Deep Representation Learning for Financial Document Analytics

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

von
Lars Patrick Hillebrand
aus
Troisdorf

Bonn, 2025

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

Gutachter/Betreuer:	Prof. Dr. Christian Bauckhage
Gutachter:	Prof. Dr. Rafet Sifa
Tag der Promotion:	16. Juli 2025
Erscheinungsjahr:	2025

Abstract

In this thesis, we leverage machine learning (ML) methods primarily based on deep neural networks to drastically reduce the manual work of financial analysts, investors, auditors, and other stakeholders by automating key steps in their analysis of financial disclosure documents. A core challenge in this context is transforming highly unstructured and inherently discrete textual data into meaningful numerical representations that ML models can effectively interpret. We refer to this automated conversion process as *representation learning* and remark that the learned representations must encode the text's syntactic, semantic, and contextual structure.

We develop novel methodologies utilizing deep representation learning to improve the efficiency and quality of several financial document analysis tasks. First, we introduce an approach for the joint extraction, linking, and consistency checking of Key Performance Indicators (KPIs) from corporate disclosure reports. By fine-tuning a bidirectional text encoder neural network with classification heads for named entity recognition and relation extraction, we efficiently extract KPIs and predict their interrelationships. Building upon this, we enhance the detection of numerical inconsistencies between semantically equivalent KPIs using contrastive learning techniques. This includes joint sentence and table encoding and a contrastive autoencoder classification module, along with a filtering mechanism employing cross-attention to handle data imbalance from numerous unrelated KPI pairs.

To assist auditors in aligning regulatory requirements with relevant sections of financial reports, we introduce a context-aware recommender system designed to retrieve the most pertinent text passages in sustainability reports. The system utilizes a Transformer-based encoding module with a non-linear multi-label classification head, trained end-to-end. Recognizing the limitations of processing paragraphs in isolation, we propose a novel pre-training methodology called Pointer-Guided Segment Ordering, which enhances the model's ability to generate contextually rich paragraph embeddings by understanding narrative flow and inter-paragraph relationships.

Addressing the dynamic nature of accounting standards, we propose a flexible compliance check methodology using Large Language Models (LLMs). We combine a fine-tuned semantic text matching model with an LLM-based re-ranking module, enabling zero-shot matching between financial reports and potentially unseen legal requirements. We further integrate a compliance verification component that employs zero- and few-shot learning with prompting techniques like chain-of-thought to assess compliance with disclosure requirements from international accounting standards.

Lastly, we develop a specialized LLM-powered chatbot with an optimized Retrieval-Augmented Generation pipeline to support compliance with Risk Management and Quality (R&Q) standards. By integrating hybrid search techniques and relevance boosting, the system enhances retrieval accuracy and provides precise and contextually appropriate answers to queries related to R&Q standards, aiding employees in accessing and interpreting complex regulatory information.

Acknowledgements

This PhD journey has been both challenging and rewarding, and I am grateful to those who supported me along the way.

First and foremost, I extend my deepest gratitude to my supervisors, Prof. Dr. Christian Bauckhage and Prof. Dr. Rafet Sifa. From the very beginning, Christian granted me the freedom to explore my areas of interest without any restriction. I am equally thankful to Rafet. His mentorship guided me through both my PhD journey and my industrial pursuits at Fraunhofer. With Rafet, the glass is always half full, never half empty. His positivity in our scientific discussions not only enriched this thesis but also made the overall process very enjoyable.

To my co-authors and colleagues, your collaborations were invaluable. Your feedback and the enjoyable times we shared significantly enhanced the impact of my work.

I also want to thank my friends, whose constant question of "So, when are you finished?" served as a helpful reminder to speed up. Your persistent inquiries kept me motivated and ensured I didn't lose sight of the finish line.

To my wife, Hannah, thank you for your unwavering support and understanding. You endured long nights and weekends when our couple time was limited, always showing patience when the demands of this PhD took precedence.

Lastly, to my parents, Uschi and Egbert, your belief in my abilities has been a constant source of strength. Thank you for encouraging me to pursue my goals without adding pressure. I appreciate how you avoided burdening me with technical support problems during this time, recognizing that focusing on my PhD was more important.

To everyone who has been part of this journey, thank you for making it an experience worth undertaking.

Contents

CHAPTER 1

Introduction

In today's interconnected and rapidly evolving financial landscape, corporations are required to generate and publish vast amounts of financial documents like annual reports, sustainability disclosures, and regulatory filings that are critical for informing investors, regulators, and the public. These complex documents offer detailed insights into a company's financial health, operational strategies, and future prospects. However, the task of meticulously analyzing them to ensure accuracy, compliance with rigorous accounting standards, and transparency is formidable. Auditors are confronted with the enormous challenge of navigating through extensive narratives, dense numerical data, and intricate regulatory requirements.

This traditional auditing process is not only labor-intensive and time-consuming but also prone to human error, which can lead to major financial scandals and reputational damage, such as the Wirecard collapse in 2020 [1], and contribute to systemic risks in the global economy, as highlighted by the financial oversight issues during the 2008 financial crisis [2].

