errors. However, the outcome is still class-dependent, hinting at the fact that the current simulated data might only be suitable for testing w.r.t. some particular classes. We observe that errors tend to fall into naturally adjacent classes. The example radar plot in Figure 3.7 shows the ground truth class sidewalk getting mistaken for road and building. By definition our $\mathrm{FNS}_{c,k}$ measures the (relative) amount of wrongly segmented pixel area. Similar to IoU, we expect $\mathrm{FNS}_{c,k}$ to be more fluctuating for objects of smaller area, which is apparent in the larger stability for aforementioned classes with typically large pixel areas. Lastly, we expect the measure to be correlated to the IoU itself, as they are related quantities: To be precise, the smaller the IoU of an object, the larger it contributes to $\mathrm{FNS}_{c,k}$.

Discriminator on Model Outputs and Errors

We train the SkopeRules rule learner based on feature engineering of MetaSeg (M. Rottmann et al., 2020) to distinguish whether a model output/error belongs to a real or synthetic input image. More precisely, using MetaSeg, we compute various quantities from the predicted pixel-wise class probabilities, which are aggregated per connected component (segment) of the model's output segmentation mask. These quantities include dispersion measures, i.e., the pixel-wise entropy, probability margin, and variation ratio. These measures get aggregated over each whole predicted segment as well as only the corresponding boundary and the inner. The aggregation is performed by considering both mean and variance over all pixel-values that correspond to the whole segment, the inner or the boundary, respectively. Furthermore, we consider the size

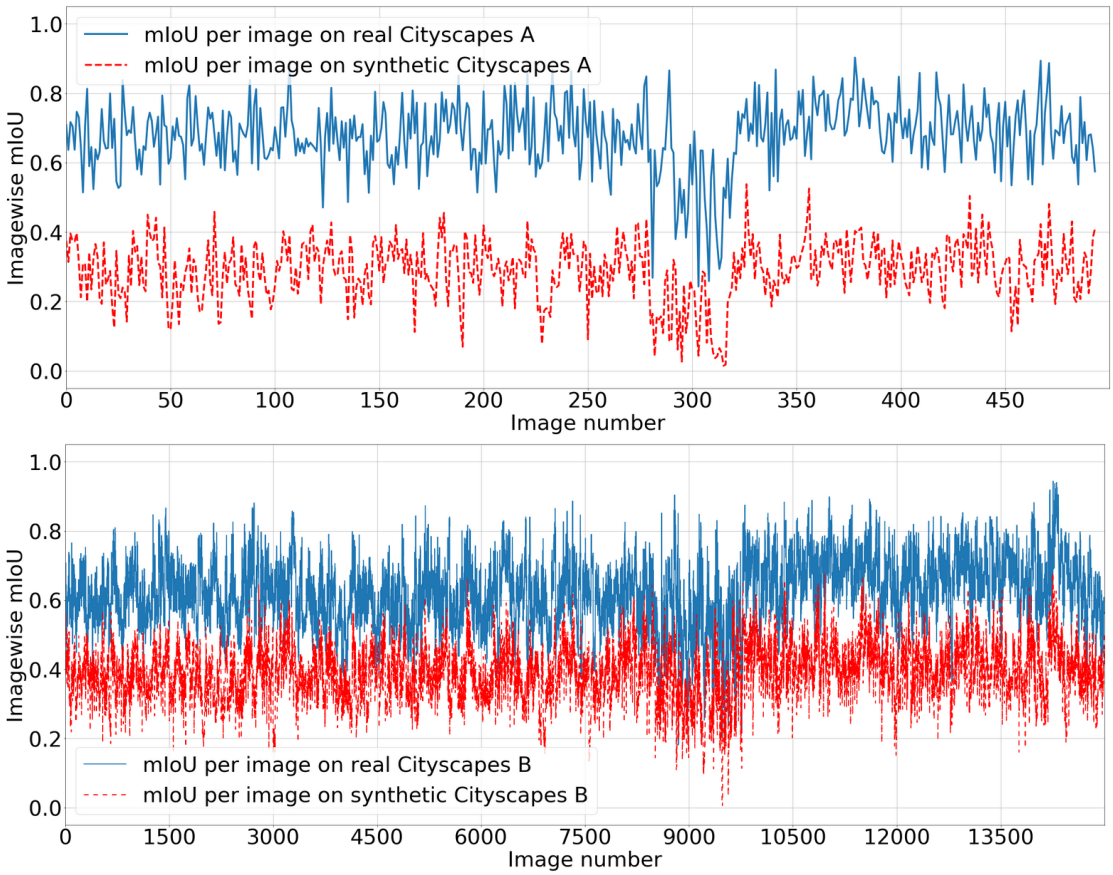


Figure 3.6: Correlation of mIoU per image on paired Cityscapes A (top) and on paired Cityscapes B (bottom).

measures, i.e., the size of the whole segment, its boundary and its inner as well as fractality measures like the segment size over the boundary size. Altogether we obtain a structured dataset that contains a number of interpretable scalar values per predicted segment. For further details, we refer to M. Rottmann et al., 2020. Moreover, MetaSeg stores labels and IoUs for each segment. This diversity of computed metrics allows for a distinct uncertainty assessment for the predicted segments, enabling a geometric interpretation of the sets of rules. For instance, a rule for some class c including boundary entropy as a classifier for real vs. synthetic input implies that differences between the sets lie on the boundaries of the respective class segments.

In total, we compute 35 different metrics per segment of the prediction mask of the HRNet for each image of paired Cityscapes A and B, respectively, and save information about the belonging dataset. We then separate the dataset according to the 19 semantic classes of Cityscapes (classwise choosing the minority dataset - real or synthetic - as the target to learn rules for) and perform a random $80:20$ train-test split. In a first step, we learn rules on all classwise segments before we filter (again classwise) for errors, i.e., for segments with IoU $= 0$. The same analysis is performed on a subset (of the same size as Cityscapes A) of images from paired Cityscapes B. The accuracy scores of the

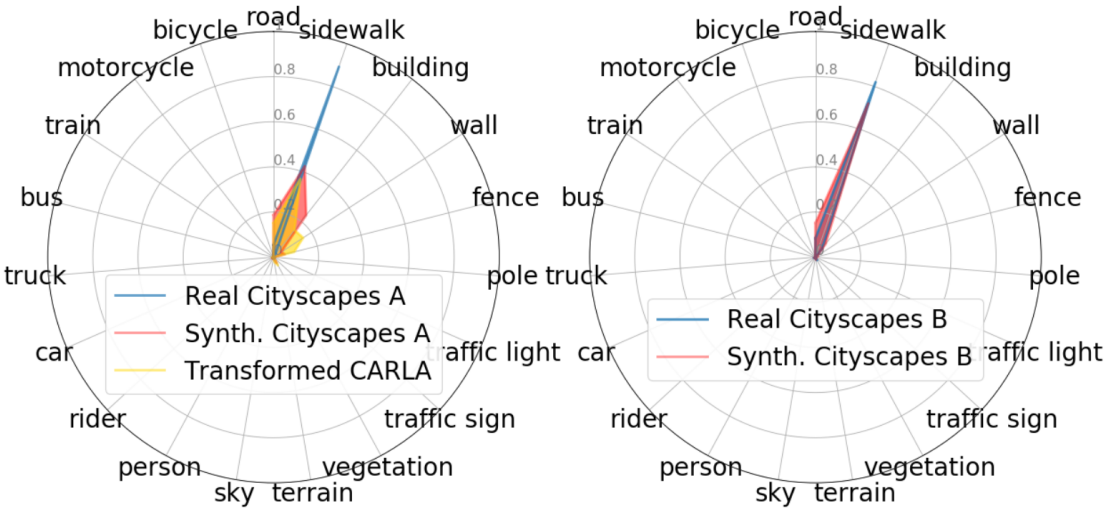


Figure 3.7: Error distribution analysis: Radar plots w.r.t. class sidewalk for paired Cityscapes A and transformed CARLA (left) as well as on paired Cityscapes B (right).

resulting rule sets can be seen in Figure 3.8.[12] Overall, the accuracy scores are rather high (see Figure 3.8a) – our discriminator can (easily) tell the inputs apart – implying that the model behavior differs on real and synthetic paired input data. Additionally, on both Cityscapes A and B, the restriction to only error components leads to higher mean accuracy of the set of rules, i.e., a better distinguishability of real and synthetic input data. However, as seen in Figure 3.8a, the accuracy scores on Cityscapes B are lower on average, indicating that better quality of synthetic data makes it more difficult to differentiate between real and synthetic inputs. We observe that the rule accuracy scores do not reflect any IoU performance gap, as we see in Figure 3.8b and Figure 3.8c. This underlines our claim that performance metrics such as (m)IoU (as described in Subsecion 3.1.3) alone cannot assess the comparability of model behavior. Interestingly, concerning hyperparameters, a maximal depth of 1 turns out to be optimal for the rule sets across all classes in all our discriminator experiments. Also, the rule sets for model outputs on paired Cityscapes A contain mostly boundary metrics, whereas the differences between the datasets are rather scattered and thus more difficult to interpret for the remaining experiments. This might be due to the increased difficulty of synthesizing boundary pixels via vid2vid.

Finally, we can say that our proposed methods and metric correlate well with the visually perceptible quality difference in the synthetic set: On the more realistic looking synthetic Cityscapes B, rule accuracy scores drop. Nevertheless, the results of the discriminator analysis show that the proposed metrics provide a good way to identify the domain shift present between real and synthesized datasets. Note, however, that for practical testing purposes it may not be necessary to fully close this gap.

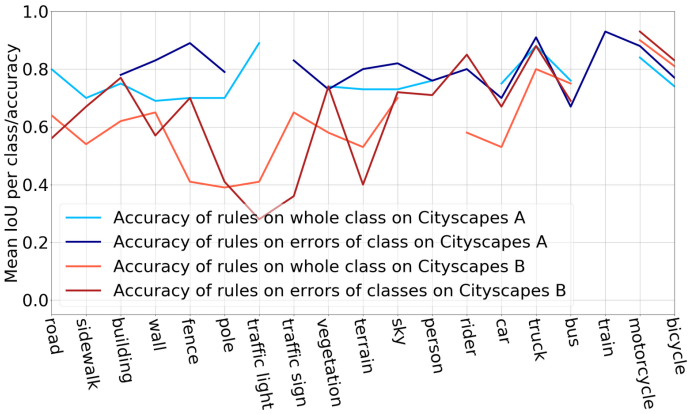
[12] We used the (top k) rules optimized for the minority class, thus sometimes degrading overall accuracy.

3.1.5 Conclusion and Discussion

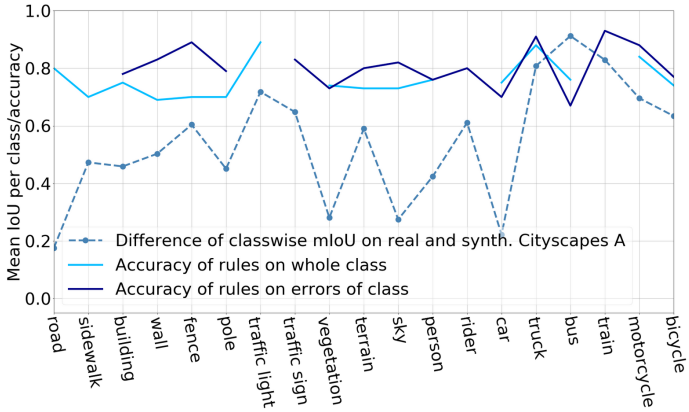
We presented a conceptual framework to validate simulation-based testing of real-world ML applications, which we instantiated on a semantic segmentation task in the context of AD. As simulation and real-world data have a domain gap, our work explicitly addresses the question of *transferability* of testing results. We employ a generative label-to-image transfer method, mapping from the ground truth labels back to the (real world) image domain, which provides two key advantages: First, we can map a given labeled (real) dataset onto a synthesized version of itself allowing us to directly and in detail investigate the resulting domain gap incurred by the generative method. Second, applying the same generative model to ground truth data from any source, e.g. a simulator, we can test the ML application. Under the condition that the simulated ground truth is of the same form as the real world one, we have almost no domain gap between the synthesized real data and the data synthesized from a simulation. So, using this two-stage approach we can largely bypass the question of domain gap regarding simulated test data, and instead shift it to a more controllable comparison between two datasets with identical ground truths.

While the performance of the generator is not crucial, it is clear that our approach still benefits from a small domain gap between the actual and the synthesized data. Improvements regarding better generative models, more available data, and additional validation metrics can easily be incorporated into our modular framework. With semantically rich enough labels to facilitate data generation, our approach could be used on other tasks such as e.g., object detection, using mean average precision as performance score, and on other application domains as, e.g. in the context of text mining.

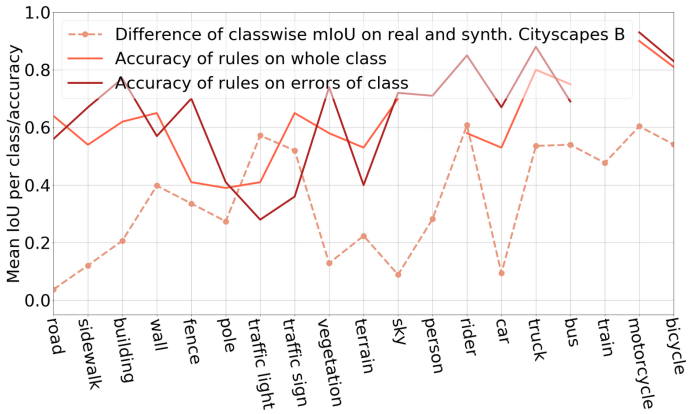
Turning, at last, to the concrete instantiating of the framework on the segmentation task, we evaluated transferability calculating class-wise mIoU correlation coefficients and found for cars or person surprisingly strong and encouraging values of 0.7. A deeper analysis of failure modes based on manual feature extraction, however, revealed that failures can be still clearly classified as belonging to the real data or its synthesized counterpart. Lastly, while we demonstrated the feasibility of the approach the actual test of the segmentation model, e.g., active weak-spot search, is left for future work.



(**a**) Classwise accuracy scores of rule-sets on model outputs and errors on paired Cityscapes A and B.



(**b**) Classwise rule-set accuracy scores together with differences in IoU performance on paired Cityscapes A.



(**c**) Classwise rule-set accuracy scores together with differences in IoU performance on paired Cityscapes B.

Figure 3.8: Results from rule learning on model outputs and errors on paired Cityscapes A and B. Note that for some classes no rules could be found or that there were not enough components to learn rules, leading to lacking data points in the plots.

3.2 Evaluating German Word Embeddings on a Sentiment Analysis Use Case

This section is an adaptation of the work presented by Brito, Sifa, Cvejoski, et al., 2017. The first author was principally responsible for the conception and execution of the experiments presented, as well as the drafting of the text.

3.2.1 Introduction

Although vector representations for text are nowadays ubiquitous across NLU applications, the majority of research still focuses on the English language. The generalization of proposed models for non-English languages such as German, or even for domain-specific English corpora, is not always clear. Furthermore, the selection of hyperparameters, a crucial factor for high-quality representations, is often under-documented. This issue was especially noticeable at the time of publication for Brito, Sifa, Cvejoski, et al., 2017 was published, when continuous distributed representations for words (the so-called *word embeddings*) were a hot topic among NLU researchers.

In the initial part of our work, we train German embeddings from SdeWaC, a large German corpus created by web-crawling the ".de" domain (Faaß and Eckart, 2013; Baroni et al., 2009). We train 11 word vector models with varying hyperparameter combinations and evaluate them on a similarity task to assess whether the general recommendations for English word embeddings also apply to German.

Using the word embeddings generated in the previous task as a foundation, we map Google Play reviews to the same vector space using the Paragraph Vector model (Le and Mikolov, 2014). We then use these new vectors to predict whether a user liked an app based on a given review using three different algorithms: logistic regression, decision trees, and random forests. Even though no correlation was found between our quality measures at a word representation level (the results of the word similarity evaluation) and at a prediction level (geometric mean of accuracy), our best word embedding models can still yield satisfactory prediction models.

This work aims to clarify how different hyperparameter combinations impact the resulting word embeddings, and how these representations can be part of more complex predictive models. The latter does not only serve as an extrinsic evaluation of the German word embeddings but also shows the feasibility of predicting preferences only from document embeddings, allowing to analyze inherently rich text corpora for Business Intelligence. Using this approach, stakeholders at any company can build predictive models over the text base that is augmented by social media content to have more insights about their customers. Namely, combined with the methods from the mature field of predictive analytics, having numerical data representations for chunks of customer text will allow us to analyze trends, implicit and explicit user feedback and future interest in company products. For that, we present a case study to predict implicit user feedback of German Google Play reviews. As a whole, this work must be understood as an initial feasibility study about performing sentiment analysis on informal and noisy German texts by means of word representations. Further optimization of our best models is left for future work.

3.2.2 Data

We use two different datasets to infer German embeddings: SdeWaC (Faaß and Eckart, 2013) for rather generic word embeddings and SCARE (Sänger et al., 2016) to produce document embeddings tailored for sentiment analysis. For evaluating the representation quality of the word embeddings, we use the Gur350 dataset (Gurevych, 2005; Zesch and Gurevych, 2006).

SdeWaC

The SdeWaC corpus is a subset of the deWaC corpus, a web-crawled collection of German texts extracted from the ".de" domain in the scope of the WaCky project (Baroni et al., 2009). It consists of 846.159.403 word tokens in 44.084.442 sentences, including 1.094.902 different word types. Thanks to its variety and size, this corpus is suitable to generate general purpose embeddings to model the German language. We apply a minimal pre-processing to this corpus before it is processed: it is only tokenized, lowercased and shuffled, taking a sentence as a text unit.

SCARE

The Sentiment Corpus of App Reviews (SCARE) consists of 802,860 German app reviews collected from Google Play Store. These reviews are divided in 11 different app categories: instant messengers, fitness trackers, social network platforms, games, news applications, alarm clocks, navigation and map applications, office tools, weather apps, sport news and music players. From each category, between 10 and 15 different apps are considered. All these reviews contain a text and a rating with a 1-5 star score. Both are used for our sentiment analysis task.

Gur350

The Gur350 dataset contains 350 pairs of German words with a human-annotated semantic relatedness score which was originally introduced to measure distances between words (Gurevych, 2005; Zesch and Gurevych, 2006).

3.2.3 Experiments

Word Embedding Benchmarking

We make use of the `word2vec` package (Mikolov, Sutskever, et al., 2013) to obtain vector representations of German words. It learns word embeddings with mini-batch asynchronous stochastic gradient descent on a shallow neural network, which can have two possible architectures: continuous Skip-gram (SG) architecture or continuous bag-of-words (CBOW) (Mikolov, K. Chen, et al., 2013). With the purpose to check the effect of SG and CBOW hyperparameters that need to be specified in `word2vec`, we train 11 different models on the SdeWaC corpus. A description of the hyperparameters is presented in Table 3.1 and the tested values for them are shown in Table 3.2.

Table 3.1: word2vec hyperparameters

Hyperparameter	Meaning
`cbow`	Network architecture: 0 for SG, 1 for CBOW
`window`	Maximum skip length between words within a context window (maximum window size)
`sample`	Threshold for word subsampling
`hs`	Hierarchical Softmax
`negative`	Number of negative samples
`min-count`	Word types below this count value are discarded

Table 3.2: Hyperparameters used for the 11 trained word vector models

Model	**cbow**	**window**	**sample**	**hs**	**negative**	**min-count**
0	0	8	0	1	10	50
1	1	8	0	1	10	50
2	0	5	0	1	10	50
3	0	15	0	1	10	50
4	0	8	1e-5	1	10	50
5	0	8	1e-3	1	10	50
6	0	8	0	0	10	50
7	0	8	0	1	0	50
8	0	8	0	1	20	50
9	0	8	0	1	10	10
10	0	8	0	1	10	100

All models were set to generate vectors of size 300, which is a frequent value among publications about word embeddings. Since the usually recommended ranges for the hyperpameters may not apply for the German language (most of the available publications focus on the English language), all models from 1 to 10 differ only in one hyperparameter compared to model 0 so that the effect of each hyperparameter can be assessed[13].

The resulting embeddings are evaluated on a similarity task. The cosine distance is our similarity measure between our word vectors. By calculating the Spearman's rank correlation coefficient (Spearman, 1904) between the cosine distances of word vector pairs and the semantic relatedness from the Gur350 dataset, we can evaluate the quality of our word embeddings.

From Table 3.3 we can observe that the model with best performance (model 4) on the similarity evaluation corresponds to the one subsampling the most frequent words with highest threshold (1e-3). Considering that the other model applying subsampling (1e-5) obtained the third best Pearson's correlation, the recommendation of introducing subsampling during the training phase seems to be also valid for German embeddings.

[13]Model 0 corresponds to the only previous work that we could find about German word embeddings in which hyperparameter settings are explicitly specified: https://github.com/devmount/GermanWordEmbeddings

Table 3.3: Spearman's ρ between the trained word embeddings and Gur350 word pairs human-annotated semantic relatedness

Model	ρ
0	0.7153
1	0.5510
2	0.6933
3	0.7325
4	0.7479
5	0.7335
6	0.7399
7	0.7002
8	0.7103
9	0.7222
10	0.7249

Table 3.4: Geometric Mean of Negative and Positive Class Accuracy Values of Classifying of Liking of Fitness Tracker Applications

Model	LR	DT	RF1	RF2
0	0.791	0.677	0.729	0.696
1	0.766	0.650	0.661	0.645
2	0.789	0.685	0.730	0.695
3	0.788	0.679	0.725	0.686
4	0.784	0.689	0.729	0.691
5	0.791	0.674	0.727	0.697
6	0.788	0.685	0.719	0.693
7	0.790	0.684	0.734	0.699
8	0.783	0.680	0.726	0.693
9	0.799	0.679	0.732	0.693
10	0.789	0.668	0.724	0.693

Besides that, we see that the only model applying the CBOW model (model 1) is clearly worse than the rest. Although one observation does not suffice to claim that SG outperforms CBOW, our result is in line with previous observations that the CBOW model cannot produce better word embeddings than SG in spite of being a more expressive model (Levy, Goldberg, and Dagan, 2015). Regarding the window size, we confirmed our expectations that larger window sizes lead to better results. Therefore, model 3 (window size 15) improves model 0 (window size 8) whereas model 2 (window size 5) drops the Spearman's correlation.

Predictive Sentiment Analysis

We evaluate our word representations with a Business Intelligence use case which involves predicting user preferences from a given text document. This does not only help

us to gain insight about the hyperparameter space for the learning process of word embeddings, but also shows that we can obtain high accuracy values for predicting user decisions by only using paragraph embeddings as data input. We train document representations for German Google Play Reviews from (Sänger et al., 2016) for the fitness trackers category. The dataset contains 22,188 anonymized reviews with ranking ranging from one to five. We binarized (or implicitized) the review score by assigning *False* class to ratings one, two and three and *True* class to ratings four and five to respectively model the notion of user's not liking and liking of the product. After this pre-processing step the resulting dataset contained a 26/74 class distribution ratio.

Similar to (Le and Mikolov, 2014), we learn document representations of the reviews by means of the *Paragraph Vector with Distributed Bag of Words* model (PV-DBOW) (Le and Mikolov, 2014) implemented in gensim `doc2vec`[14]. Since PV-DBOW does not explicitly learn word embeddings but only the paragraph vectors (document embeddings), the `doc2vec` implementation initializes the word vectors randomly if not specified otherwise. Although the learning algorithm should be able to work with this setting, the performance degrades severely in practice (Lau and Baldwin, 2016). Therefore, we initialize our model with the pre-trained word vectors trained with the large external corpus SdeWaC that we obtained from our experiment explained in Section 3.2.3. Then, we train 300-dimensional document vectors during 1000 epochs using PV-DBOW with a maximum window size of 15 and 5 negative samples per context window. We also subsample the most frequent words setting the threshold to 1e-5. By keeping this hyperparameter combination fixed, we can assess which of the pre-trained word embedding models lead to a better performance for our predictive sentiment analysis task.

For the supervised learning task of sentiment analysis, we used Logistic Regression (LR), Decision Trees (DT) and Random Forests with 101 and 11 random trees (RF1 and RF2 respectively). In Table 3.4 we show the evaluation of the predictions using 10-fold cross-validation in terms of the geometric mean of the accuracy values of both of the classes.

Overall, Logistic Regression yielded the best results for predicting the user's preferences with respect to the geometric mean accuracy. When we compare the experimental settings we observe that reducing the minimum count of words to be considered in the learning phase improves the representation with larger window sizes reaching almost up to 0.8 percent of the geometric mean for positive and negative classes.

It is also noticeable that the CBOW word embedding model has not performed well in terms of representation learning compared to all the SG models, which was already indicated by Levy, Goldberg, and Dagan, 2015 for the English language. However, the results on the sentiment analysis correlate poorly with the intrinsic evaluation using the Gur350 dataset. This fact is in line with recent research on evaluating word embeddings showing that most word similarity datasets available for word similarity evaluations are not useful to predict a good performance on an extrinsic task such as sentiment analysis (Faruqui et al., 2016). Our results suggest that Gur350 also suffers from this problem and it is thus not suitable to be used as an intrinsic evaluation dataset for German word embeddings.

[14] https://github.com/RaRe-Technologies/gensim

3.2.4 Conclusion and Future Work

We observed that some of the general recommendations to learn word embeddings in English (such as those from Levy, Goldberg, and Dagan, 2015) also apply for the German language when they are evaluated on a word similarity task: the continuous skip-gram model outperforms the continuous bag of word models and negative sampling provides better results than hierarchical softmax, where more negative samples improve the result. Furthermore, we detected that the threshold to downsample the most frequent words has the highest impact on the similarity score among all tested hyperparameters. Nonetheless, we also noticed that the models that perform the best on the similarity task do not necessarily provide the best contributions when they are used to infer document embeddings for the predictive sentiment analysis task. Like Faruqui et al., 2016 does for the English language, we suggest not to rely on similarity tasks to assess the quality of German word embeddings until a suitable evaluation dataset is available. We also showed the feasibility of performing sentiment analysis on informal German text using document embeddings as features of the classifiers.

As a future work, we will focus on optimizing the sentiment analysis approach and the document representations on concrete use cases. It would be also interesting to check if our findings generalize across other German text collections and with other text pre-processing approaches.

3.3 Hybrid Ensemble Predictor as Quality Metric for German Text Summarization: Fraunhofer IAIS at GermEval 2020 Task 3

This section is based on the work presented by Biesner et al., 2020, where the author of this dissertation served as a co-first author. As such, the author contributed to the conception and execution of the experimental framework, and drafting the results in the submitted manuscript.

3.3.1 Introduction

In our previous work on automatic text summarization (Brito, Lübbering, et al., 2019), we concluded criticizing the suitability of ROUGE scores (C.-Y. Lin, 2004) for overall evaluation purposes. These and other common quality metrics found in the automatic text summarization literature like BLEU (Papineni et al., 2002) or METEOR (Banerjee and Lavie, 2005) are far from being optimal since they only focus on the lexical overlap as a proxy for assessing content selection. They do not only penalize certain abstractions (e.g., when the original sentences are heavily reformulated or when synonyms are applied) but they also ignore other aspects that are usually considered desirable in good summaries, including grammatical correctness and compactness.

The second German Text Summarization Challenge aims to address this issue by releasing a text corpus with several summaries per text[15]. Its participants were asked to rate these summaries with new ideas and solutions regarding an automatic quality assessment of German text summarizations. We propose to combine the advantages of neural approaches that excel at encoding semantic textual similarity (and are thus suitable to predict content) with statistical and rule-based metrics that can evaluate other important summarization aspects such as compactness and abstractiveness.

In our approach, we employ an ensemble of 7 statistically significant predictors (p-value $< 15\%$) in a linear regression model (see Table 3.6). Comparing our predictions to the competition host's own non-public annotations we achieved a score (i.e. loss) of 33.72, one of the lowest and therefore best scores of participating teams.

In the following sections, we detail the different metrics that we considered and how we optimized its combination.

3.3.2 Experimental Setup

This section describes our experimental setup, namely the underlying dataset and the methodological approach.

Data

The shared task organizers released a corpus consisting of 216 texts with a corresponding reference summary and a generated summary, each of them rated with a value between 0 (bad) to 1 (excellent).

[15] https://swisstext-and-konvens-2020.org/2nd-german-text-summarization-challenge.

In order to evaluate the methods we manually annotated all summaries in the dataset with a score from 0 to 1. We independently rated a part of the corpus each, such that different human biases can be compensated to a certain extent. A submission of these annotations to the competition received a high score, indicating a large similarity to the gold standard annotations set by the organizers. Additionally, we expanded the dataset by considering the given reference summaries as perfect generated summaries with an automatic score of 1.

This results in a dataset of 248 summary texts with their corresponding score, which is used to evaluate the unsupervised methods described below.

Methodology

We address this challenge as a metric learning problem, where we define a set of unsupervised predictors covering one or several features that answer the required properties of a good summary (content relevancy, compactness, abstractiveness and grammatical correctness). After calculating all predictor scores (unsupervised) for each document we apply min-max normalization to assure all scores lay in the closed 0-1 interval. In a final step, we ensemble these predictors in a capped linear regression model (output between 0 and 1), which is trained via ordinary least squares on our manual summary annotations (see Section 3.3.2). We iteratively remove non-significant predictors, p-value $\geq 15\%$, and re-run the regression model until all predictors yield significant t-statistics, namely their coefficients lay within the two-sided 85% confidence interval. Due to the limited amount of documents and the loss of interpretability, we refrain from including non-linearities (e.g. multiple layers, non-linear activation functions, interaction terms of different polynomial degrees, etc.) into the regression model. Also, by using a simple linear ensemble model, we reduce the likelihood of overfitting on our annotations, especially since no validation set for parameter tuning is available.

The following subsections lay the focus on our predictors and describe their functionality. We start presenting three content predictors, which all determine the most important words in the original text and compute the fraction of how many of these words occur in the generated summaries. We assume that the most important words in a document capture the essence of the text and thus, function as proxy for contentual relevance. We continue with neural language model driven predictors which primarily focus on contentual relevance and grammatical correctness. We also include the standard quality metrics for automatic summary evaluation, ROUGE, BLEU, and METEOR, which all aim to measure contentual relevance, as well. The remaining predictors are mainly rule-based and refer largely to compactness and abstractiveness.

- **Tf-Idf content predictor** A very popular text vectorization method is tf–idf (Term frequency – Inverse document frequency). It is a frequency-based statistic, which intends to reflect how important a word is to a "document" in a corpus. Given that our entire corpus contains N documents and the vocabulary of our corpus is of size K, we can collect the individual tf–idf scores in some matrix $\mathbf{M} \in \mathbb{R}^{\mathbf{N} \times \mathbf{K}}$. Each row vector in this matrix corresponds to a document embedding. We find the top 10 important words per document by decreasingly sorting the tf-idf scores within

each embedding. We utilize the `sklearn` [16] implementation of the `TfidfVectorizer` and restrict our vocabulary to words with a document frequency below 0.9. Before vectorization, we apply lower-casing, punctuation and stop word removal, and stemming to the entire text corpus, which helps to better capture meaning and content in the text's vector representation.

- **NMF content predictor** NMF (Nonnegative Matrix Factorization) (Paatero and Tapper, 1994; D. D. Lee and Seung, 2001) is a common matrix factorization technique frequently used for topic modeling. In previous work, we find that NMF achieves good results in clustering document words to a predefined number of latent topics. Assuming that a good summary should cover all main topics in a text, we apply NMF on each document, and determine the top 5 important words per latent topic dimension. In particular, we factorize the document's symmetrical co-occurrence matrix[17] $\mathbf{S} \in \mathbb{R}^{\mathbf{N}\times\mathbf{N}}$ into a nonnegative loading matrix $\mathbf{W} \in \mathbb{R}^{\mathbf{N}\times\mathbf{M}}$ and a nonnegative affinity matrix $\mathbf{H} \in \mathbb{R}^{\mathbf{M}\times\mathbf{N}}$,

$$\mathbf{S} = \mathbf{WH} + \mathcal{E}, \tag{3.1}$$

  where N is the vocabulary size of the document at question, $M = 10$ is the number of latent topics and $\mathcal{E} \in \mathbb{R}^{\mathbf{N}\times\mathbf{N}}$ is the error matrix, whose elements approach zero for a perfect decomposition. For both, $\mathbf{W}$ and $\mathbf{H}^{\mathbf{T}}$ we assign each word (row vector) to the latent topic dimension with the highest value. Next, we decreasingly sort the assigned words per topic, so that the most distinct topic words are ranked on top. Finally, we get the important words per document by removing all duplicates from the selected topic words of $\mathbf{W}$ and $\mathbf{H}$.
- **Flair NER content predictor** Flair (Akbik, Blythe, and Vollgraf, 2018) is a specific contextual string embedding architecture. The backbone of the flair framework is a pretrained character-based language model (based on an LSTM[18]-RNN), which is bidirectionally trained on a huge independent text corpus for different languages, including German. Build on top of this language model, the framework provides a German named entity tagger, which is pretrained on the Conll-03 dataset (Sang and De Meulder, 2003). First, raw and unprocessed text is fed sequentially into the encoding part of the bidirectional language model. Second, we retrieve for each word i a contextual embedding by concatenating the forward model's hidden state after word i and the backward model's hidden state before word i. This word embedding is then passed into a vanilla BiLSTM-CRF[19] sequence labeler. We apply this sequence tagger on our raw input documents and consider all predicted named entities as the document's important words.
- **Flair grammar predictor** In order to evaluate grammatical correctness, we again leverage the aforementioned flair language model, which was trained as an auto-encoder to correctly predict the next character in a text. For a grammatically correct text we would expect the model to mostly guess the next character correctly.

[16] https://github.com/scikit-learn/scikit-learn.

[17] We apply the same document preprocessing as in Section 3.3.2 before calculating the co-occurrence matrix. Also, we choose a window size of 5 and each context word j contributes $1/d$ to the total word pair count, given it is d words apart from the base word i.

[18] Long Short Term Memory.

[19] Bi-directional Long Short-Term Memory Conditional Random Field.

A text with grammatical errors however would not match the expectations of the model, thus creating a larger reconstruction error on the characters that do not fit grammatically. To assess grammatical correctness we feed the summary text through the model and score the summary based on the accumulated reconstruction error.

- **Sentence-BERT predictor** We explore how sentence embeddings can be used to measure "how similar" (semantically) a summary is compared to its original text. In particular, we infer sentence embeddings with the pretrained *bert-base-german-uncased* BERT model from the HuggingFace's transformers library (Wolf et al., 2019) in the fashion proposed with the Sentence-BERT architecture (Reimers and Gurevych, 2019). The output of the BERT model is max-pooled to obtain a fixed-size vector for each processed piece of text. This way, we can obtain embeddings for both the original text and each of the summaries. The resulting predictor score is thus the cosine similarity of the summary vector with the original text vector.
- **ROUGE predictor** The ROUGE score is a classic metric for assessing the quality of summaries. Even though it alone is not sufficient to evaluate summaries it can give useful insight when applied in an ensemble setting. We calculate the rouge-1, rouge-2 and rouge-L scores between the summary and both the full original text and the reference summary. While rouge-1 and rouge-2 calculates the overlap of unigrams and bigrams (i.e. single words and adjacent word pairs) between reference text and summary, rouge-L evaluates the longest common subsequence between reference and summary.
- **BLEU predictor** BLEU is a metric that calculates an n-gram precision between one or multiple reference texts and a summary hypothesis, in which n-gram counts in the summary are compared to their maximum count in one of the references.
- **METEOR predictor** METEOR is a metric that calculates a harmonic mean between the recall and precision of an n-gram matching which considers word order between a reference text and a summary.
- **Compactness predictor** We calculate the compactness score as the compression rate with respect to the original text, where the text length is measured by the number of characters.
- **Number matching predictor** A good summary should be factually correct. While there might be some ambiguity from different word choices between original text and summary, there usually is only one way to display exact numbers like dates. We thus expect every number in the summary to also appear in the original text. To assess factual correctness regarding numbers, we count how many of the numbers in the summary are also present in the text.
- **Sentence copying predictor** At times, one can generate a usable summary by simply extracting the first sentences of the original text, since they often provide an introduction and therefore a mini-summary of the remaining text. However, the goal of our evaluation is finding abstractive and novel summaries. We therefore perform a binary check on whether the summary exactly matches the first sentences of the original text and assign a 1 if they are extracted from the original text and a 0 if they are more abstracted.

	coef	std_err	P> |t|
constant	0.072	0.095	0.447
tfidf_content	0.535	0.107	0.000
flair_grammar	0.226	0.109	0.038
sbert	0.169	0.106	0.110
sentence_copying	−0.168	0.064	0.009
rouge-1	2.560	0.571	0.000
rouge-2	−1.531	0.340	0.000
rouge-L	−1.329	0.646	0.041

Table 3.5: Regression coefficients, standard errors and p-values for final predictor set.

3.3.3 Evaluation

In this section, we report and analyze our results of employing a capped linear regression model to ensemble the significant subset of our predictors to generate a representative summarization quality metric. We start by fitting a capped linear regression model to the full set of predictors, including an intercept, and consider the p-values of each predictor. We iteratively remove the most insignificant predictor (largest p-value) and re-run the linear regression. We stop once all predictors are statistically significant to the 15% level.

The final regression model on the remaining 7 significant predictors is described in Table 3.5.

The columns show the estimated coefficients, standard errors and p-values of each predictor. Since all predictors have been normalized (min-max normalization) prior to the regression, their regression coefficients are directly comparable in magnitude. It can be seen that the rouge-1 predictor has the highest coefficient and thus, is most important for predicting the summary evaluation score. However, the other predictors also contribute significantly to the prediction outcome, which gets evident when comparing the final ensemble error of 33.72 (see Table 3.6) to the individual rouge-1 error of 35.99 (see Table 3.7).

Further, the coefficients of the sentence copying, rouge-2 and rouge-L predictors imply a negative correlation to the annotated summary scores. This is expected because all three predictors yield high scores, when entire sentences, bigrams or common subsequences of the original documents get copied to or make up the generated summaries. Yet, our annotations favor abstractive summaries which is why a higher score of one of the above predictors indicates a worse summary when taking abstractiveness as a quality indicator into account.

Table 3.6 shows the final error values obtained by different predictor ensembles in the shared task public ranking. Despite of more predictors increasing the likelihood of overfitting on our manual annotations and thereby lowering our final error score, one can observe the opposite. Removing insignificant predictors actually yields the best performing model and puts us among the top participating teams.

Ensemble	Error	Predictors
7 predictors	33.72	constant, tfidf_content, flair_grammar, sentence_copying sbert, rouge-1, rouge-2, rouge-L
10 predictors	33.90	+ nmf_content, bleu, meteor
13 predictors	33.82	+ flair_ner_content, compression, number_matching

Table 3.6: Error values obtained in the shared task public ranking by different predictor ensembles. A lower value means better performance.

Predictor	Error (original)	Error (fitted)
rouge-1	44.26	35.99
rouge-2	52.50	36.08
rouge-L	44.27	36.12
bleu	64.16	36.11
meteor	53.05	36.06

Table 3.7: Error values obtained by some of the common evaluation metrics for automatic text summarization after uploading their scores to the shared task public ranking. A lower value means better performance. The middle column represents the errors for the min-max normalized predictor scores. The right column shows the final errors for the normalized predictor scores being fitted via linear regression to our manual summary annotations.

3.3.4 Comparison with Standard Metrics

In order to show the validity of our approach and its improvement over previously established methods, we take a look at the performance of BLEU, METEOR and ROUGE as single predictors.

We implement each metric using the standard definition and further employ min-max normalization as described above in order to receive a metric that assigns a score between 0 (bad) and 1 (good) so that both extremes appear in the dataset. This approach is developed entirely without manual annotations. The scores received on the challenge task are depicted in the middle column of Table 3.7.

Furthermore, we use our manual annotations to adjust the predictors to the available dataset, fitting a linear regression of a single predictor to the annotated summary scores. These scores are depicted in the right column of Table 3.7.

As already signified, we see that using these metrics out-of-the-box results in significantly worse performance than both the fitted algorithm and our ensemble approach. While the fitted metrics score is considerably higher than their original counterpart, we still see a distinct improvement when employing an ensemble of different predictors.

3.3.5 Conclusion and Future Work

We showed that a hybrid combination of rule-based, statistical and deep-learning techniques outperforms other alternatives for automatic evaluation of automatically generated German text summarization given the provided shared task dataset.

Although the text corpus covers a wide range of topics, the text style is quite homogeneous. Mostly, it consists of generally grammatically perfect descriptive texts. It would be interesting to test if our approach also works for more informal noisy texts. Furthermore, it would be also interesting to evaluate different state-of-the-art summarization approaches with our new metric.

Chapter 4

RatVec: An Explainable Similarity-based Representation Learning Framework for Finite Domain Sequences

The evaluation strategies presented in Chapter 3 help to alleviate the opacity of the models under inspection to increase trust in the deployed applications. Nevertheless, the insights these approaches can provide about the underlying models may be insufficient. Additionally, deep learning models may be too computationally expensive for some settings, prompting a search for explainable, resource-aware alternatives.

In this chapter, we put together the theoretical representation learning framework via rational kernels elaborated by Brito, Georgiev, et al., 2019 with its published applications for text, including spelling correction (Beeksma et al., 2018), historical text analysis (Schraagen, Wall, and Brito, 2020) and part-of-speech tagging (Brito, Sifa, and Bauckhage, 2017); and for biological sequences, in particular, splice-junction detection on DNA sequences (Brito, Sifa, and Bauckhage, 2017) and protein family classification (Brito, Georgiev, et al., 2019). We demonstrate how to construct an explainable-by-architecture classification pipeline that can achieve competitive performance while keeping its computational requirements much lower than the best-performing models.

4.1 Introduction

The success of distributed vector representations in various natural language processing tasks has motivated their adoption in other fields, such as biological sequence analysis (Asgari and Mofrad, 2015). Most existing approaches consider similarity between words in terms of the *distributional hypothesis* (Harris, 1954): "words that occur in similar contexts are similar". Consequently, they generate vector representations that are close to each other in their vector space if the entities they represent frequently appear together. However, state-of-the-art models of this kind usually encounter issues that preclude them from a series of use cases, as we discussed in Chapter 3. On the other

hand, relying solely on interpretable features can result in models that are too limited when the goal is to increase user trust in the systems built upon them.

As an alternative to both, we implement a general framework to generate dense vector representations of non-numeric entities such as text, DNA, and protein sequences. This is based on similarity functions that can be expressed as rational kernels (see Section 2.6). By obtaining vector representations via KPCA that we use for a straightforward KNN classifier, we construct an efficient and explainable (due to its similarity-based nature) classification pipeline demonstrating competitive performance across various tasks.

In essence, the machinery of rational kernels allows us to design domain-specific (i.e., knowledge-based) similarity functions, supported by a robust theoretical framework and efficient computation algorithms. Moreover, applying domain-specific similarity functions curated by domain experts (i.e., an informed machine learning methodology) can not only be more efficient than learning from scratch in a fully data-driven approach, but it can also reduce complexity, diminishing the size of necessary training data to generate high-quality representations. In setups where data is high-dimensional and the number of examples is limited, knowledge-based learning seems to be more appropriate than purely data-driven deep neural models.

Our work is characterized by the following features:

(i) By learning suitable representations, we derive competitive solutions with no need for complex (computationally expensive) classification models: a simple KNN classifier with a very small k suffices.

(ii) Our similarity-based classification pipeline is *explainable*: an entity will be classified into a particular class because it is *similar* (in a domain-specific, explainable sense) to the labeled entities that the classifier was trained with.

Since many reasonable knowledge-based similarity functions tend to lack certain formal properties (e.g., definiteness), we highlight some central insights from the field of indefinite kernel methods to provide a sound theoretical interpretation of our empirical results (e.g., KPCA algorithms in the setting of pseudo-Euclidean geometry).

4.2 Related Work

The approach presented here generalizes our previous work on *KPCA embeddings* (Brito, Sifa, and Bauckhage, 2017), where we studied vector representations for words and DNA sequences via KPCA, with the dot (scalar) product replaced by a specific similarity function. There exists a large body of relevant literature - below we mention several works that are most related to our work.

Cristianini, Shawe-Taylor, and Lodhi, 2002 presented a kernel method for text classification that learns a kernel function by applying singular value decomposition (SVD) on a document matrix, where each document is represented by a term frequency vector. The kernel function is used to train a support vector machine (SVM) for text classification. This is analogous to our approach for documents viewed as a bag of words where the similarity function is the intersection of terms of both documents. In this case, the similarity is learned rather than given (contrasting with our "informed learning" ap-

proach). Furthermore, while their approach involves training an SVM classifier, our work leverages KNN classifiers.

Similarly, Lodhi et al., 2002 used string kernels for the same task. Although the kernels they present are analogous to the similarity functions used within our framework, they trained the SVM model directly. Conversely, the approach in Lodhi et al., 2002 does not learn any explicit vector representation, and the direct SVM approach appears to lack explainability.

Further studies (Giuliano, 2009) combine a latent semantic kernel similar to (Cristianini, Shawe-Taylor, and Lodhi, 2002) for named entity recognition. In this case, the logarithm of the inverse document frequency $(log(idf))$ is also added.

Ishii et al., 2006 presented a pipeline where they first grouped similar words, after which LSA was applied for document classification via KNN. Assuming one omits the grouping step this approach introduces, it is similar to our work, particularly with a similarity function based on term frequency and $log(idf)$.

4.3 Theoretical Background

We utilize the framework of rational kernels, i.e., kernels induced by certain automatons, and KPCA. We briefly recalled the main tools and ideas on Sections 2.5, 2.6, and 2.7. For thorough background on the subject, we refer to Cortes, Haffner, and Mohri, 2004.

4.3.1 Indefinite Kernel PCA

We briefly discuss the case of indefinite KPCA which will be relevant for our experimental results. Suppose a data set $\{x_i\}_{i=1}^n \subset X$ and a symmetric kernel function $\kappa : X \times X \to \mathbb{R}$ are given and consider the eigenequation:

$$L\alpha = \lambda\alpha, \tag{4.1}$$

where L is the (normalized) matrix as in the application of :

$$L_{ij} := \kappa(x_i, x_j) - \frac{1}{n}\sum_{s=1}^{n}\kappa(x_i, x_s) - \frac{1}{n}\sum_{s=1}^{n}\kappa(x_j, x_s) + \frac{1}{n^2}\sum_{s,l=1}^{n}\kappa(x_s, x_l). \tag{4.2}$$

Here, according to the indefiniteness assumption, the matrix κ possesses both positive and negative eigenvalues. This is not an obstruction for applying the projection methods outlined in Section 2.5 and performing experiments. However, the above eigen- and optimization- problems should be seen from the perspective of pseudo-Euclidean spaces:

Proposition (X. Huang et al., 2016). Suppose κ is an indefinite kernel and let α^* be a solution of (4.1).

Then, there exists feature mappings F_+, F_- giving the stationary point

$$(w_+^*, w_-^*) := \left(\sum_{i=1}^{n}\alpha_i^*\left(F_+(x_i) - \mu_+\right), \sum_{i=1}^{n}\alpha_i^*\left(F_-(x_i) - \mu_-\right)\right), \tag{4.3}$$

to the following primal problem:

$$\max_{w_+,w_-,\xi} \frac{\gamma}{2}\sum_{i=1}^{n}\xi_i^2 - \frac{1}{2}\left(w_+^T w_+ - w_-^T w_-\right), \tag{4.4}$$

$$\text{s. t.} \quad \xi = w_+^T\left(F_+(x_i) - \mu_+\right) + w_-^T\left(F_-(x_i) - \mu_-\right), \quad i = 1,\ldots,n. \tag{4.5}$$

Here we have also set

$$(\mu_+, \mu_-) := \left(\frac{1}{n}\sum_{i=1}^{n}F_+(x_i), \frac{1}{n}\sum_{i=1}^{n}F_-(x_i)\right). \tag{4.6}$$

Furthermore, the primal problem (4.4) is dual to the eigenproblem (4.1). ∎

Proof (*Sketch*). We decompose the indefinite kernel κ into positive and negative part, that is

$$\kappa = \kappa_+ - \kappa_-, \tag{4.7}$$

where κ_+, κ_- are both positive semidefinite. Then according to Theorem 2.7.3 we find feature mappings F_+, F_- with

$$\kappa_+(x_i,x_j) = F_+(x_i,x_j)^T F_+(x_i,x_j), \quad \kappa_-(x_i,x_j) = F_-(x_i,x_j)^T F_-(x_i,x_j) \tag{4.8}$$

We now need to demonstrate that the above primal problem has (w_+^*, w_-^*) as a stationary point and is moreover dual to the eigenproblem. This is achieved in a direct manner by writing out explicitly the corresponding Lagrangian equations. ∎

To summarize, the above discussion sheds light on the interpretation of indefinite kernel methods. In particular, the KPCA algorithm can be understood as an optimal variance problem in a suitable pseudo-Euclidean space. This standpoint aids the theoretical explanation of our empirical results as we mainly work in a setting where the kernels are not by definition definite and we do not pursue the introduction of suitable (positive) definite approximations. As it is seen below, the experiments with our particular indefinite kernels deliver competitive performance.

4.3.2 Similarity Functions Based on *n*-grams

Many entities can be represented as a string of elements from a predefined finite vocabulary. For instance, words can be seen as a sequence of characters from a specific alphabet, DNA sequences as a finite series of the four nucleobases or proteins as a sequence of amino acids. Hence, string similarities are widely applied in natural language processing and bioinformatics. Moreover, many of these string similarities can be expressed by means of rational kernels.

The length of the longest common subsequence (LCS) of two sequences can be used as similarity function. For an alphabet Σ, LCS can be trivially modeled as a transducer with one initial and accepting state and $|\Sigma + 1|^2$ transitions, where each transition has

either weight 1 if the input label belongs to the alphabet Σ and its identical to the output level, or it has weight 0 else.

Kondrak, 2005 generalizes LCS to *n*-grams by defining the *n*-gram similarity (*n*-SIM). His concept of *n*-SIM of is equivalent to defining an *n*-gram alphabet Σ_{n-gram} as the *n*-fold Cartesian product of Σ with itself, then converting each sequence to a sequences its *n*-grams and finally computing LCS on them. Hence, *n*-SIM can also be expressed as a finite-weighted transducer and is thus a rational kernel.

4.3.3 Similarity Based on Sequence Alignments

Sequence alignment algorithms use dynamic programming and backtracking techniques to optimize an alignment scoring function that can either match two identical tokens, match two different tokens at a cost, or insert a gap at a cost then ultimately recover the sequence with the minimum score. These scores can be transformed into similarity metrics, and an all-by-all calculation of pairwise alignments of a given corpus can be used to generate a similarity matrix appropriate for KPCA embedding.

Global alignment algorithms (Needleman and Wunsch, 1970) are appropriate for scoring pairwise alignments of protein sequences, whose entire sequences should be aligned. Alternatively, local alignment algorithms (Smith and Waterman, 1981) are more appropriate scoring pairwise alignments of genomic sequences, in which several regions in each of a pair of sequences may be independently related. In a contrast to more simplistic implementations of pairwise alignment algorithms which use a constant cost for mismatching tokens, biological applications use cost matrices with entries for each pair of tokens i and j mapping to integers that are calculated based on prior biological knowledge about sequence alignments. The most famous, the blocks substitution matrices (BLOSUM), were introduced in order to support the calculation of protein sequence alignments (S. Henikoff and J. G. Henikoff, 1992).

Derivation of BLOSUM

Prior knowledge and programs like PROTOMAT (S. Henikoff and J G Henikoff, 1991) used to categorize similar sections, or domains, of proteins were used to manually identify highly locally aligned regions. The observed frequency of occurrence f_{ij} of a pair of amino acids i and j was calculated counting the occurrence of all pairs of amino acids over all pairs of sequences in the local alignments (which usually contain more than two sequences).

$$q_{ij} = \frac{f_{ij}}{\sum_{m=1}^{20}\sum_{n=1}^{m} f_{mn}} \tag{4.9}$$

The observed probability of occurrence q_{ij} of a pair of amino acids i and j is presented in equation (4.9).

$$p_i = q_{ii} + \frac{1}{2}\sum_{j=1, i\neq j}^{20} q_{ij} \tag{4.10}$$

The probability of the occurrence of the i^{th} amino acid in a given pair is given by equation 4.10.

$$e_{ij} = \begin{cases} p_i p_j & i = j \\ 2 p_i p_j & i \neq j \end{cases} \tag{4.11}$$

The expected probability of occurrence e_{ij} of a pair of amino acids i and j is presented in equation 4.11.

$$s_{ij} = \log_2 \frac{q_{ij}}{e_{ij}} \tag{4.12}$$

The log of odds ratio s_{ij} of the observed probability q_{ij} and the expected probability e_{ij} of of a pair of amino acids i and j is presented in equation 4.12. Ultimately, these ratios are scaled by a factor of 2 and rounded to the nearest integer to generate the BLOSUM matrix.

Using standard implementations of either local or global sequence alignments, such as those provided by BioPython (Dalke et al., 2009), we can calculate all-by-all pairwise alignments. Finally, a matrix encoding the pairwise similarities of all elements of a corpus S can be generated using the following transformation:

$$S_{ij} = \frac{1}{1 + score(i, j)} \tag{4.13}$$

4.4 Approach

We exploit the fact that we can define kernel functions also on non-numeric entities (e.g., words) so that we can generate vector representations from the principal components derived from KPCA. As we saw in Section 2.5, KPCA can extract an arbitrary number d of principal components of a data point t, by

(i) Computing a kernel matrix $\mathbf{K}$ as in Equation (2.6),
(ii) Diagonalizing $\mathbf{K}$ to construct a projection matrix $\mathbf{P}$ as in Equation (2.9),
(iii) Project the data point t to a d-dimensional vector $\mathbf{u_t}$ as in Equation (2.10).

Let X be a dataset consisting of n (non-numeric) elements, to which we will further refer as the *full vocabulary*; and k a rational kernel (a particular similarity function). Computing a kernel matrix $\mathbf{K}$ from X may be computationally prohibitive for large datasets due to its time and space complexity ($O(n^2)$) unless approximations or implementations tricks are performed. Since KPCA allows to project unseen data points as long as you can evaluate the kernel function on them together with the elements processed to build $\mathbf{K}$, we can also select just the m elements that we consider "most representative" (in a domain-specific formulation) from X. These representative elements constitute our *representative vocabulary* V. The choice of m can be adjusted to fit the computation resources of the user.

Once the representative vocabulary V is defined, we can compute a kernel matrix $\mathbf{K_V}$ from all element pairs from V. After centering and diagonalizing $\mathbf{K_V}$, we construct our projection matrix P_V. This is required to generate m-dimensional representations for the

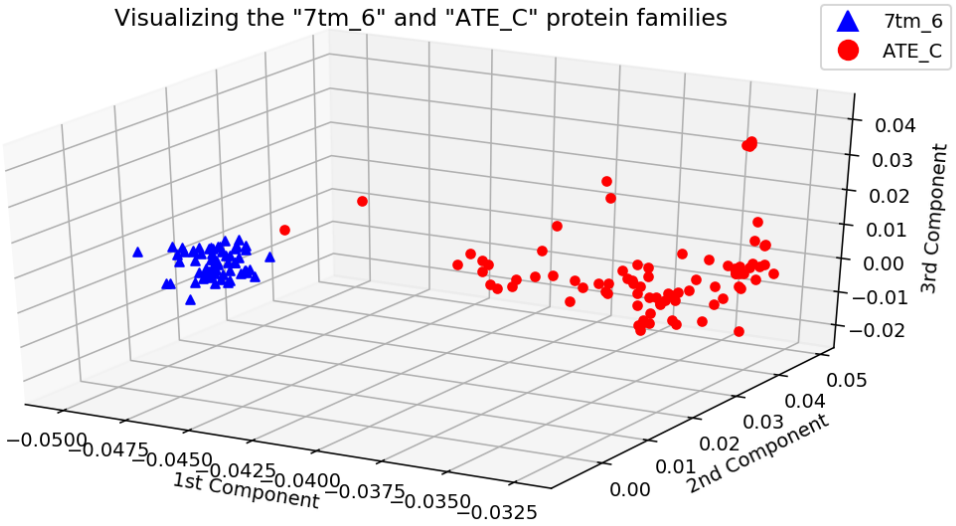


Figure 4.1: Visualization of learned vectors for two different protein families using bigram similarity and a representative vocabulary of 1,000 sequences. The first three components of the vectors provide clear separation between the two families, demonstrating the effectiveness of our approach. Best seen in color.

full vocabulary. In a last step, we assign the first $d \leq m$ components of each computed vector to each full vocabulary element, which constitute the final d-dimensional vector representations that we obtain from our approach. When the applied similarity function is suitable for the task, d being of a lower order of magnitude than m can lead to optimal results, as we will see in Section 4.5.

The produced representations tend to cluster naturally according to the domain-specific concept of similarity applied by the selected rational kernel, as we can see in Fig. 4.1. This has two main advantages:

(i) A simple KNN classifier (eventually 1NN) can suffice for classification tasks.
(ii) Learning the vector representations and KNN constitute an *explainable* classification pipeline: an entity is assigned to a particular class because it is *similar* to the labeled entities used to train the classifier. This level of interpretability is a significant advantage of our method, as it provides transparency into the decision-making process of the model. In many applications, particularly those with regulatory oversight or significant consequences (such as medical diagnoses or financial predictions, as motivated in Section 1.1), the ability to understand and justify model decisions can be just as important as accuracy.

4.5 Applications

The RatVec framework enables classification of sequences from very different nature, including

(i) Text: for part-of-speech (PoS) tagging (Brito, Sifa, and Bauckhage, 2017), for spelling correction; (Beeksma et al., 2018), historical text analysis (Schraagen, Wall, and Brito, 2020)
(ii) DNA: for splice-junction detection (Brito, Sifa, and Bauckhage, 2017);
(iii) and proteins: for protein family classification (Brito, Georgiev, et al., 2019).

4.5.1 German Verb Classification

We chose German verb classification as our first proof-of-concept task due to the challenges it involves. German is a language with a rich morphological structure, where changes in tense, person, number, and mode are often expressed through modifications to the verb itself. As such, a model capable of accurately classifying German verbs would demonstrate its strength in handling complex morphological patterns, thus serving as a strong proof-of-concept for our RatVec framework.

We restrict our vocabulary to the tokens tagged as verbs from the TIGER treebank (Brants et al., 2004). This simplifies the problem to classify the correct morphological tag (consisting of grammatical person, number, tense and mode when they apply) of a German verb only from its representation.

First, we extract all tokens tagged as verb (corresponding to the TIGER tags VVFIN, VAFIN, VMFIN, VVIMP, VAIMP, VVINF, VVIZU, VAINF, VMINF, VVPP, VMPP, VAPP) and remove all duplicates. This leads to 13370 unique verbs with 31 different morphological tags, whose distribution is showed in figure 4.2. We build a training set consisting of 80% of the verbs and a test set with the remaining 20%. Then, we apply the approach described in section 4.4.

We adapt the the Sørensen-Dice coefficient by considering not only bigrams, but n-grams of any length in general to construct our similarity function. Let $\mathcal{G}_n(w)$ the n-grams of a word w. We define a similarity function s of two words $x, y \in V$ as follows:

$$s(x,y) = \sum_{n \in \mathbb{N}^+} \alpha_n \frac{2|\mathcal{G}_n(x) \cap \mathcal{G}_n(y))|}{|\mathcal{G}_n(x)| + |\mathcal{G}_n(y)|}, \quad \sum_n \alpha_n = 1 \tag{4.14}$$

where α_i determines the weight of the Sørensen-Dice coefficient term for each n-gram length. We apply non-linear kernel function (in particular different RBF kernels and polynomial kernels) to s to define the kernel function k that fits in our framework. We consider only bigrams and trigrams to compute the similarity function for each pair of verbs by selecting five different weight distributions (different values for α_2 and α_3 in equation 4.14). We also incorporate an additional character at the beginning and at the end of each verb when producing the n-grams. After running our framework on the training set, we infer vector represen tations of the verbs from the test set. By using the produced representations as a features and the morphological tag as label, we train k-nearest neighbors classifiers to predict the morphological tag of a word from only

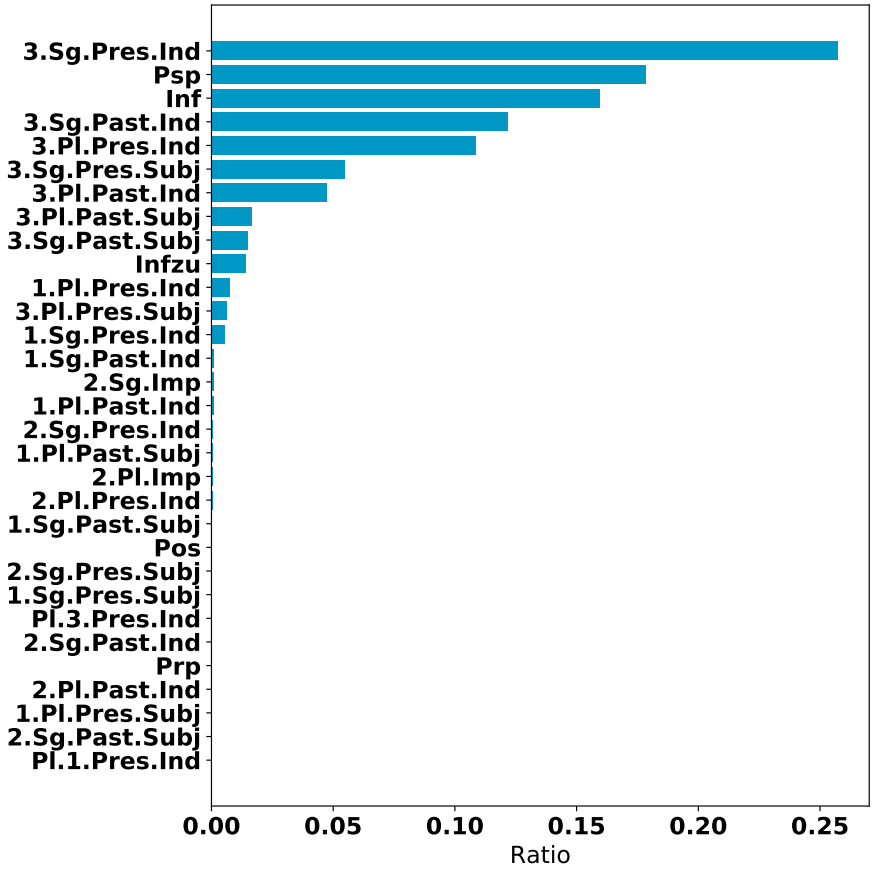


Figure 4.2: Distribution of 31 different morphological tags from the TIGER treebank. Note the imbalance especially with respect to first and second person forms.

the KPCA embedding. As baseline representations, we also learn a word2vec model (Mikolov, Sutskever, et al., 2013; Mikolov, K. Chen, et al., 2013) for each different vector size (by the time we released our experiments, word2vec was the de facto standard for word embeddings). These word vectors were learned applying the default hyperparameter values.

From Table 4.1 we can observe that a mean accuracy above 77% can be achieved by classifiers taking only the nearest neighbor ($k = 1$). This can be interpreted as a high accuracy considering the extremely unbalanced label distribution (see figure 4.2). Among the trained classifiers, we can also find some improvement when the trigram similarity weight (α_3) is at least as high as the bigram similarity (α_2). In addition, any RatVec model beats all word2vec models for this task. For the sake of a fair comparison, the displayed word2vec results from Table 4.1 correspond to models where their vector size and the k values for k-nearest neighbors match. Nonetheless, we also tested additional word2vec models with vector sizes up to 100 and up to 100 neighbors. None of these larger models reached a mean accuracy above 27%.

d	α_3	$k=1$	$k=3$	$k=5$	$k=10$
RatVec					
5	1	76.76	65.20	63.02	60.56
5	0.75	76.67	65.94	63.74	60.85
5	0.5	76.56	65.78	63.33	60.50
5	0.25	76.72	65.14	63.01	60.27
5	0	**77.01**	65.14	62.64	59.60
word2vec					
5		12.08	10.92	13.43	15.56
RatVec					
10	1	**77.38**	66.64	64.92	62.59
10	0.75	77.13	66.26	63.70	61.37
10	0.5	76.59	64.87	62.19	59.91
10	0.25	76.63	65.31	62.60	59.65
10	0	77.05	65.96	63.72	60.68
word2vec					
10		13.91	12.23	15.41	17.39
RatVec					
15	1	77.55	66.96	65.13	63.27
15	0.75	76.97	65.96	64.02	61.79
15	0.5	77.28	66.55	64.62	62.22
15	0.25	77.46	67.45	65.45	63.65
15	0	**77.72**	67.46	65.43	63.83
word2vec					
15		14.29	13.99	17.69	20.08
RatVec					
20	1	77.22	66.43	64.92	62.75
20	0.75	77.11	65.84	64.68	61.65
20	0.5	**77.56**	66.42	65.11	62.52
20	0.25	77.55	67.03	65.16	63.20
20	0	77.49	66.96	65.47	63.59
word2vec					
20		13.76	14.47	18.61	20.91

(**a**) Polynomial kernel (degree 3)

d	α_3	$k=1$	$k=3$	$k=5$	$k=10$
RatVec					
5	1	**77.08**	64.95	63.02	60.48
5	0.75	76.89	66.03	64.50	61.38
5	0.5	76.75	66.05	63.78	61.86
5	0.25	76.95	65.69	63.64	61.02
5	0	76.89	65.53	63.65	61.26
word2vec					
5		12.08	10.92	13.43	15.56
RatVec					
10	1	76.80	66.38	64.54	61.92
10	0.75	76.72	66.83	64.30	61.95
10	0.5	77.17	66.70	64.36	61.96
10	0.25	**77.23**	66.78	64.40	62.43
10	0	77.13	66.67	64.42	62.16
word2vec					
10		13.91	12.23	15.41	17.39
RatVec					
15	1	77.28	66.96	64.82	62.60
15	0.75	77.05	66.66	64.44	61.90
15	0.5	**77.23**	66.17	64.01	62.00
15	0.25	76.82	65.51	63.45	61.11
15	0	76.51	65.34	63.76	60.98
word2vec					
15		14.29	13.99	17.69	20.08
RatVec					
20	1	**77.51**	67.00	65.06	62.67
20	0.75	76.93	66.00	64.23	62.22
20	0.5	77.05	65.29	63.76	61.61
20	0.25	76.67	64.94	63.10	61.22
20	0	76.43	64.70	62.76	60.94
word2vec					
20		13.76	14.47	18.61	20.91

(**b**) RBF kernel ($\sigma = 2.26$)

Table 4.1: Mean accuracy in % predicting the verb tag with k nearest neighbors, trigram ratio α_3 ($\alpha_2 = 1 - \alpha_3$) and d principal components applying different kernel functions. For the word2vec baselines, d refers to the word vector size.

Error type (occurrences)	Valkuil (baseline)			RatVec (ours)		
	TP	FN	Acc.	TP	FN	Acc.
capitalization (165)	0	0	-	105	12	0.90
non-word (62)	21	0	1.0	24	15	0.62
redundant punctuation (53)	0	3	0.0	8	5	0.62
missing punctuation (18)	1	0	1.0	3	0	0.5
archaic spelling (5)	0	0	-	1	0	1.0
TOTAL (303)	22	3	0.88	141	35	0.80

Table 4.2: Results of our system compared to Valkuil (CLIN28 shared task baseline) in terms of successful corrections (TP), wrong corrections (FN), and accuracy (Beeksma et al., 2018)[a].

[a] https://github.com/LanguageMachines/CLIN28_ST_spelling_correction

4.5.2 Dutch Spelling Correction

Dutch spelling correction presents a significant challenge due to its complex morphology and the influence of multiple dialects. Spell checkers involve two steps: misspelling detection and correction. The latter generally requires ranking a set of correction candidates. Generating them implies finding all valid words differing from the detected misspelling less than a determined edit distance, which is mostly set to 1 for real-world applications to limit computation time (Tijhuis, 2014), but this strategy may miss valid corrections. Our RatVec approach bypasses this constraint by computing a vector representation for all words in our vocabulary and selecting corrections based on similarity in the vector space. This method increases the potential pool of corrections without an exponential increase in computational cost.

This section reports our participation in the CLIN28 shared task, where our system obtained the best F1 score among the competing teams (Beeksma et al., 2018). The task focused on correcting errors in extracts from Dutch Wikipedia pages. We used an obsolete implementation of our approach that we keep for reproducibility purposes[1].

In our RatVec framework, our full vocabulary is *wdutch*, a word list from the OpenTaal project, from which the 3000 most frequent words form our representative vocabulary. The applied rational kernel is the composition of the bigram similarity (Kondrak, 2005) with the homogeneous polynomial kernel of degree 2. We generated a vector representation for each full vocabulary word with 2000 dimensions.

Once a misspelling is detected, we compute its RatVec representation and search for the closest precomputed vector. Its related word (its nearest neighbor) is our correction to the misspelling. Formally, this is equivalent to training a 1NN classifier where each valid word is assigned a different label.

Our RatVec framework is only relevant during the correction phase for spelling mistakes that are related to the word form. Hence, we restrict our analysis to the correction results on the five error categories where the word form is relevant, namely those displayed in Table 4.2. Although the baseline system Valkuil achieves a better average

[1] https://github.com/fraunhofer-iais/kpca_embeddings

(i) *Graf Hendrick van den Berch heeft daer gewoont* ***gehadt***.
(ii) *Graf Hendrick van den Berch heeft daer gewoont*.

Figure 4.3: The first sentence shows an original example where have-doubling takes place (translated: "Count Hendrick van den Berch has there lived had"). Removing the *have*-participle "gehadt" results in the second sentence, which is used as example to construct the dataset to train text classifiers.

accuracy than our approach for the analyzed misspellings, RatVec outperforms Valkuil in some categories where it completely fails (redundant punctuation errors) or where it cannot be even evaluated because it failed to detect any error (capitalization and archaic spelling errors). From these results, we interpret that our word vectors encode word forms in a suitable way so that similar words can be retrieved.

4.5.3 Have-doubling Context Detection in Historical Varieties of Dutch

The objective of the CLIN30 Shared Task was to determine whether a given sentence provides the context for a have-doubling construction, which is a syntactic phenomenon combining a past participle construction with an additional participle, e.g., "he has had lived there" (Schraagen, Wall, and Brito, 2020). Although have-doubling takes place nowadays only in some dialects of Dutch, German, and French and it is not present in their modern standard variants, recognizing this structure has some applications on historical text analysis and under-resourced NLP (Schraagen, Wall, and Brito, 2020).

The main aspect of interest is not to detect have-doubling as such, but rather to discover which properties of the sentence are related to it. Hence, the defining occurrence of the past participial form of *have* were striped out from examples of have-doubling to construct the shared task dataset. The task for a text classifier is thus to determine whether a sentence originally contained have-doubling or not, i.e., whether the *have*-participle has been removed from a sentence or not. Figure 4.3 illustrates an example where a sentence is constructed from an actual example of have-doubling, for which the classifier should predict that this sentence was constructed from an actual example of have-doubling. If it is feasible to classify these sentences, we can argue that sentences with have-doubling are different in some extent. If we can also make use of interpretable features and analyze the results of the text classifier, we may better understand the nature of the have-doubling phenomenon.

The resulting dataset consists of 1044 example sentences in historical Dutch, ranging from the 13th century to the 19th century. Half of the examples (522 sentences) contain *have*-doubling. The other half of the data contains negative examples, i.e., sentences without perfect doubling. All data is selected from two different sources: 284 positive examples and 522 negative examples from the Digital Library of Dutch Literature (DBNL)[2], containing documents from all historical time periods for written Dutch of usually very high quality, with virtually no transcription errors; and the Nederlab project, predominantly (227 examples) from the correspondence archive of the Dutch

[2] https://www.dbnl.org

Approach	*Accuracy*
RatVec	0.67
Naive Bayes	0.77
Logistic Regression	**0.79**
LSTM (pre-trained embeddings)	0.65
LSTM (on-the-fly embeddings)	0.72

Table 4.3: Overview of classification accuracy of the tested approaches for the CLIN 30 Shared Task.

politician Anthonie Heinsius (1641-1720). The latter resource consists of letters digitized using OCR, and contains a high degree of transcription errors. Almost all of these documents (all but 11) originate from the period between 1710 and 1720, from a variety of correspondents (Schraagen, Wall, and Brito, 2020).

We train a RatVec model computing the composition of the 2-spectrum kernel (Leslie, Eskin, and Noble, 2002) with the RBF kernel ($\gamma = 0.8$) on coarse PoS sequences related to 3000 randomly selected sentences from the subcorpora j,k, l, m, and n of the Corpus Gesproken Nederlands ("Corpus Spoken Dutch") (Eynde, Zavrel, and Daelemans, 2000). The dataset sentences were converted to PoS sequences by means of the Frog parser(Bosch et al., 2007). In order to improve PoS tagging, we translate the historical sentences to modern Dutch with the tool provided for the CLIN27 shared task (Tjong Kim Sang et al., 2017). Then, our RatVec pipeline transforms their PoS tag sequences to 22-dimensional vectors, which constitutes the training dataset for a 19–nearest neighbors classifier.

In parallel, other three machine learning algorithms were applied (Schraagen, Wall, and Brito, 2020):

(i) **Multinomial naive Bayes** with the Scikit-learn implementation (Szymański and Kajdanowicz, 2017), converting the input to lower case and uses a pattern consisting of 2 or more alphanumeric characters to represent tokens, with punctuation characters used as token separator. This creates a vocabulary of around 10 thousand words on which the term frequency matrix is based. The classifier itself has very few parameters, which have been left to Scikit defaults.

(ii) **Logistic regression** also with the Scikit-learn library and same tokenization. This classifier uses the Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm (lbfgs) as solver and a multinomial distribution over the labels.

(iii) **Long Short Term Memory network (LSTM)** as implemented in the Python library Keras with TensorFlow backend, using the softmax activation function, the categorical cross-entropy loss function and the Adaptive Moment Estimation optimizer (Adam). Two different sets of word embeddings were used : pre-trained Word2Vec embeddings generated from a 4.5 million word corpus of Early Modern Dutch (1600–1750) obtained from DBNL, and on-the-fly embeddings based on the input data which are trained together with the classifier.

The RatVec pipeline achieved a mean 10-fold-crossvalidation accuracy of 0.672 (0.701 on the positive class, 0.647 on the negative class). The necessary code to reproduce the

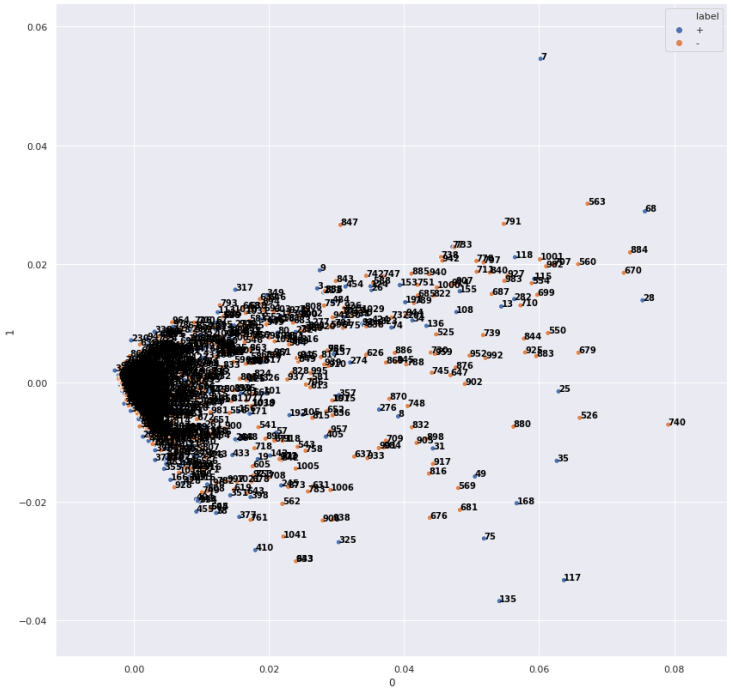


Figure 4.4: Sentences represented by their two first principal components obtained via the RatVec approach. The positive class refers to the sentences containing perfect doubling. Best seen in color.

Class	Nr. sequences	Ratio
EI	767	25%
IE	768	25%
Neither	1655	50%

Table 4.4: Class distribution of the splice-junction DNA sequences dataset

results can be found on Github[3]. The generated vector representations allow the visualization of the represented sentences: similar sentences (according to the applied similarity function) are mapped to vectors close to each other in their vector space. Figure 4.4 shows the two first principal component analysis for the RatVec classifier colored according to their class. This visualization provides some insights. By checking the represented sentences, the first principal component (x-axis) seems to be correlated with sentence length. In fact, a Pearson's correlation of -0.62 (calculated with the sentence length measured as number of tokens) validates this observation (shorter sentences tend to appear rather in the right-hand side of Figure 4.4). This can be explained by the applied 2-spectrum kernel. It indirectly makes sentences "distinguishable" by their number of tokens since the longer the sentences are, the more likely they are going to common bigrams with other sequences. Sentences containing perfect doubling (positive class) are on average longer than the rest, which helps to distinguish them from the rest based on their length (Schraagen, Wall, and Brito, 2020).

The results from Table 4.3 show that simpler models such as Naive Bayes and Logistic Regression generally perform better than the more complex models such as neural networks and the RatVec approach. The RatVec system was trained on the Corpus Spoken Dutch, which is rather different from the language used in the historical sources of the Shared Task. Training on a different, more closely related corpus is likely to result in improved performance. None of the proposed models provide straightforward ways to explain the predictions of the classifier, limiting the possibilities of gaining linguistic insights from the models. However, feature and error analysis has shown a linguistic approach of analyzing the examples of perfect doubling based on classifier results (Schraagen, Wall, and Brito, 2020).

4.5.4 Splice Junction Recognition on DNA Sequences

In this section, we demonstrate the application of RatVec to recognize splice junctions in DNA sequences. We utilized the "Molecular Biology (Splice-junction Gene Sequences) Data Set" from the UCI Machine Learning Repository (Lichman, 2013)[4]. The dataset contains DNA subsequences consisting of 30 characters from the four nucleobases (A, T, C, G), along with four other characters (D, N, S, R) indicating ambiguity. Each sequence might contain a splice junction between the first 30 and the last 30 characters. The sequences are classified into three classes: exon/intron boundary (EI class), intron/exon boundary (IE class), or neither (N class). The class distribution is presented

[3] https://github.com/ebritoc/clin30_ratvec

[4] https://archive.ics.uci.edu/ml/datasets/Molecular+Biology+(Splice-junction+Gene+Sequences)

d	k					
	1	3	6	10	14	19
	Polynomial kernel (degree 2)					
1	57.21	55.49	58.46	61.91	**62.54**	62.07
2	73.98	76.18	79.31	79.78	**81.03**	80.72
3	92.16	92.63	93.42	93.57	**94.51**	93.89
4	91.22	92.32	93.10	93.26	**93.42**	**93.42**
5	90.44	92.63	93.10	93.26	**93.89**	**93.89**
6	89.66	92.79	93.42	92.95	93.57	**94.04**
7	90.75	92.48	92.95	93.57	93.57	**94.51**
8	90.13	93.57	**93.89**	**93.89**	**93.89**	**93.89**
9	90.13	91.22	92.95	93.73	**94.67**	93.57
10	89.50	91.69	92.16	92.95	**93.42**	**93.42**
	RBF kernel ($\sigma = 0.72$)					
1	52.66	56.11	59.09	58.78	**61.44**	60.19
2	70.06	75.24	74.92	**77.12**	76.18	76.49
3	90.60	90.44	90.28	91.38	92.01	**92.32**
4	92.79	93.42	94.04	93.26	93.26	**94.04**
5	92.32	92.79	93.26	94.04	**94.20**	93.89
6	91.85	92.79	93.42	**93.89**	92.95	93.26
7	90.91	93.42	93.89	**94.04**	93.73	93.73
8	90.13	92.16	93.42	**93.57**	92.95	93.42
9	88.87	91.22	92.16	93.89	**94.67**	94.04
10	88.87	91.85	92.63	**93.26**	92.63	91.54

Table 4.5: Mean accuracy in % predicting splice junctions with k nearest neighbors and d principal components applying different kernel functions. Several results beat the baseline system KBANN (93.68% mean accuracy).

in Table 4.4.

To calculate the similarity function as defined in Equation 4.14, we considered all n-grams and assigned an equal weight to all terms:

$$\alpha_i = \begin{cases} 1/58, & i \in \{2, \cdots, 59\} \\ 0, & i \notin \{2, \cdots, 59\} \end{cases} \tag{4.15}$$

We trained *k*-NN classifiers on the computed representations to predict the class of each DNA sequence. The prediction performance, summarized in Table 4.5, reveals a mean accuracy of 94.67% with two different kernels. This performance surpasses all baseline systems provided with the dataset, including the knowledge-based artificial neural network (KBANN) (Noordewier, Towell, and Shavlik, 1991).

The explainable character of our classification pipeline allows to understand why our approach is best-performing by just visualizing the generated representations. Figure 4.5 plots our DNA sequence embeddings using the RBF kernel with $\sigma = 0.72$ and seven

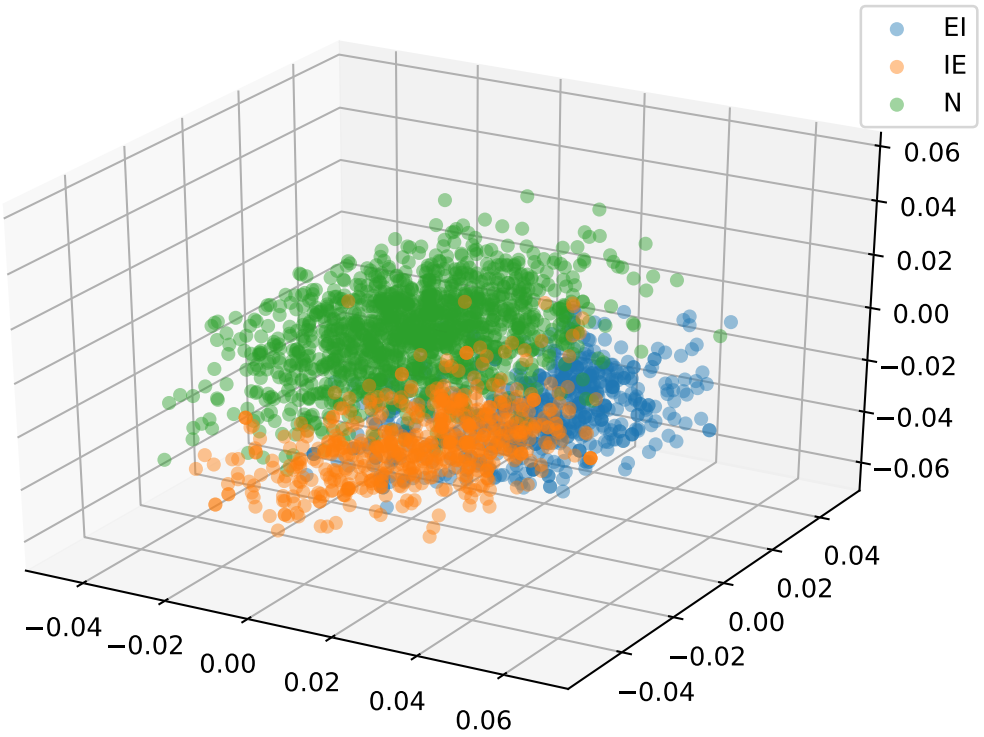


Figure 4.5: Visualization of our embeddings generated from the DNA sequences with RBF kernel, $\sigma = 0.72$ and seven principal components. Our dataset consists of DNA subsequences represented as 30 characters out of the four nucleobases (A, T, C, G) plus other four characters (D, N, S, R) which mark ambiguity. The sequences may contain a spline junction between the 30 first and the 30 last characters. They are thus labeled with three different categories depending if they contain exon/intron boundary (EI class), intron/exon boundary (IE class) or neither (N). Only their first three components are plotted. Best seen in color.

principal components, illustrating a clear differentiation between the three classes.

4.5.5 Protein Family Classification

In this task, we classify protein sequences to their corresponding families with the same Swiss-Prot dataset as stated by Asgari and Mofrad, 2015, containing 7,027 protein families and 324,018 protein sequences[5]. The system is evaluated for each protein family by 10-fold cross-validation on a balanced dataset consisting of all amino acid sequences of the family and as many randomly sampled sequences from all other families.

We produce protein representations with 25 dimensions applying our approach with bigram and trigram similarity and train a nearest neighbor classifier (1NN). We generate our representative vocabulary by taking the shortest sequence of the 1,000 most

[5] http://dx.doi.org/10.7910/DVN/JMFHTN

Dataset	BioVec (baseline)	RatVec (bigram sim.)	RatVec (trigram sim.)
$n = 1000$	0.94	0.94	0.91
$n = 2000$	0.93	0.93	0.91
$n = 3000$	0.92	0.93	0.91
$n = 4000$	0.91	0.93	0.91
All families	0.93	0.93	0.91

Table 4.6: Accuracy of RatVec in different subdatasets compared to BioVec Asgari and Mofrad, 2015 on datasets consisting of the top n families and on the full dataset.

$n = 1000$		$n = 2000$		$n = 3000$		$n = 4000$		All families	
SVM	1NN	SVM	1NN	SVM	1NN	SVM	1NN	SVM	1NN
0.94	0.94	0.93	0.94	0.92	0.93	0.91	0.93	0.93	0.93

Table 4.7: Accuracy of BioVec representations with a SVM (Asgari and Mofrad, 2015) and a KNN classifier on the top n families and on the full dataset.

frequent protein families. We evaluate this pipeline in the same setting as Asgari and Mofrad, 2015. We report the weighted average accuracy results in Table 4.6. Also, the results of the evaluation limited to subsets of the full dataset are presented (e.g. sequences belonging to the 1,000 most frequent protein families).

The results from Table 4.6 show that our approach with the bigram similarity reaches the same accuracy as the baseline and or even outperforms it when we restrict the dataset to the first 3,000 or 4,000 most frequent protein families. Given that the reported BioVec representations are four times longer than ours, our approach to encode proteins demonstrates higher space efficiency, providing comparable performance with a significantly reduced representation size. Additionally, our representations are explainable in the sense that proteins with similar sequences get assigned close vectors in the output vector space.

In contrast to Asgari and Mofrad, 2015, we do not train any support-vector machine (SVM) but a 1NN classifier. To assess the impact of the different classification algorithm, we evaluate a pretrained BioVec model [6] with our setup. The results from Table 4.8, showing that 1NN is more suitable than SVMs for this task, are in line with our previous experiments on classification tasks using distributed vector representations, where simple algorithms such as KNN and logistic regression outperform more complex ones such as random forests or neural networks (Brito, Sifa, and Bauckhage, 2017; Brito, Sifa, Cvejoski, et al., 2017).

4.6 Conclusion and Future Work

We showed that our approach involving rational kernels on KPCA for vector space embeddings provides rich representations for different kinds of sequences, including text,

[6] https://github.com/kyu999/biovec

Dataset	SVM	1NN
Top 1000 families	0.94	0.94
Top 2000 families	0.93	0.94
Top 3000 families	0.92	0.93
Top 4000 families	0.91	0.93
All families	0.93	0.93

Table 4.8: Accuracy achieved by BioVec representations as reported by Asgari by training a SVM (Asgari and Mofrad, 2015) compared to training a KNN classifier

DNA, and proteins. Our results are comparable to state-of-the-art approaches in various tasks. Nonetheless, our approach is comparatively advantageous for real-world applications: on one hand, it makes use of simple KNN classifiers enabling explainability and reducing complexity compared to other existing neural network models; on the other hand, we back our framework by connecting it to the existing theoretical work about on indefinite learning and rational kernels. We only set the constraints on the user-defined similarity function that can be used for our approach to work: they have to be rational kernels.

In future work, we aim to learn optimal similarity metrics (modeled as transducers) that, incorporated in our presented approach, solve a particular task.

Chapter 5

Explainable Text Classification via Semantic Similarity From Foundation Models

Throughout previous chapters, we have examined the opaque nature of neural classification models and the issues it presents. An alternative can consist of a similarity-based (KNN-like) classification pipeline, since similar items can explain how the model classifies. We explored this option in Chapter 4 using rational kernels to measure the similarity between two entities. In this chapter, we derive the similarity metric from foundation models aimed to capture semantic textual similarity. Although the particular models that we experiment within our setting are models mostly designed for information retrieval tasks, we can adapt them for classification. An advantage of such a similarity-based pipeline is that we can combine the explanations for each classification of the form "an element x is classified as C because similar elements belong to this class" while incorporating the capabilities of foundation models. This implies relaxing our goal to have models with low resource consumption.

In this chapter, we start measuring the performance gap between of a similarity-based classification pipeline with a more standard neural baseline classifier. We continue with a first attempt to close the measured gap via fine-tuning on the classification task. We conclude proposing an explanation method applicable to the mentioned similarity-based approaches and discuss its potential in the context of trustworthy information retrieval.

5.1 Assessing the Performance Gain on Retail Article Categorization at the Expense of Explainability and Resource Efficiency

In this section, we explicitly assess the "trade-off" triangle presented in Section 1.1, specifically within the context of product cataloging for online shops. The content is adapted from the work presented by Brito, Gupta, et al., 2022, where the author of this dissertation served as the first author. As such, the author was responsible for designing the experimental framework, executing the experiments concerning the topic models and the SBERT models, and drafting the manuscript.

5.1.1 Introduction

The online shopping expansion from the last years has led to a more varying product offer so that an automated product cataloguing process is necessary when the manual maintenance becomes unsustainable. This generally involves solving a multi-class (eventually multi-label) text classification task based on the product descriptions, which is challenging due to the often very skewed label distribution: few categories are dominant, appearing on a majority of products, while many categories hardly appear on few products. This power-law distribution that appears on many NLU tasks can be extreme on some datasets. While the current Transformer-based approaches can achieve astonishing results even in such challenging settings, these mainly consist of very large models, whose complexity may be problematic for real-world implementations. Not only are the implementation-related costs and unexpected high energy consumption generally ignored by machine learning practitioners (García-Martín et al., 2019), but also the current best performing models normally run on specialized hardware (mostly GPUs or TPUs) to obtain results in a feasible time. This requirement limits the access to these models and may also be responsible for a significant carbon footprint (Strubell, Ganesh, and McCallum, 2019). Even ignoring training time, the computational complexity during the inference phase may be an issue when it introduces extra latency on web applications, eventually deteriorating user satisfaction. Furthermore, black-box models are difficult to inspect when issues arise ("why this unexpected label for this product?"). We present exactly such an use case where our client required a low-resource solution where some categories must not be confused (e.g. "sex toys" must not be misclassified as "toys").

We perceive for this particular use case an example of the "trade-off triangle" presented in Section 1.1: we have three aspects that we cannot fully achieve at the same time when choosing a classification model architecture: performance, explainability, and low resource requirements. In this work, we aim to quantify the performance gap between black-box transformer-based methods and more lightweight explainable models. For the former, we deliberately omit to experiment with current state-of-the-art deep networks for extreme multi-label classification. Instead, DistilBERT (Sanh et al., 2019), usually performing slightly worse than its transformer-based competitors while being significantly more compact, can act as a "ceiling model" for the presented alternatives in this work. In parallel, we evaluate a KNN-based classification pipeline trained on a

set of various text representations of different degrees of computational complexity and explainability, including topic models and neural language models. We find this framework explainable since we can trivially show the neighbors (the most similar product descriptions) as explanations for every classified article. We test our pipeline on two retail article categorization datasets of different complexity and compare it with DistilBERT. Although we observe the expected performance gap, the difference is hardly relevant on the simpler dataset while we measure much larger CPU times for DistilBERT. We also consider the explainability aspect, questioning the suitability of huge neural networks as standard option when they need to be run in a production environment.

5.1.2 Related Work

Recent proposals for text classification are generally transformer-based models (Vaswani et al., 2017). In the particular case of multi-label classification, various deep learning models are state of the art (Bhatia et al., 2016). All these models share a lack of explainability. In the broader context of (text) sequence classification, some approaches rely on similarity measures to a set of prototypes (Pluciński, Lango, and Stefanowski, 2021; Hong, Baek, and Tong Wang, 2021; Brito, Georgiev, et al., 2019). Although there is no consensus to measure interpretability in machine learning (Molnar, 2020), we find these approaches as inherently explainable since the computed similarities can serve as explanations for the model decisions. Our presented similarity-based classifier fits to this class of models but we rather focus on evaluating existing representation learning methods within our framework. We also evaluate the runtime-related cost, which only partially correlates to model complexity i.e., to less explainability.

5.1.3 Experiments

Data

German Product Migration (GPM) is an internal dataset used to automate the product cataloging process for a retail online store, where certain run time and explainability requirements were requested. Each article contains a title and a description in German and is assigned a label from a three-level category hierarchy. For our experiments, we focus on the 599 categories belonging to the deepest level, ranging from clothing and accessories to health and personal care. Figure 5.1 shows the very skewed label distribution.

AmazonCat-13K is an extreme multi-label classification dataset provided by Amazon. It consists of product reviews tagged with $\approx 13K$ product categories. The train split has 1.186.239 instances and the test split has 306.782 instances. In Figure 5.2 we provide visualizations of the distribution of the number of labels associated to input texts, the distribution of the number of positive training instances per label and the distribution of the document lengths measured using a DistilBERT-tokenizer.

Methods

Nearest Neighbor

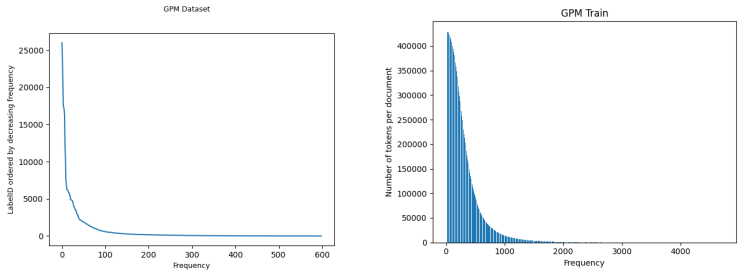


Figure 5.1: Label frequency distribution and tokens per document from GPM.

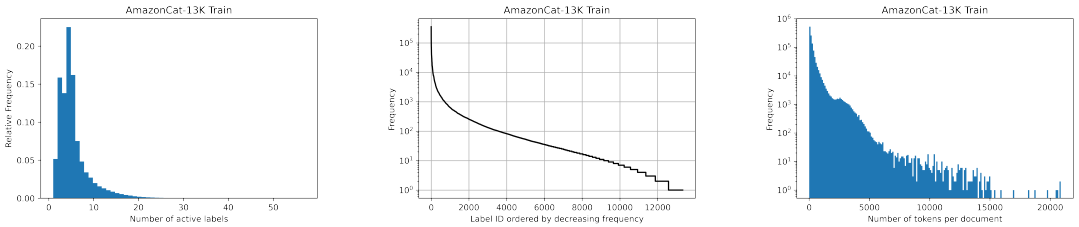


Figure 5.2: Distribution of active labels, label frequency, and number of tokens per document from AmazonCat-13K.

We vectorize each article title and description with a common dimensionality to make them comparable (512 for GPM, 768 for AmazonCat-13K, both determined by the pretrained models we use). Each represented article from the test set obtains the label(s) from the training set article whose vector representation is closest according to cosine distance. The respective vector is derived from the following four methods.

Latent Dirichlet allocation (LDA) (Blei, Ng, and Jordan, 2003). We train a topic model from the word tokens marked as (proper) nouns by the spaCy POS tagger (Honnibal et al., 2020) using the `de_core_web_sm` model for GPM and `en_core_web_sm` for AmazonCat-13K.

Anchored CorEx (Gallagher et al., 2017). We train two topic models with the same preprocessing pipeline as for LDA. The first one is trained in a purely unsupervised fashion. The other by assigning 10 anchor words to start the topic model training. We automatically generate anchor words by finding the words that have the highest mutual information with each label, as proposed by Jagarlamudi, Daumé III, and Udupa, 2012. In the case of the AmazonCat-13K, we restrict this procedure to the 768 most frequent labels.

FastText (Bojanowski et al., 2017). We train a fastText embedding model for each dataset, whose obtained article representations are based on the average of k-gram features.

SentenceBERT (SBERT) (Reimers and Gurevych, 2019). We transform each text with a pretrained model: `distiluse-base-multilingual-cased-v1` for GPM, and `sentence-transformers/all-mpnet-base-v2` for AmazonCat-13K.

ML-KNN We train a ML-KNN model (M.-L. Zhang and Zhou, 2007) on the already

by the previous methods computed vector representations with the scikit-multilearn implementation (Szymański and Kajdanowicz, 2017). For each of the produced representations, we run a 3-fold cross-validation on 100,000 random articles from the training set to determine the number of neighbors k and the smoothing parameter s.

DistilBERT We fine-tune a DistilBERT model. The architecture is given by DistilBERT as the encoder and a linear decoder with fan-out to all labels, activated with sigmoid and loss function given by binary cross-entropy. The forward pass of the complete architecture is given by

$$\begin{aligned} x_{\text{id}}, x_{\text{attn}} &\longleftarrow \text{DistilBERTTok}(x), \\ \text{out} &\longleftarrow \text{Pool}(\text{DistilBERT}_\theta(x_{\text{id}}, x_{\text{attn}})), \\ \text{out} &\longleftarrow \text{GELU}(W_1\text{out} + b_1), \\ \text{out} &\longleftarrow \text{GELU}(W_2\text{out} + b_2), \\ \text{out} &\longleftarrow \text{GELU}(W_3\text{out} + b_3), \\ \text{out} &\longleftarrow \text{BCE}(\sigma(\text{out}), \hat{y}) \end{aligned} \tag{5.1}$$

for input text x and multi-label binarized target $\hat{y} \in \{0,1\}^L$ where L denotes the number of distinct labels. The DistilBERT-tokenizer returns the input-ids and the attention mask $x_{\text{id}}, x_{\text{attn}}$ of x. These are fed into the transformer and the hidden state of the [CLS]-token is pooled. Afterwards the hidden state is fed through 3 linear layers with biases each of which is activated with GELU (Gaussian error linear unit) (Hendrycks and Gimpel, 2016). The final logits are activated with sigmoid σ and the loss function is given by binary cross entropy as usual for multi-label classification problems. The linear layers are initialized with Xavier initialization (Glorot and Bengio, 2010) and have shapes $(768, 2048)$, $(2048, 1024)$ and $(1024, L)$. For training we used smart padding and mixed precision (Benesty, 2020). The idea is to sort train and test examples into batches such that elements in batches have a similar length. Afterwards the batches are padded dynamically which means that each element is padded to the longest length inside the batch or truncated to the tokenizer max length T_{max} if the element is longer than T_{max}. This gives a computational speed-up since it reduces the expected number of [PAD]-tokens during training compared to dynamic padding. This results in faster time until convergence.

Evaluation

Metrics

We evaluate each method applied in the single-label classification setting (GPM dataset) on the F1-score, both micro-averaged and macro-averaged on all the classes. For the multi-label setting (AmazonCat-13K), we include not only the widespread metrics P@k (precision at k) and nDCG@k (normalized discounted cumulative gain at k) with $k \in \{1, 3, 5\}$ but also their propensity-scored variants PSP and PSnDCG to better assess the performance on the less frequent classes as proposed in (Jain, Prabhu, and Varma, 2016), taking the default propensity values $A = 0.55, B = 1.5$. We also measure the training time and prediction time of each method when running on a single CPU.

Table 5.1: Performance on the GPM dataset.

Method	F-1 score	
	Macro avg.	Micro avg.
Nearest Neighbor		
+ LDA	0.65	0.90
+ CorEx	0.69	0.92
+ Anchored CorEx	0.69	0.91
+ SBERT	0.87	0.97
+ FastText	0.66	0.94
DistilBERT	0.92	0.98

Table 5.2: Training and prediction time with AmazonCat-13K on an Intel® Xeon® Gold 6226R CPU @ 2.90GHz.

	CPU time (in min.)	
Method	Training	Prediction
Nearest Neighbor	0.05	156
MLkNN	5,000	930
+ LDA	+ 1,370	+ 25.8
+ (Anchored) CorEx	+ 1,880	+ 5.22
+ SBERT	+ 0[1]	+ 29.2
+ FastText	+ 86	+ 9.4
DistilBERT	72,000[1]	510

Table 5.3: Performance of the evaluated methods on AmazonCat-13K evaluated as in Bhatia et al., 2016. Please note that nDCG@1=P@1 and PSnDCG@1=PSP@1.

Method	P			nDCG		PSP			PSnDCG	
	@1	@3	@5	@3	@5	@1	@3	@5	@3	@5
Nearest Neighbor										
+ LDA	58.81	52.51	42.31	57.84	56.74	31.53	42.03	46.42	39.74	43.60
+ CorEx	56.10	48.47	38.76	54.45	52.94	28.19	36.37	39.96	35.21	38.33
+ Anchored CorEx	55.50	48.25	38.22	53.79	52.24	27.95	36.41	39.53	34.87	37.92
+ SBERT	**75.94**	**67.92**	**55.58**	**74.50**	**73.45**	**41.65**	**55.51**	**62.18**	**52.29**	**57.58**
+ FastText	67.98	61.54	51.18	67.42	67.31	36.81	49.50	55.96	46.63	51.81
MLkNN										
+ LDA	60.93	50.87	39.62	39.62	54.36	31.89	39.86	42.57	38.24	40.93
+ CorEx	56.27	49.09	38.95	54.67	53.19	28.30	36.96	39.96	35.37	40.17
+ Anchored CorEx	55.93	48.52	38.51	54.11	52.62	28.17	36.64	39.84	35.09	38.21
+ SBERT	**80.14**	**71.25**	**57.10**	**78.14**	**75.97**	**42.78**	**56.88**	**62.54**	**53.60**	**58.32**
+ FastText	75.56	65.53	51.73	72.49	69.93	38.94	50.35	54.30	47.95	51.71
DistilBERT	**95.91**	**82.21**	**66.60**	**90.99**	**88.93**	**55.48**	**68.05**	**74.98**	**65.59**	**71.22**

Results

The performance results of the evaluated methods are displayed in Table 5.1 for the GPM dataset and in Table 5.3 for the AmazonCat-13K dataset. As expected, the neural-based methods perform better on both datasets than the topic models. In particular, DistilBERT performs best on all evaluation metrics. However, the performance gap is reduced on the GPM dataset, especially on micro-averaged precision. We also observe that ML-KNN brings some performance improvement compared to the nearest neighbor baseline, although the gain is limited in several metrics, in particular when applied to the least performing topic models. We can see the training and prediction time of the different methods in Table 5.2. As expected, DistilBERT takes the longest CPU time.

[1]We omit the needed time for the pretrained model due the given CPU-only setting.

5.1.4 Discussion

The results seem to confirm the hypothesized triangular trade-off: the most complex model (DistilBERT) outperforms by a large margin the simpler interpretable models (LDA and CorEx), while those in between in terms of explainability (SBERT somewhat more interpretable than DistilBERT due to its similarity-based nature) or regarding computational complexity (FastText) achieve intermediate results. However, the performance gap is notably smaller in the single-label classification setting of the GPM dataset. Since many online shops do not handle datasets as complex as AmazonCat-13K but rather like GPM, we question the convenience of applying the best performing available model by default disregarding significant computational and implementation costs. In some cases, the precision gain separating these models from "cheaper" models is not relevant. The combination of SBERT with a KNN-based classifier may be a good trade-off: it can perform close to the best model while keeping explainable predictions (an article gets a label because similar articles have that label). FastText can also be an option in setups where no GPU is available. Moreover, there are still open options to increase the accuracy of the explainable framework. For instance, the SBERT representations do not incorporate the label information from Amazon-Cat-13k as DistilBERT did during training. Hence, we may easily improve the SBERT-backed model by just fine-tuning it. This is in line with some authors claiming that there is not always necessarily an accuracy-explainability trade-off when standard processes for knowledge discovery are followed (Rudin, 2019). Although using anchor words within anchored CorEx improved no model on any dataset (probably because of the anchor selection method), we value the possibility of selecting anchor words to control how specific topics are built for use cases where sensitive categories must not be confused.

5.1.5 Conclusion and Future Work

We evaluated several text representations on two datasets for retail product categorization within an explainable similarity-based framework. We compared them with a pure neural-based baseline not only on classification performance but also on the required training and prediction time. We additionally discussed the trade-off between obtaining a good performance, having some degree of explainability, and keeping the required computational resources low depending on the application. For future work, we plan to fine-tune an SBERT model on Amazon-Cat-13K by assigning product pairs a similarity score. We also envision a systematic model robustness inspection for specific sensitive labels. Enforcing separations via anchor words may turn anchored CorEx more valuable than our work showed.

5.2 Fine-tuning Sentence Transformers for Similarity Based Multi-label Text Classification

We showed in Section 5.1 how an SBERT model can be used for a similarity-based classification pipeline and achieve close to state-of-the-art results while keeping some degree of explainability. In this section, we aim to close the performance gap between the SBERT-based pipeline and the pure Transformer-based models by fine-tuning the SBERT model on a classification task. This section is based on the results obtained after supervising the Master's student Mahnaz Mirhaj in the scope of the *Lab Development and Application of Data Mining and Learning Systems: Data Science and Big Data* during the winter semester 2022-2023 at the University of Bonn.

5.2.1 Introduction

Transformers are often considered the top-performing models for text classification, as they excel at capturing the contextual relationships between words and phrases within a sentence or document (Devlin et al., 2019). Their powerful capabilities stem from understanding these relationships within the full context of the text. However, as we noted in 5.1, their functioning as a black-box model can make it difficult to interpret how they generate their predictions or encode semantic information in sentences. In this work, we aim to provide a more transparent approach to text classification by using semantic similarity as our model explanation: "a document is assigned specific labels *because* the most similar documents in the training set have them".

We build upon the framework presented by Brito, Gupta, et al., 2022 (also exposed in Section 5.1) and fine-tune an SBERT model based on the idea that documents with similar labels are likely to be related. Using the shared labels of documents, we construct characteristic vectors and compute cosine similarity scores between them as a measure of similarity. With this fine-tuned SBERT model, we generate sentence embeddings and train a KNN model to perform the classification task. We then evaluate our approach and compare our results with those obtained from the original SBERT model. By incorporating label information from the training dataset and using semantic similarity as our model explanation, we aim to achieve a high-performing explainable model.

To this end, we experiment on the AmazonCat13k dataset and evaluate our results using the Precision@k and nDCG@k metrics.

5.2.2 Experiments

Data

We conducted our experiments on the AmazonCat-13k dataset (Bhatia et al., 2016), which consists of product reviews and their associated labels from the "Amazon.com" website. This dataset spans across 13,000 different product categories and contains 1,186,239 instances in the training split and 306,782 instances in the testing split.

Pipeline

We executed the following pipeline to conduct our experiments:

(i) **Preprocessing**: To preprocess the AmazonCat13k dataset, we downsampled the training data to 10% (118,623 documents), selecting them based on the number of shared labels. Since the objective is to incorporate label information in creating text embeddings and many of the documents share 0 labels which leads to 0 cosine similarity. We utilized 1% of the down-sampled documents due to computational resource limitations. We calculated the characteristic vectors and the cosine similarity for each pair of the selected documents.

(ii) **Fine-tuning SBERT**: Using the sentence-transformers Python library (Reimers and Gurevych, 2019), we fine-tuned the SBERT model over 1,000 training documents. We trained the model by feeding it document pairs and their cosine similarity as the similarity score. We used the all-mpnet-base-v2 model as the SBERT model, which maps documents to a 768-dimensional dense vector space. We trained the model in 2 steps using a batch size of 5. We used warm-up steps of 20 and the cosine similarity loss function. No further hyperparameters were tested due to the limited availability of computational resources. With the fine-tuned model, we computed the embeddings over the entire dataset for the train and test split.

(iii) **KNN model**: Due to memory limitations, we randomly chose 3,500,000 training documents and fed their embeddings and labels to a KNN model using the scikit-learn (Pedregosa et al., 2011) implementation for a prediction task. We evaluated the model with over 15,000 validation documents and predicted their labels. In the embedding space, we chose the k documents whose embeddings were closest to the embedding of the validation sample based on cosine similarity. Based on majority voting on the k neighbors predicted label(s), the KNN model assigned label(s) to the validation sample. Based on our evaluation results, we identified 4 as the optimum number of neighbors. Additionally, we chose the KDTree algorithm for the KNN model as it is efficient in memory.

Evaluation

We evaluate our pipeline on the following evaluation metrics for multi-label classification:

- **nDCG@k** (normalized Discounted Cumulative Gain at k), a metric used to evaluate the quality of a ranking algorithm. The nDCG@k score ranges from 0 to 1, with 1 being the best possible score. nDCG@k is based on DCG (Discounted Cumulative Gain) which measures the quality of a ranking algorithm based on the relevance scores of the items in the ranked list. nDCG@k is defined as

$$nDCG@k = \frac{DCG@k}{IDCG@k}, \quad DCG@k = \sum_{i=1}^{k} \frac{2^{rel_i}-1}{log_2(i+1)}, \tag{5.2}$$

  where rel_i is the relevance score of the item at position i and $IDCG@k$ is the ideal DCG at position k. That is the maximum possible DCG@k score for the list, calcu-

Table 5.4: Evaluation Results

Metric	Fine-tuned SBERT Model	Original SBERT Model
P@1	90.56	82.95
P@3	89.11	84.15
P@5	82.68	79.96
nDCG@1	77.08	75.63
nDCG@3	68.89	64.59
nDCG@5	63.51	59.97

lated by sorting the list by relevance score and taking the $DCG@k$ of that sorted list. We used a scikit-learn implementation to compute nDCG@1, nDCG@3, and nDCG@5 to determine how relevant the predicted labels are to the documents.

- **Precision@k** (p@k), a metric used to evaluate the effectiveness of a search or recommendation algorithm by measuring the proportion of relevant items in the top k results. In other words, it measures how many of the items at the top of the list are relevant to the user query or preferences. We sorted the predicted labels based on the most frequent ones. The intersection of the top-k predicted labels and true labels was divided by the total number of predicted labels for each validation document to compute the p@k. The reported p@k is the mean p@k.

Results

Table 5.4 displays the performance outcomes of the evaluated methods. As expected, the KNN classifier which is trained over the fine-tuned model embedding results performs better on all evaluation metrics than the pretrained SBERT model on the AmazonCat13k dataset.

5.2.3 Conclusion and Future Work

In this work, we demonstrated how incorporating label information in the SBERT model can increase its performance and create an explainable model for multi-label classification tasks. Our results demonstrate that fine-tuning the SBERT model on a small subset of training data that includes label information, and subsequently using the resulting embeddings in a KNN model, can lead to improved performance with respect to precision and nDCG metrics. This suggests that there is not necessarily a trade-off between performance and explainability in classification pipelines.

However, due to computational resource limitations, we could only fine-tune the SBERT model on a relatively small number of training samples, and future work could investigate the performance of this approach on larger datasets. Additionally, since the labels in the AmazonCat13k dataset are skewed, we suggest exploring weighting schemas such as TF-IDF for the label information to further enhance the pipeline's performance. Overall, this work highlights the potential benefits of incorporating label information in SBERT models for multi-label classification tasks.

5.3 MaxSimE: Explaining Transformer-based Semantic Similarity via Contextualized Best Matching Token Pairs

As discussed in the previous sections, KNN-based classification pipelines employing contextualized text representations not only deliver competitive performance, but inherently provide an explanation for each classified item. This explanation takes the form of "X is classified as category C because the most similar items are labeled as such". This section is an adaptation of the work presented by Brito and Iser, 2023, where we dig deeper into this kind of explanations and inspect if we can also explain the underlying semantic similarity by finding token pairs that are most similar when computing the similarity between two texts.

5.3.1 Introduction

Modern ranking systems often depend on pre-trained language models to compute representations for queries and documents (Paaß and Giesselbach, 2023). Because of the black-box nature of the large deep neural networks they mostly rely on, these models are not suitable when the user requires some explanation to trust the system or to correct it when its output is erroneous (Beckh et al., 2023). Among the recent Transformer-based (Vaswani et al., 2017) approaches, ColBERT (Khattab and Zaharia, 2020) introduces a late interaction mechanism to a pre-trained BERT model (Devlin et al., 2019). This additional layer is used to calculate a similarity score between a query and a document by matching each token vector representation from the query to the closest document token representations, summing them all into a global similarity score. This sum of similarity scores over query terms is similar to more standard ranking methods such as BM25 (Robertson, Zaragoza, et al., 2009) and we can exploit it to generate explanations about the similarity score. Under the hood, this so-called *MaxSim* operation matches each query token to the most semantically similar document token within their respective contexts. Since we can compute the cosine similarity between any two token representations, we can show the matched tokens by decreasing order of similarity i.e., by decreasing contribution to the global similarity score, so that we can visualize why a retrieved document is (not) similar to the input query. Since the BERT tokens are mostly (sub)words, the matched token pairs can be interpretable terms that are found to be similar.

In this work, we provide examples of where these matches seem informative and discuss the limitations of their interpretability. Additionally, we extend this approach to more 'standard' BERT-based models and compare the resulting explanations to those obtained from ColBERTv2. Our contribution on MaxSim-based Explanations (MaxSimE)[2] is twofold:

(i) We propose an explainability method for Transformer-based semantic similarity, whose fidelity is maximal when applied to models fine-tuned via late interaction such as ColBERTv2 (Santhanam et al., 2022). Visualizing the contextualized best matching tokens can help to confirm a highly ranked document or to hint at some

[2]Source code available on https://github.com/fraunhofer-iais/MaxSimE

model failure e.g., paired tokens wrongly contributing to a high similarity score.

(ii) We intrinsically measure the correctness of MaxSimE taking ColBERTv2 as a proxy for the ground truth to discuss the settings where our explanations are most informative while considering their limitations as well.

5.3.2 Related Work

From the wide spectrum of available explanation methods (Beckh et al., 2023; Burkart and Huber, 2021; Molnar, 2020), feature attribution aims to identify the important features or terms that contribute to a particular result. Among them, Local Interpretable Model-agnostic Explanations (LIME) (Ribeiro, S. Singh, and Guestrin, 2016) is a popular method that has been adapted for information retrieval tasks (J. Singh and Anand, 2019; Verma and Ganguly, 2019). More recent approaches focus on generating explanations that consider not only individual retrieved documents but also the context of the entire search result list to provide more coherent and diverse explanations (Yu, Rahimi, and Allan, 2022). While all these approaches provide post-hoc explanations, whose fidelity to the ranker cannot be guaranteed, we focus instead on an explainable by architecture approach. Formal, Piwowarski, and Clinchant, 2021 report how BERT-based representations implicitly capture term importance and how the ColBERT fine-tuning approach amplifies this effect, improving the retrieval results. Our approach explicitly exploits this fact to generate explanations highlighting the matched terms and their contribution to the similarity score. Some frameworks focus on inspecting ranking models by evaluating on diagnostic datasets to detect global properties of the tested ranking models (MacAvaney et al., 2022; Câmara and Hauff, 2020; Rau and Kamps, 2022). They progress towards a better understanding of why contextualized word embeddings outperform traditional term-based IR methods. Our approach does not aim to analyze model behavior as a whole like them but rather explain a similarity score i.e., to provide local explanations. Calculating semantic similarity based on token embeddings is not a new idea and it has been explored to rank documents (Xiong et al., 2017). However, we do not aim to build a ranking model from the computed similarity scores but to explain existing models instead.

5.3.3 MaxSimE

MaxSimE is a method to generate local explanations for document retrieval systems using language models from which the semantic similarity between two tokens can be measured by the cosine similarity between their vector representations. Its purpose is to provide insights into why a document was retrieved given a query by highlighting the tokens in both the query and the document that contribute the most to their similarity score. We adopt the notation from Santhanam et al., 2022 and define a similarity function $S_{q,d}$ between a query q of N tokens and a document d of M tokens as the summation of query-side MaxSim operations, namely, the maximum cosine similarity between each query token embedding and all document token embeddings (implemented as

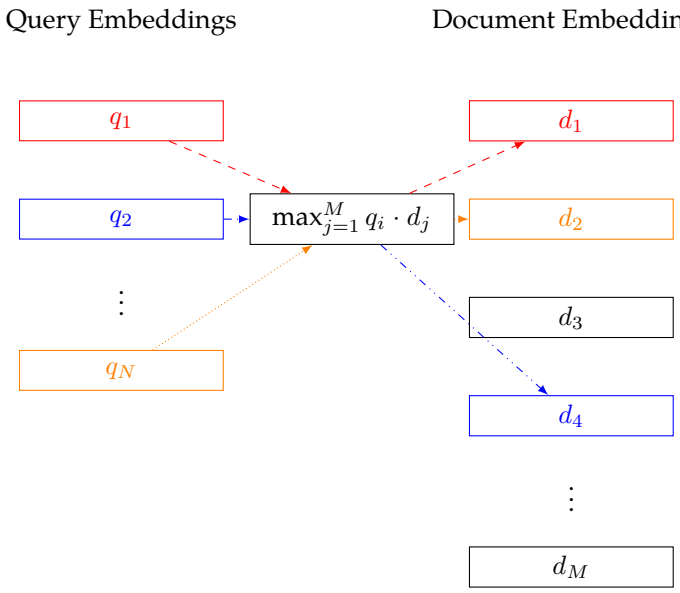


Figure 5.3: Visualization of the MaxSim operation. Each embedding represents a token created by the BERT tokenizer. Given a query q and a document d, for a query embedding q_i, MaxSim selects the closest document embedding d_j. When the represented query token is an interpretable term, this is equivalent to finding the most semantically similar term appearing in d, represented by the document embedding d_j.

dot-products assuming normalized embeddings):

$$S_{q,d} := \sum_{i=1}^{N} \max_{j=1}^{M} Q_i \cdot D_{d_j}^T, \tag{5.3}$$

where Q is a matrix of N vectors encoding q and D a matrix of M vectors encoding d, being each vector an embedding of a token.

We match each query token to the most similar document token (given a context) according to the MaxSim operation, as displayed in Figure 5.3. Formally, given a query embedding q_i, our matching function f_{match} returns the document token embedding d_j with the highest dot product to q_i:

$$f_{match}(q_i) := \arg\max_{d_j} q_i \cdot d_j, \tag{5.4}$$
$$i \in [\![1..N]\!], j \in [\![1..M]\!]$$

Applying our matching function to all embeddings from a query results in a list of token pairs with the highest similarity according to the cosine similarity of their respective embeddings. These token pairs with their respective similarity scores (computed from their dot product) construct an explanation about "why" document d_j was retrieved given q_i as a query.

5.3.4 Experiments

Data

Our experiments are performed on the LoTTE benchmark, a collection of questions and answers sourced from StackExchange. The benchmark covers a wide range of topics, including writing, recreation, science, technology, and lifestyle (Santhanam et al., 2022). To pair documents, we use ColBERTv2 to rank the documents, and we select the top-1-ranked document for each question.

5.3.5 Fully Faithful Explanations from ColBERT-based Models

We apply our approach first to a ColBERTv2 model to generate explanations. The first observed explanations seem to be informative from a qualitative point of view, as seen in the example from Table 5.5. The fidelity of the explanations is maximal because ColBERTv2 scoring is directly reliant on the sum of query side MaxSim scores, and the similarity function has been optimized through fine-tuning, thereby giving more significance to the best matching token pairs. In addition, these explanations come at no cost, since the MaxSim scores for each query token are already computed in the retrieval process. Considering that ColBERTv2 approaches state-of-the-art level according to most of the metrics from the BEIR benchmark for dense retrieval (Thakur et al., 2021), we assume these explanations to be our "gold standard" for further experiments.

Explanations from Other BERT-based Models

We generate explanations with our approach from other BERT-based models that were not fine-tuned with a late interaction mechanism like ColBERT. We aim to confirm if these explanations are trustworthy and we thus compare the resulting explanations with those extracted from ColBERTv2 as in Section 5.3.5, assuming the latter as the reference. As shown in the example from Table 5.5, the matched tokens partially coincide with those obtained from the ColBERTv2 model although the contribution of the token pairs to the similarity score differs to a greater extent. Performance-wise, generating explanations for non-ColBERT architectures involves $N \cdot M$ cosine distance computations (see Equation 5.3).

Evaluation

Despite the absence of ground truth and user feedback, we aim to evaluate the correctness of our explanations extracted from several BERT-based models by comparing them with the ColBERTv2 explanations we generated in Section 5.3.5, which we take as a proxy for a "gold standard". Let T be the number of correctly retrieved document tokens, P the number of retrieved query/token pairs according to the gold standard, and N the number of query tokens. For each query document, we evaluate the following metrics on the Top-1 document retrieved by ColBERTv2:

(i) Token precision: $\frac{T}{N}$

Table 5.5: Matched tokens from the query "Why do kittens love packets?" and first ranked document by the pretrained ColBERTv2 model. MaxSim was performed on ColBERTv2 and S-BERT$_{base}$, sorted by descending similarity score.

Query	ColBERTv2		S-BERT$_{base}$	
Token	Token	Score	Token	Score
why	**because**	0.874	**because**	0.911
kitten	[D]	0.809	cats	0.891
##s	**they**	0.756	**they**	0.874
[CLS]	[CLS]	0.728	[CLS]	0.843
do	which	0.722	to	0.848
love	**love**	0.694	**love**	0.912
packets	boxes	0.485	dart	0.787
?	boxes	0.466	means	0.843

(ii) Matching accuracy: $\frac{P}{N}$

(iii) Spearman's rank correlation of the matching token scores with the gold standard.

Notice that the matching accuracy is a stricter variant of the token precision since the token precision just measures how many of the expected document tokens were retrieved (independent from the query tokens they were matched to), whereas the matching accuracy only counts the matches where the tokens are correct both from the query and the document side. The Spearman's rank correlation is intended to capture the similarity in terms of ranking query tokens.

We compare the explanations from two model architecture classes: Cross-Encoders, which use a regression head to compute the similarity of two input texts directly; and Bi-Encoders, which produce one embedding per document either by Mean/Max Pooling token embeddings or by selecting the `[CLS]` token embedding so that the similarity of two texts is measured by the cosine similarity of the respective embeddings. Bi-Encoders therefore also use a late-interaction mechanism for similarity estimation whereas Cross-Encoders are fully attention-based. We analyze the effect this has on the generated explanations. For Cross-Encoders we choose the MSMARCO pretrained TinyBERT and MiniLM-L6 model, provided by the `sentence-transformers` library (Reimers and Gurevych, 2019). For Bi-Encoders we compare the S-BERT$_{base}$ model with its distilled variant DistilBERT and with the MiniLM-L6 model.

Results

We first analyze the explanations generated by both ColBERTv2 and the Bi-Encoder S-BERT$_{base}$. Table 5.5 shows token matching pairs of both models. Qualitatively, we can observe that both explanations match similar document tokens to the query. Partially these matches coincide between the two models. From Figure 5.4 we can observe a noticeable difference in absolute score values, especially in the ranking of the matching token pairs. In comparison, S-BERT$_{base}$ yields higher scores for query tokens ranked lower by ColBERTv2. Furthermore, the score values produced by ColBERTv2 exhibit a

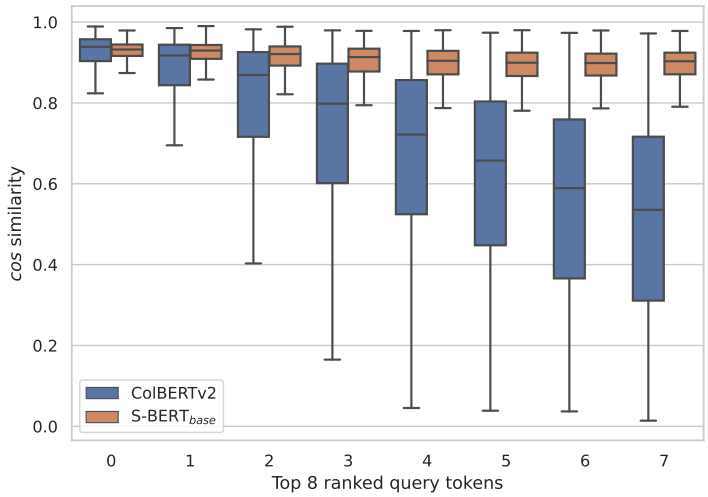


Figure 5.4: Cosine similarity distribution of the top 8 ranked query tokens for each query from the LoTTE dataset.

greater degree of variance, especially for these lower-ranked tokens. We assume that this is due to the fine-tuning of ColBERTv2 token representations with the MaxSim late-interaction mechanism, which forces the model to also perform a fine-grained ranking on the token level.

When evaluating the correctness of our explanations on non-ColBERT models, we observe that token precision is generally high across most models (as displayed in Table 5.6). All metrics have high variance, which suggests that the quality of the explanations is highly dependent on the query sentence. Especially the matching accuracy and ranking of the tokens are inconsistent throughout the dataset. For the smaller model MiniLM-L6, we see that the Bi-Encoder variant provides explanations closer to our gold standard. This could be explained by the fact that the late-interaction mechanism used in sentence transformers (especially with mean pooling) is more similar to the MaxSim operation than the regression head in Cross-Encoders.

Discussion

We observe that, although the non-ColBERT models were not trained using the MaxSim operations, the generated explanations largely align with those of ColBERTv2, as demonstrated by the example in Table 5.5. The similarity between the explanations suggests that they similarly capture term importance, in line with previous white box analysis on ColBERT (Formal, Piwowarski, and Clinchant, 2021). Considering that the ranking performance does differ, we guess that the different similarity value distributions assigned to the matches have a noticeable impact on the global similarity score and thus on the ranked documents. The distributions in Figure 5.4 illustrate how ColBERTv2 weights with significantly higher similarity scores for the most semantic relevant terms than the rest of the tokens, whereas the similarity score difference among embeddings coming

Table 5.6: Similarity of explanations from BERT-based models to our ColBERTv2 gold standard measured by token precision (TP), match accuracy (MA), and Spearman's rank correlation (SR).

Model	TP	MA	SR
	Bi-Encoders		
S-BERT$_{base}$	0.730 ± 0.153	$\mathbf{0.471} \pm 0.213$	0.427 ± 0.380
DistilBERT	0.740 ± 0.163	0.444 ± 0.212	0.349 ± 0.386
MiniLM-L6	0.664 ± 0.149	0.411 ± 0.200	$\mathbf{0.473} \pm 0.376$
	Cross-Encoders		
TinyBERT	$\mathbf{0.749} \pm 0.158$	0.446 ± 0.204	0.391 ± 0.343
MiniLM-L6	0.387 ± 0.233	0.307 ± 0.192	0.270 ± 0.284

from BERT$_{base}$ is clearly less differentiated. Despite this, the high token precision (displayed in Table 5.6) implies that non-ColBERT models frequently match the same tokens as ColBERT.

Although we demonstrated how we can generate meaningful explanations for both ColBERT and other BERT-based models using the MaxSim operation, we acknowledge two main limitations of our approach: the limited faithfulness to the model for non-ColBERT architectures and the limited interpretability of some explanations because of the contribution of the `[MASK]` tokens to the similarity score.

First, our explanations from non-late-interaction-based models i.e., Bi- and Cross-encoders (Reimers and Gurevych, 2019), cannot guarantee faithfulness to their respective ranking models because their computed similarity usually comes from either a regression head or from the cosine similarity of `[CLS]` or mean pooled embeddings. Although Cross-Encoder models may achieve better evaluation scores, their computational cost is much higher, becoming impractical for most setups. Hence, we favor late interaction models for ranking not only because of their efficiency on ranking tasks but also because we can extract fully faithful explanations from the underlying language model. Second, (Khattab and Zaharia, 2020) use `[MASK]` tokens within the ColBERTv2 model for query expansion. These non-interpretable tokens are also included in the late-interaction scoring mechanism, leading to best-matching token pairs that cannot be explained in a meaningful way. Depending on the length of the query, these `[MASK]` tokens make up for up to 62% of the final score of the retrieved document. Nonetheless, Lassance et al., 2021 show that these special tokens can be safely removed without affecting model performance in a significant way.

Finally, we could only evaluate the correctness of the explanations extracted from the different models by comparing them to our ColBERTv2 gold standard, which we consider confirmed when they correlate but we cannot discard otherwise. Other explainability aspects such as plausibility (Jacovi and Goldberg, 2020) are yet to be assessed as well. Despite the limitations, we find our first exploratory results promising and we hope to motivate more work towards trustworthy information retrieval.

5.3.6 Conclusion and Future Work

We leveraged the MaxSim operation from the ColBERT approach to generate explanations for the documents retrieved by the ranking system, based on the most relevant document tokens that match those of the query. We also demonstrated that our method can be applied to other BERT-based models, although we cannot guarantee its fidelity for those models. The correlation between the explanations generated by the different models confirms that our proposed method can provide insights into the underlying model, and can be used as a proxy to evaluate explanation correctness. Our presented method enables "explanations for free" i.e., without needing to learn any explanation model, from similarity functions constructed upon BERT-based language models. Our proposed approach may have applications beyond information retrieval e.g., text classification use cases where unfaithful explanations from black-box models are not acceptable and where a similarity-based classifier can be used without a dramatic performance loss compared to the best-performing black-box deep learning model (Brito, Gupta, et al., 2022); or even less related areas where Transformer-based models can deal with a concept of semantic similarity such as computer vision (C. Zhang, Liwicki, and Cipolla, 2022).

In future work, we aim to systematically compare the ranking performance of different BERT-based models with our evaluation results, including additional evaluation criteria and benchmark datasets where ColBERTv2 was not fine-tuned. From a more applied perspective, we also plan to apply our approach to domain-specific settings e.g., information retrieval on legal texts to support lawyers finding previous similar legal cases when facing a new one, which is an opportunity to assess the plausibility of our explanations.

Chapter 6

Conclusion

This chapter provides a summary of the thesis, discusses the main results obtained, and outlines possible research directions to be pursued based on this work.

6.1 Summary

The thesis began by presenting the triangular trade-off that ML practitioners face: achieving maximum accuracy, explainability, and resource efficiency simultaneously is challenging.

Chapter 3 displayed three use cases where we used evaluation as a mean to achieve trustworthy systems: semantic segmentation for autonomous driving systems, language modelling for sentiment analysis, and automatic text summarization. Each use case utilized different evaluation methods to enhance trust in the system. For the first, an approach to validate simulation-based testing was proposed while its limitations, mainly dependend on the quality of the available simulations, where also exposed. The experiments testing German word embeddings on sentiment analysis provided some implementation insights and showed the inadequacy of some intrisic evaluations. For the latter use case on automatic text summarization, an ensemble of evaluation metrics was combined to automatically evaluate text summaries in German, obtaining a higher correlation with human judgement than well-established evaluation metrics.

Chapter 4 introduced the RatVec framework as an alternative to pure neural methods for reducing computational requirements and increasing model interpretability. The RatVec framework generates vector representations using rational kernels within KPCA, which were later used for classification tasks solved by a KNN classifier, making it resource-efficient and explainable in the sense that elements are classified according to similar elements from the training set, where the similarity is given by the applied kernel function. Competitive performance was achieved on various tasks, such as spelling correction, DNA splice junction detection, and protein family classification. However, the approach was limited to problems closely related to aligning finite domain sequences, where sequence similarity was measured using string kernels such as edit distance-based kernel functions. To overcome these limitations, the combination of the explainability of the RatVec framework with the rich text representations provided

by foundation models was later explored in Chapter 5. The performance gap between a pipeline consisting of SBERT text representations and a KNN classifier, and a pure BERT-based classifier was measured on two datasets in the context of the product cataloguing, showing that the proposed similarity-based pipeline has potential to reach competitive performance. Encouraged by the results, the SBERT model was fine-tuned for multi-label text classification to narrow the performance gap. Additionally, an explanation method for the classifications delivered by this pipeline was proposed, which not only showed the most similar texts as explanations but also highlighted the tokens contributing most to the measured semantic similarity.

6.2 Discussion

Significant contributions were made towards enhancing trustworthiness in the use cases presented in Chapter 3. For semantic segmentation in street scenes, the ability to identify whether simulation-based testing performs similarly to testing on real data was demonstrated. This transferability of testing results is essential when obtaining sufficient real data is challenging. The key point of this use case was to detect model failures on simulated data that would have occurred with equivalent real data. The results showed a high correlation in mIoU when comparing the results between testing on simulated and real data. However, a rule learner could easily identify whether errors were made on synthetic or real data. The performance of the rule learner dropped significantly on synthetic data with higher quality, suggesting that validating simulation-based testing may be feasible when the quality of the simulations is sufficiently high.

The fine-tuning of German word embeddings for sentiment analysis, a use case that provided valuable insights at the time of its publication, also gave us the opportunity to critique the use of intrinsic evaluation metrics. However, with the emergence of foundation models, the current relevance of these results is somewhat limited.

In the automatic text summarization use case, an evaluation metric that correlated better with human judgment than well-established metrics like ROUGE was developed. This metric was also to some extent interpretable, as it consisted of an ensemble of different summarization aspects. This result is significant because summarization methods are often considered ”better” if they outperform previous models on established metrics, which continue to be standard despite criticism.

We extended previous work on rational kernels to create an explainable network comprising a representation learning part based on rational kernels and a KNN classifier. State-of-the-art performance was achieved on several tasks with a significantly less complex model and an explainable-by-design pipeline. The RatVec framework excelled in classification tasks where data points of the same class tended to cluster based on string similarity. This was evident in tasks such as spelling correction and protein family classification, where sequence similarity played a crucial role. However, the approach did not perform as well in tasks where there was no obvious translation to a sequence similarity problem, such as converting sentences to a PoS sequence for double-having detection in historical Dutch texts. Furthermore, the experiments were limited to predefined string kernels without learning the underlying similarity function, which may

lead to better results.

Regarding the similarity-based pipelines using representations encoding semantic similarity extracted from foundation models, we measured the performance gap between these pipelines and plain foundation models fine-tuned for the task. Despite limitations in the experiments, such as the generalization of results (we tested only on two datasets for the product cataloguing use case) and the specificity of the tested methods, the findings were consistent with previous experiences. The most complex language model achieved the best results, but its lack of interpretability and computational costs may not outweigh the performance advantage. Motivated by this, a SBERT model was fine-tuned to capture semantic text similarity for multi-label task classification. The presented approach represented a preliminary step towards narrowing the performance gap between our proposed similarity-based pipeline and competing models.

Further contributions were made to enhance the explainability of similarity-based classification pipelines by generating more fine-grained explanations with the MaxSimE approach. These explanations not only pointed to the most similar neighbors for classification but also identified the contextualized tokens that contributed the most to the measured semantic similarity. The explanations appeared informative, particularly for the ColBERT case. However, their usefulness needs to be confirmed through user experience studies and their generalization to diverse datasets. Furthermore, the highlighted tokens are mostly words obtained from a BERT tokenizer. For some use cases that involve longer documents and longer queries, less granular explanations may be more suitable e.g., by matching at sentence or paragraph level.

6.3 Outlook

The research presented in this thesis can be further pursued in several interesting directions. Firstly, a valuable line of investigation would be to explore the boundaries of the RatVec framework by learning rational kernels from available data, as opposed to relying solely on predefined string kernels. As rational kernels are weighted transducers, finding an approach to determine the suitable number of states and optimal weighted transitions would be necessary. Additionally, quantifying the computational resource savings offered by RatVec-based solutions and assessing their adjustability by varying the rational kernel and embedding dimensionality would be essential to assess their plausibility for production environments. Also in the context of resource-awareness, the performance gap analysis in Section 5.1 only measured CPU time as a proxy for resource consumption, which is a limited metric. Memory consumption and the utilization of specialized hardware, such as GPUs, should also studied as well.

Closing the performance gap between similarity-based frameworks and pure neural models can be pursued through more exhaustive fine-tuning of the semantic similarity metric learning with a SBERT model. However, this would inherently increase complexity and computational resource consumption, necessitating an assessment of whether the trade-off is justified for each specific use case.

Regarding the advancements made in explainability, while the generated MaxSimE explanations appear promising, their usefulness and generalization to diverse datasets

need to be confirmed through user studies. Moreover, the scope of the matched tokens apart from the word level should be further investigated to figure out if less granular explanations would be more suitable for each particular use case. Anyhow, conducting user studies would provide quantitative evaluations of the extent to which the proposed explanation method increases trust in the system and saves user time.

Acronyms

AD Autonomous Driving.
AI Artificial Intelligence.

BERT Bidirectional Encoder Representations from Transformers.

DNN Deep Neural Network.

KNN *k*-Nearest Neighbors.
KPCA Kernel Principal Component Analysis.

LCS Longest Common Subsequence.

mIoU mean Intersection over Union.
ML Machine Learning.

NLU Natural Language Understanding.

PCA Principal Component Analysis.
PoS Part-of-Speech.

SBERT Sentence-BERT.
SVM Support Vector Machine.

List of Figures

List of Tables

Bibliography

Akbik, Alan, Duncan Blythe, and Roland Vollgraf (Aug. 2018). "Contextual String Embeddings for Sequence Labeling." In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, pp. 1638–1649. URL: https://www.aclweb.org/anthology/C18-1139.

Arora, Sanjeev, Rong Ge, and Ankur Moitra (2012). "Learning Topic Models – Going beyond SVD." In: *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pp. 1–10. DOI: 10.1109/FOCS.2012.49.

Asgari, Ehsaneddin and Mohammad RK Mofrad (2015). "Continuous distributed representation of biological sequences for deep proteomics and genomics." In: *PloS one* 10.11, e0141287.

Banerjee, Satanjeev and Alon Lavie (2005). "METEOR: An automatic metric for MT evaluation with improved correlation with human judgments." In: *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pp. 65–72.

Barbier, M. et al. (2019). "Validation of Perception and Decision-Making Systems for Autonomous Driving via Statistical Model Checking." In: *Intelligent Vehicles Symposium (IV)*, pp. 252–259. DOI: 10.1109/IVS.2019.8813793.

Baroni, Marco et al. (Sept. 2009). "The WaCky Wide Web: A Collection of Very Large Linguistically Processed Web-Crawled Corpora." In: *Language Resources and Evaluation* 43.3, pp. 209–226.

Beckh, Katharina et al. (2023). "Harnessing Prior Knowledge for Explainable Machine Learning: An Overview." In: *First IEEE Conference on Secure and Trustworthy Machine Learning*. URL: https://openreview.net/forum?id=1KE7TlU4bOt.

Beeksma, Merijn et al. (2018). "Detecting and correcting spelling errors in high-quality Dutch Wikipedia text." In: *Computational Linguistics in the Netherlands Journal* 8, pp. 122–137.

Benesty, Michaël (2020). "Divide Hugging Face Transformers training time by 2 or more with dynamic padding and uniform length batching." In: *https://towardsdatascience.com/divide-hugging-face-transformers-training-time-by-2-or-more-21bf7129db9q-21bf7129db9e*. URL: https://towardsdatascience.com/divide-hugging-face-transformers-training-time-by-2-or-more-21bf7129db9q-21bf7129db9e.

Berg, C., J. P. R. Christensen, and P. Ressel (1984). *Harmonic Analysis on Semigroups*. Berlin: Springer.

Bhatia, K. et al. (2016). *The extreme classification repository: Multi-label datasets and code*. URL: http://manikvarma.org/downloads/XC/XMLRepository.html.

Biesner, David et al. (2020). "Hybrid Ensemble Predictor as Quality Metric for German Text Summarization: Fraunhofer IAIS at GermEval 2020 Task 3." In: *5th Swiss Text Analytics Conference (SwissText) & 16th Conference on Natural Language Processing (KONVENS)*.

Blei, David M, Andrew Y Ng, and Michael I Jordan (2003). "Latent dirichlet allocation." In: *the Journal of machine Learning research* 3, pp. 993–1022.

Bojanowski, Piotr et al. (2017). "Enriching Word Vectors with Subword Information." In: *Transactions of the Association for Computational Linguistics* 5, pp. 135–146. DOI: 10.1162/tacl_a_00051. URL: https://aclanthology.org/Q17-1010.

Bosch, Antal van den et al. (2007). "An efficient memory-based morphosyntactic tagger and parser for Dutch." In: *Selected Papers of the 17th Computational Linguistics in the Netherlands Meeting*. Netherlands Graduate School of Linguistics, pp. 99–114.

Braiek, Houssem Ben and Foutse Khomh (2020). "On Testing Machine Learning Programs." In: *Journal of Systems and Software* 164, p. 110542. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2020.110542.

Brants, Sabine et al. (2004). "TIGER: Linguistic interpretation of a German corpus." In: *Research on Language and Computation* 2.4, pp. 597–620.

Breiman, L. (2001). "Random Forests." In: *Machine Learning* 45.1, pp. 5–32.

Brito, Eduardo, Bogdan Georgiev, et al. (2019). "RatVec: A General Approach for Low-dimensional Distributed Vector Representations via Rational Kernels." In: *LWDA, KDML workshop*.

Brito, Eduardo, Vishwani Gupta, et al. (2022). "Assessing the Performance Gain on Retail Article Categorization at the Expense of Explainability and Resource Efficiency." In: *KI 2022: Advances in Artificial Intelligence*. Ed. by Ralph Bergmann et al. Cham: Springer International Publishing, pp. 45–52. ISBN: 978-3-031-15791-2.

Brito, Eduardo and Henri Iser (2023). "MaxSimE: Explaining Transformer-based Semantic Similarity via Contextualized Best Matching Token Pairs." In: *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '23. Taipei, Taiwan: Association for Computing Machinery. ISBN: 978-1-4503-9408-6/23/07. DOI: 10.1145/3539618.3592017. URL: https://doi.org/10.1145/3539618.3592017.

Brito, Eduardo, Max Lübbering, et al. (2019). "Towards Supervised Extractive Text Summarization via RNN-based Sequence Classification." In: *arXiv preprint arXiv:1911.06121*. URL: https://arxiv.org/abs/1911.06121.

Brito, Eduardo, Rafet Sifa, and Christian Bauckhage (2017). "KPCA Embeddings: an Unsupervised Approach to Learn Vector Representations of Finite Domain Sequences." In: *LWDA, KDML workshop*.

— (2019). "Two Attempts to Predict Author Gender in Cross-Genre Settings in Dutch." In: *Shared Task on Cross-Genre Gender Prediction in Dutch at CLIN29*.

Brito, Eduardo, Rafet Sifa, Christian Bauckhage, et al. (2019). "A Hybrid AI Tool to Extract Key Performance Indicators from Financial Reports for Benchmarking." In: *Proceedings of the ACM Symposium on Document Engineering 2019*.

Brito, Eduardo, Rafet Sifa, Kostadin Cvejoski, et al. (2017). "Towards German word embeddings: A use case with predictive sentiment analysis." In: *Data Science–Analytics and Applications*. Springer Vieweg, Wiesbaden, pp. 59–62.

Buciluǎ, Cristian, Rich Caruana, and Alexandru Niculescu-Mizil (2006). "Model Compression." In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '06. Philadelphia, PA, USA: Association for Computing Machinery, pp. 535–541. ISBN: 1595933395. DOI: 10.1145/1150402.1150464. URL: https://doi.org/10.1145/1150402.1150464.

Burkart, Nadia and Marco F Huber (2021). "A survey on the explainability of supervised machine learning." In: *Journal of Artificial Intelligence Research* 70, pp. 245–317.

Bussler, A. et al. (2020). "Application of Evolutionary Algorithms and Criticality Metrics for the Verification and Validation of Automated Driving Systems at Urban Intersections." In: *Intelligent Vehicles Symposium (IV)*, pp. 128–135.

Câmara, Arthur and Claudia Hauff (2020). "Diagnosing BERT with Retrieval Heuristics." In: *Advances in Information Retrieval*. Ed. by Joemon M. Jose et al. Cham: Springer International Publishing, pp. 605–618. ISBN: 978-3-030-45439-5.

Chen, Dongdong et al. (2017). "Coherent Online Video Style Transfer." In: *International Conference on Computer Vision (ICCV)*.

Clauwaert, Jim and Willem Waegeman (2019). "Novel transformer networks for improved sequence labeling in genomics." In: *bioRxiv*. DOI: 10.1101/836163. eprint: https://www.biorxiv.org/content/early/2019/11/13/836163.full.pdf. URL: https://www.biorxiv.org/content/early/2019/11/13/836163.

Cordts, Marius et al. (2016). "The Cityscapes Dataset for Semantic Urban Scene Understanding." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Cortes, Corinna, Patrick Haffner, and Mehryar Mohri (Dec. 2004). "Rational Kernels: Theory and Algorithms." In: *Journal of Machine Learning Research* 5, pp. 1035–1062. ISSN: 1532-4435.

Cotterell, Ryan and Hinrich Schütze (2015). "Morphological Word-Embeddings." In: *HLT-NAACL*, pp. 1287–1292.

Cristianini, Nello, John Shawe-Taylor, and Huma Lodhi (2002). "Latent semantic kernels." In: *Journal of Intelligent Information Systems* 18.2-3, pp. 127–152.

Dalke, Andrew et al. (2009). "Biopython: freely available Python tools for computational molecular biology and bioinformatics." In: *Bioinformatics* 25.11, pp. 1422–1423.

Devaranjan, Jeevan, Amlan Kar, and Sanja Fidler (2020). "Meta-Sim2: Unsupervised Learning of Scene Structure for Synthetic Data Generation." In: *European Conference on Computer Vision (ECCV)*, pp. 715–733. ISBN: 978-3-030-58520-4.

Devlin, Jacob et al. (June 2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

Dosovitskiy, Alexey et al. (2017). "CARLA: An Open Urban Driving Simulator." In: *1st Annual Conference on Robot Learning*, pp. 1–16.

Dumais, Susan T (2004). "Latent semantic analysis." In: *Annual Review of Information Science and Technology (ARIST)* 38, pp. 189–230.

Eynde, Frank van, Jakub Zavrel, and Walter Daelemans (2000). "Part of Speech Tagging and Lemmatisation for the Spoken Dutch Corpus." In: *Proceedings of LREC 2000*, pp. 1427–1434.

Faaß, Gertrud and Kerstin Eckart (2013). "SdeWaC–a corpus of parsable sentences from the web." In: *Language processing and knowledge in the Web*. Springer, pp. 61–68.

Faruqui, Manaal et al. (2016). "Problems With Evaluation of Word Embeddings Using Word Similarity Tasks." In: *Proceedings of the 1st Workshop on Evaluating Vector Space Representations for NLP*.

Formal, Thibault, Benjamin Piwowarski, and Stéphane Clinchant (2021). "A White Box Analysis of ColBERT." In: *Advances in Information Retrieval*. Ed. by Djoerd Hiemstra et al. Cham: Springer International Publishing, pp. 257–263. ISBN: 978-3-030-72240-1.

Friedman, J. H. (2001). "Greedy Function Approximation: A Gradient Boosting Machine." In: *Annals of Statistics* 29.5, pp. 1189–1232.

Gaidon, Adrien et al. (2016). "Virtual Worlds as Proxy for Multi-Object Tracking Analysis." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Gallagher, Ryan J. et al. (2017). "Anchored Correlation Explanation: Topic Modeling with Minimal Domain Knowledge." In: *Transactions of the Association for Computational Linguistics* 5, pp. 529–542. DOI: 10.1162/tacl_a_00078. URL: https://aclanthology.org/Q17-1037.

García-Martín, Eva et al. (2019). "Estimation of energy consumption in machine learning." In: *Journal of Parallel and Distributed Computing* 134, pp. 75–88.

Geiger, A., P. Lenz, and R. Urtasun (2012). "Are we Ready for Autonomous Driving? The KITTI Vision Benchmark Suite." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3354–3361. DOI: 10.1109/CVPR.2012.6248074.

Giuliano, Claudio (2009). "Fine-grained classification of named entities exploiting latent semantic kernels." In: *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pp. 201–209.

Glorot, Xavier and Yoshua Bengio (13–15 May 2010). "Understanding the difficulty of training deep feedforward neural networks." In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, pp. 249–256. URL: https://proceedings.mlr.press/v9/glorot10a.html.

Goldstein, Alex et al. (2015). "Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation." In: *Journal of Computational and Graphical Statistics* 24.1, pp. 44–65.

Grootendorst, Maarten (2020). *BERTopic: Leveraging BERT and c-TF-IDF to create easily interpretable topics.* Version v0.9.4. DOI: 10.5281/zenodo.4381785. URL: https://doi.org/10.5281/zenodo.4381785.

Grover, Aditya and Jure Leskovec (2016). "node2vec: Scalable feature learning for networks." In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 855–864.

Gurevych, Iryna (2005). "Using the structure of a conceptual network in computing semantic relatedness." In: *International Conference on Natural Language Processing*. Springer, pp. 767–778.

Haasdonk, Bernard (2005). "Feature Space Interpretation of SVMs with Indefinite Kernels." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.4, pp. 482–492. ISSN: 0162-8828.

Haq, F. U. et al. (2020). "Comparing Offline and Online Testing of Deep Neural Networks: An Autonomous Car Case Study." In: *International Conference on Software Testing, Validation and Verification (ICST)*, pp. 85–95. DOI: 10.1109/ICST46399.2020.00019.

Harris, Zellig S (1954). "Distributional structure." In: *Word* 10.2-3, pp. 146–162.

He, Kaiming et al. (2017). "Mask R-CNN." In: *International Conference on Computer Vision (ICCV)*, pp. 2961–2969.

Hendrycks, Dan and Kevin Gimpel (2016). "Gaussian Error Linear Units (GELUs)." In: *http://arxiv.org/abs/1606.08415v3*.

Henikoff, S and J G Henikoff (Dec. 1991). "Automated assembly of protein blocks for database searching." In: *Nucleic Acids Research* 19.23, pp. 6565–72. ISSN: 0305-1048.

— (Nov. 1992). "Amino acid substitution matrices from protein blocks." In: *PNAS* 89.22, pp. 10915–9.

Hillebrand, Lars et al. (2021). "Interpretable Topic Extraction and Word Embedding Learning Using Non-Negative Tensor DEDICOM." In: *Machine Learning and Knowledge Extraction* 3.1, pp. 123–167.

Hinton, Geoffrey, Oriol Vinyals, and Jeffrey Dean (2015). "Distilling the Knowledge in a Neural Network." In: *NIPS Deep Learning and Representation Learning Workshop*. URL: http://arxiv.org/abs/1503.02531.

Hong, Dat, Stephen S. Baek, and Tong Wang (2021). *Interpretable Sequence Classification Via Prototype Trajectory*. arXiv: 2007.01777 [cs.LG].

Honnibal, Matthew et al. (2020). "spaCy: Industrial-strength Natural Language Processing in Python." In: DOI: 10.5281/zenodo.1212303.

Huang, Lang et al. (2019). "Interlaced Sparse Self-Attention for Semantic Segmentation." In: *arXiv preprint arXiv:1907.12273*.

Huang, Xiaolin et al. (2016). "Indefinite kernels in least squares support vector machines and principal component analysis." In: *Applied and Computational Harmonic Analysis* 43.1, pp. 162–172.

Ingraham, John et al. (2019). "Generative Models for Graph-Based Protein Design." In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., pp. 15820–15831. URL: http://papers.nips.cc/paper/9711-generative-models-for-graph-based-protein-design.pdf.

Ishii, Naohiro et al. (2006). "Text classification by combining grouping, LSA and kNN." In: *Proc. IEEE/ACIS Int. Conf. on Computer and Information Science*. IEEE, pp. 148–154.

Isola, Phillip et al. (2017). "Image-To-Image Translation With Conditional Adversarial Networks." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jacovi, Alon and Yoav Goldberg (2020). *Towards Faithfully Interpretable NLP Systems: How should we define and evaluate faithfulness?* DOI: 10.48550/ARXIV.2004.03685. URL: https://arxiv.org/abs/2004.03685.

Jagarlamudi, Jagadeesh, Hal Daumé III, and Raghavendra Udupa (Apr. 2012). "Incorporating Lexical Priors into Topic Models." In: *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Avignon, France: Association for Computational Linguistics, pp. 204–213. URL: https://aclanthology.org/E12-1021.

Jain, Himanshu, Yashoteja Prabhu, and Manik Varma (2016). "Extreme Multi-Label Loss Functions for Recommendation, Tagging, Ranking & Other Missing Label Applications." In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: Association for Computing Machinery, pp. 935–944. ISBN: 9781450342322. DOI: 10.1145/2939672.2939756. URL: https://doi.org/10.1145/2939672.2939756.

Jurdzinski, Grzegorz et al. (2016). "Word Embeddings for Morphologically Complex Languages." In: *Schedae Informaticae* 25, pp. 127–138.

Kar, Amlan et al. (2019). "Meta-Sim: Learning to Generate Synthetic Datasets." In: *International Conference on Computer Vision (ICCV)*.

Khattab, Omar and Matei Zaharia (2020). "ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT." In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '20. Virtual Event, China: Association for Computing Machinery, pp. 39–48. ISBN: 9781450380164. DOI: 10.1145/3397271.3401075. URL: https://doi.org/10.1145/3397271.3401075.

Kim, Been et al. (Oct. 2018). "Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV)." In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 2668–2677. URL: https://proceedings.mlr.press/v80/kim18d.html.

Kondrak, Grzegorz (2005). "N-gram similarity and distance." In: *Int. Symp. on String Processing and Information Retrieval*. Springer, pp. 115–126.

Kouw, Wouter M. and Marco Loog (2019). "A Review of Domain Adaptation without Target Labels." In: *arXiv preprint arXiv:1901.05335*. eprint: 1901.05335 (cs.LG).

Lassance, Carlos et al. (Dec. 2021). *A Study on Token Pruning for ColBERT*. arXiv:2112.06540 [cs]. DOI: 10.48550/arXiv.2112.06540. URL: http://arxiv.org/abs/2112.06540 (visited on 02/09/2023).

Lau, Jey Han and Timothy Baldwin (2016). "An empirical evaluation of doc2vec with practical insights into document embedding generation." In: *Proceedings of the 1st Workshop on Representation Learning for NLP*, pp. 78–86.

Le, Quoc V and Tomas Mikolov (2014). "Distributed Representations of Sentences and Documents." In: *ICML*. Vol. 14, pp. 1188–1196.

Lee, Daniel D and H Sebastian Seung (2001). "Algorithms for non-negative matrix factorization." In: *Advances in neural information processing systems*, pp. 556–562.

Leslie, Christina, Eleazar Eskin, and William Stafford Noble (2002). "The spectrum kernel: a string kernel for SVM protein classification." In: *Proceedings of the 7th Pacific Symposium on Biocomputing*. World Scientific, pp. 566–575.

Levenshtein, VI (1966). "Binary Codes Capable of Correcting Deletions, Insertions and Reversals." In: *Soviet Physics Doklady* 10, p. 707.

Levy, Omer, Yoav Goldberg, and Ido Dagan (2015). "Improving distributional similarity with lessons learned from word embeddings." In: *Transactions of the Association for Computational Linguistics* 3, pp. 211–225.

Lichman, M. (2013). *UCI Machine Learning Repository*. URL: http://archive.ics.uci.edu/ml.

Lim, Bryan et al. (2019). "Temporal fusion transformers for interpretable multi-horizon time series forecasting." In: *arXiv preprint arXiv:1912.09363*.

Lin, Chin-Yew (July 2004). "ROUGE: A Package for Automatic Evaluation of Summaries." In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, pp. 74–81. URL: https://www.aclweb.org/anthology/W04-1013.

Lin, Tsung-Yi et al. (2014). "Microsoft COCO: Common objects in context." In: *European Conference on Computer Vision (ECCV)*, pp. 740–755.

Lodhi, Huma et al. (2002). "Text classification using string kernels." In: *Journal of Machine Learning Research* 2.Feb, pp. 419–444.

Lundberg, Scott M and Su-In Lee (2017). "A unified approach to interpreting model predictions." In: *Advances in neural information processing systems* 30.

MacAvaney, Sean et al. (2022). "ABNIRML: Analyzing the Behavior of Neural IR Models." In: *Transactions of the Association for Computational Linguistics* 10, pp. 224–239. DOI: 10.1162/tacl_a_00457. URL: https://aclanthology.org/2022.tacl-1.13.

Mei, Wang and Weihong Deng (2018). "Deep Visual Domain Adaptation: A Survey." In: *Neurocomputing*. DOI: 10.1016/j.neucom.2018.05.083.

Mikolov, Tomas, Kai Chen, et al. (2013). "Efficient estimation of word representations in vector space." In: *arXiv preprint arXiv:1301.3781*.

Mikolov, Tomas, Ilya Sutskever, et al. (2013). "Distributed Representations of Words and Phrases and their Compositionality." In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges et al. Curran Associates, Inc., pp. 3111–3119. URL: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Molnar, Christoph (2020). *Interpretable machine learning*. URL: https://christophm.github.io/interpretable-ml-book/.

Needleman, Saul B. and Christian D. Wunsch (1970). "A general method applicable to the search for similarities in the amino acid sequence of two proteins." In: *Journal of Molecular Biology* 48.3, pp. 443–453.

Neurohr, C. et al. (2020). "Fundamental Considerations around Scenario-Based Testing for Automated Driving." In: *Intelligent Vehicles Symposium (IV)*, pp. 121–127. DOI: 10.1109/IV47402.2020.9304823.

Noordewier, Michiel O, Geoffrey G Towell, and Jude W Shavlik (1991). "Training knowledge-based neural networks to recognize genes in DNA sequences." In: *Advances in Neural Information Processing Systems*, pp. 530–536.

Paaß, Gerhard and Sven Giesselbach (2023). "Foundation Models for Natural Language Processing–Pre-trained Language Models Integrating Media." In: *arXiv preprint arXiv:2302.08575*.

Paatero, Pentti and Unto Tapper (1994). "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values." In: *Environmetrics* 5.2, pp. 111–126.

Papineni, Kishore et al. (2002). "BLEU: a method for automatic evaluation of machine translation." In: *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pp. 311–318.

Paranjape, I. et al. (2020). "A Modular Architecture for Procedural Generation of Towns, Intersections and Scenarios for Testing Autonomous Vehicles." In: *Intelligent Vehicles Symposium (IV)*, pp. 162–168. DOI: 10.1109/IV47402.2020.9304625.

Pedregosa, Fabian et al. (2011). "Scikit-learn: Machine learning in Python." In: *the Journal of machine Learning research* 12, pp. 2825–2830.

Pielka, Maren et al. (2020). "Fraunhofer IAIS at FinCausal 2020, Tasks 1 & 2: Using Ensemble Methods and Sequence Tagging to Detect Causality in Financial Documents." In: *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pp. 64–68.

Pluciński, Kamil, Mateusz Lango, and Jerzy Stefanowski (2021). "Prototypical Convolutional Neural Network for a Phrase-Based Explanation of Sentiment Classification." In: *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. Ed. by Michael Kamp et al. Cham: Springer International Publishing, pp. 457–472. ISBN: 978-3-030-93736-2.

Rau, David and Jaap Kamps (2022). "How Different Are Pre-Trained Transformers For Text Ranking?" In: *Advances in Information Retrieval: 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway, April 10–14, 2022, Proceedings, Part II*. Stavanger, Norway: Springer-Verlag, pp. 207–214. ISBN: 978-3-030-99738-0. DOI: 10.1007/978-3-030-99739-7_24. URL: https://doi.org/10.1007/978-3-030-99739-7_24.

Reimers, Nils and Iryna Gurevych (Nov. 2019). "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks." In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 3982–3992. DOI: 10.18653/v1/D19-1410. URL: https://www.aclweb.org/anthology/D19-1410.

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin (2016). *Model-Agnostic Interpretability of Machine Learning*. DOI: 10.48550/ARXIV.1606.05386. URL: https://arxiv.org/abs/1606.05386.

Rives, Alexander et al. (2019). "Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences." In: *bioRxiv*, p. 622803.

Robertson, Stephen, Hugo Zaragoza, et al. (2009). "The probabilistic relevance framework: BM25 and beyond." In: *Foundations and Trends® in Information Retrieval* 3.4, pp. 333–389.

Rosenzweig, Julia et al. (2021). "Validation of simulation-based testing: Bypassing domain shift with label-to-image synthesis." In: *2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)*. IEEE, pp. 182–189.

Rottmann, M. et al. (2020). "Prediction Error Meta Classification in Semantic Segmentation: Detection via Aggregated Dispersion Measures of Softmax Probabilities." In: *International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9. DOI: 10.1109/IJCNN48605.2020.9206659.

Rudin, Cynthia (2019). "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead." In: *Nature Machine Intelligence* 1.5, pp. 206–215.

Rudolph, Alexander, Stefan Voget, and Jürgen Mottok (2018). "A Consistent Safety Case Argumentation for Artificial Intelligence in Safety Related Automotive Systems." In: *European Congress on Embedded Real Time Software and Systems (ERTS)*.

Rueden, Laura von, Sebastian Mayer, Katharina Beckh, et al. (2020). "Informed Machine Learning - A Taxonomy and Survey of Integrating Knowledge into Learning Systems." In: *arXiv preprint arXiv:1903.12394v2*.

Rueden, Laura von, Sebastian Mayer, Rafet Sifa, et al. (2020). "Combining Machine Learning and Simulation to a Hybrid Modelling Approach: Current and Future Directions." In: *International Symposium on Intelligent Data Analysis*. Springer, pp. 548–560.

Sang, Erik F and Fien De Meulder (2003). "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition." In: *arXiv preprint cs/0306050*.

Sänger, Mario et al. (May 2016). "SCARE – The Sentiment Corpus of App Reviews with Fine-grained Annotations in German." In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. Portorož, Slovenia: European Language Resources Association (ELRA).

Sanh, Victor et al. (2019). "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter." In: *arXiv preprint arXiv:1910.01108*.

Santhanam, Keshav et al. (July 2022). "ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction." In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Seattle, United States: Association for Computational Linguistics, pp. 3715–3734. DOI: 10.18653/v1/2022.naacl-main.272. URL: https://aclanthology.org/2022.naacl-main.272.

Schölkopf, B. and AJ. Smola (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning. Cambridge, MA, USA: MIT Press, p. 644.

Schölkopf, Bernhard, Alexander Smola, and Klaus-Robert Müller (1998). "Nonlinear component analysis as a kernel eigenvalue problem." In: *Neural computation* 10.5, pp. 1299–1319.

Schraagen, Marijn, Joanna Wall, and Eduardo Brito (2020). "The CLIN30 shared task: Have-doubling in historical varieties of Dutch." In: *Computational Linguistics in the Netherlands Journal* 10, pp. 161–178.

Schwalbe, Gesina and Martin Schels (2020). "A Survey on Methods for the Safety Assurance of Machine Learning Based Systems." In: *European Congress on Embedded Real Time Software and Systems (ERTS)*.

Shrikumar, Avanti, Peyton Greenside, and Anshul Kundaje (June 2017). "Learning Important Features Through Propagating Activation Differences." In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 3145–3153. URL: https://proceedings.mlr.press/v70/shrikumar17a.html.

Shrivastava, Ashish et al. (2017). "Learning From Simulated and Unsupervised Images Through Adversarial Training." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Singh, Jaspreet and Avishek Anand (Jan. 2019). "EXS: Explainable Search Using Local Model Agnostic Interpretability." en. In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. Melbourne VIC Australia: ACM, pp. 770–773. ISBN: 978-1-4503-5940-5. DOI: 10.1145/3289600.3290620. URL: https://dl.acm.org/doi/10.1145/3289600.3290620 (visited on 02/10/2023).

Smith, T.F. and M.S. Waterman (1981). "Identification of common molecular subsequences." In: *Journal of Molecular Biology* 147.1, pp. 195–197.

Spearman, Charles (1904). "The proof and measurement of association between two things." In: *The American journal of psychology* 15.1, pp. 72–101.

Strubell, Emma, Ananya Ganesh, and Andrew McCallum (July 2019). "Energy and Policy Considerations for Deep Learning in NLP." In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 3645–3650. DOI: 10.18653/v1/P19-1355. URL: https://www.aclweb.org/anthology/P19-1355.

Szymański, P. and T. Kajdanowicz (Feb. 2017). "A scikit-based Python environment for performing multi-label classification." In: *ArXiv e-prints*. arXiv: 1702.01460 [cs.LG].

Thakur, Nandan et al. (2021). "BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models." In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. URL: https://openreview.net/forum?id=wCu6T5xFjeJ.

Tijhuis, LJ (2014). "Context-Based Spelling Correction for the Dutch Language: Applied on spelling errors extracted from the Dutch Wikipedia revision history." In.

Tjong Kim Sang, Erik et al. (2017). "The CLIN27 Shared Task: Translating Historical Text to Contemporary Language for Improving Automatic Linguistic Annotation." In: *Computational Linguistics in The Netherlands Journal*, pp. 53–64.

Turian, Joseph, Lev Ratinov, and Yoshua Bengio (2010). "Word representations: A simple and general method for semi-supervised learning." In: *48th Annual Meeting of the Association for Computational Linguistics*. Ed. by Association for Computational Linguistics, pp. 384–394.

Vaswani, Ashish et al. (2017). "Attention is All you Need." In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., pp. 5998–6008. URL: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Verma, Manisha and Debasis Ganguly (2019). "LIRME: Locally Interpretable Ranking Model Explanation." In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR'19. Paris, France: Association for Computing Machinery, pp. 1281–1284. ISBN: 9781450361729. DOI: 10.1145/3331184.3331377. URL: https://doi.org/10.1145/3331184.3331377.

Wachter, Sandra, Brent Mittelstadt, and Chris Russell (2017). "Counterfactual explanations without opening the black box: Automated decisions and the GDPR." In: *Harv. JL & Tech.* 31, p. 841.

Wagner, S. et al. (2019). "Towards Cross-Verification and Use of Simulation in the Assessment of Automated Driving." In: *Intelligent Vehicles Symposium (IV)*, pp. 1589–1596. DOI: 10.1109/IVS.2019.8814268.

Wang, J. et al. (2020). "Deep High-Resolution Representation Learning for Visual Recognition." In: *Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1.

Wang, T. et al. (2018). "High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8798–8807. DOI: 10.1109/CVPR.2018.00917.

Wang, Ting-Chun et al. (2018). "Video-to-Video Synthesis." In: *Advances in Neural Information Processing Systems*. Curran Associates, Inc., pp. 1144–1156.

Willers, Oliver et al. (2020). "Safety Concerns and Mitigation Approaches Regarding the Use of Deep Learning in Safety-Critical Perception Tasks." In: *Computer Safety, Reliability, and Security. SAFECOMP 2020 Workshops*. Ed. by António Casimiro et al. Cham: Springer International Publishing, pp. 336–350. ISBN: 978-3-030-55583-2.

Wolf, Thomas et al. (2019). "Transformers: State-of-the-art Natural Language Processing." In: *arXiv preprint arXiv:1910.03771*. URL: https://arxiv.org/abs/1910.03771.

Xie, Ning et al. (2020). "Explainable deep learning: A field guide for the uninitiated." In: *arXiv preprint arXiv:2004.14545*.

Xiong, Chenyan et al. (2017). "End-to-End Neural Ad-Hoc Ranking with Kernel Pooling." In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '17. Shinjuku, Tokyo, Japan: Association for Computing Machinery, pp. 55–64. ISBN: 9781450350228. DOI: 10.1145/3077136.3080809. URL: https://doi.org/10.1145/3077136.3080809.

Yu, Puxuan, Razieh Rahimi, and James Allan (2022). "Towards Explainable Search Results: A Listwise Explanation Generator." In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '22. Madrid, Spain: Association for Computing Machinery, pp. 669–680. ISBN: 9781450387323. DOI: 10.1145/3477495.3532067. URL: https://doi.org/10.1145/3477495.3532067.

Yuan, Yuhui, Xilin Chen, and Jingdong Wang (2020). "Object-Contextual Representations for Semantic Segmentation." In: *arXiv preprint arXiv:1909.11065*.

Yuan, Yuhui and Jingdong Wang (2018). "Ocnet: Object Context Network for Scene Parsing." In: *arXiv preprint arXiv:1809.00916*.

Yuan, Yuhui, Jingyi Xie, et al. (2020). "SegFix: Model-Agnostic Boundary Refinement for Segmentation." In: *arXiv preprint arXiv:2007.04269*.

Zesch, Torsten and Iryna Gurevych (2006). "Automatically creating datasets for measures of semantic relatedness." In: *Proceedings of the Workshop on Linguistic Distances*. Association for Computational Linguistics, pp. 16–24.

Zhang, Chao, Stephan Liwicki, and Roberto Cipolla (2022). "Beyond the CLS Token: Image Reranking using Pretrained Vision Transformers." In: *33rd British Machine Vision Conference 2022, BMVC 2022, London, UK, November 21-24, 2022*. BMVA Press. URL: https://bmvc2022.mpi-inf.mpg.de/0080.pdf.

Zhang, J. M. et al. (2020). "Machine Learning Testing: Survey, Landscapes and Horizons." In: *Transactions on Software Engineering*. DOI: 10.1109/TSE.2019.2962027.

Zhang, Min-Ling and Zhi-Hua Zhou (2007). "ML-KNN: A lazy learning approach to multi-label learning." In: *Pattern recognition* 40.7, pp. 2038–2048.