Recent advancements in artificial intelligence (AI) and machine learning (ML) present transformative opportunities to address these challenges. Specifically, deep representation learning [3] has emerged as a powerful foundation for modern ML systems that automate and refine the analysis of complex textual and numerical data inherent in financial documents. Hence, this thesis explores the naturally arising question:

> *How can we leverage deep representation learning to improve the efficiency, accuracy, and reliability of financial document analytics?*

By investigating this question, we aim to enhance auditing processes, reduce the potential for errors, and ultimately contribute to greater transparency and trust in financial reporting.

To set the stage, we start by discussing the significance of representation learning in modern ML and its transformative impact on various domains. We then delve into the difficulties of auditing financial documents, highlighting the complexities and intricacies that auditors face in analyzing financial reports. Subsequently, we introduce the key contributions of this thesis, which focus on advancing representation learning techniques to address the challenges in financial document analytics.

1.1 The Significance of Representation Learning

Representation learning is a fundamental aspect of modern ML that involves the automatic discovery and extraction of features from raw data that capture their underlying patterns and latent structures as part of the learning process [4]. Transforming raw data into meaningful numerical representations is essential for the ML model's performance in solving supervised or unsupervised learning tasks. For instance, in image classification, analyzing each pixel individually is impractical. Instead, a holistic representation that encapsulates both local and global patterns is necessary to understand the image's overall content. Similarly, in natural language processing (NLP), understanding a sentence requires more than just analyzing its individual characters. It necessitates representations that capture the semantics of words within their broader context.

Unlike images and audio, where pixels and waveforms inherently have numeric representations that can be directly fed into ML algorithms expecting vector inputs, text is composed of discrete symbols, characters, and words, without immediate numerical equivalents. This discrete nature of text poses significant challenges in representation learning, as the raw textual data must be transformed into numerical vector representations that effectively capture syntactic structure and semantic meaning, especially in the light of word ambiguity, polysemy, and context dependence.

A real-life example that illustrates this importance of data representation is the process of performing long division. As noted by Goodfellow et al. [5], when dividing 210 by 6 using Arabic numerals, the task is straightforward due to the place value system and the structured method of long division. However, if the same division is attempted using Roman numerals (CCX divided by VI), the process becomes significantly more complex and cumbersome. This example highlights how the choice of representation can dramatically affect the ease and efficiency of problem-solving. Just as the right numerical representation simplifies arithmetic operations, appropriate data representations in ML can substantially enhance the model's ability to process and understand information.

In the early days of ML, models often relied on manual feature engineering, where domain experts crafted input features believed to be relevant for specific tasks. This process was labor-intensive and limited by human intuition, often failing to capture complex non-linear patterns inherent in high-dimensional data. Representation learning addresses these limitations by enabling models to automatically learn hierarchical features from data, thus reducing the need for manual intervention and potentially uncovering more abstract and powerful representations [3].

A significant turning point in representation learning was the advent of deep learning, which utilizes deep neural networks to recognize patterns at multiple levels of abstraction. Foundational algorithms like backpropagation [6–8], developed and popularized in the 1970s and 1980s, provided the mathematical underpinnings for training multi-layer neural networks. However, practical applications were initially constrained by limited computational resources and insufficient data, which hindered the training of large-scale models.

The rise of big data and advancements in computational hardware, particularly the development of powerful graphical processing units (GPUs), overcame these limitations. Large-scale datasets like ImageNet [9] supplied the necessary amount of data for training deep neural networks, while GPUs offered the computational power to handle complex matrix and tensor

calculations efficiently. This convergence yielded unprecedented progress across various domains. In computer vision, deep learning has revolutionized image classification [10], object detection [11], and image generation [12, 13]. In speech recognition, it has enabled more accurate interpretation of human speech [14]. Moreover, deep reinforcement learning has achieved remarkable feats, such as AlphaGo's victory over the world champion in Go [15] and advancements in protein folding [16], a long-standing problem in biology.

In NLP, representation learning has been particularly transformative considering that the vast flexibility of language can be used to formulate and subsequently solve virtually any problem that humans can communicate. Leveraging vast amounts of textual data and recent advances in self-supervised and transfer learning, powerful large language models (LLMs) like OpenAI's GPT (Generative Pre-trained Transformer) [17, 18] have demonstrated expert-level capabilities in a range of applications, from machine translation [19] and text summarization [20] to highly complex tasks involving mathematical and legal reasoning [21, 22].

In this thesis, we focus on the evolution of representation learning in the context of NLP with applications in the financial auditing domain. In the following, we introduce the challenges of analyzing financial documents in the context of auditing and tie them to our contributions in textual representation learning.

1.2 Challenges of Auditing Financial Documents

Auditing financial documents is a critical function in the financial industry, ensuring that an organization's financial information is accurate, reliable, and compliant with applicable laws and regulations [23]. By conducting comprehensive reviews and evaluations, independent auditors uphold transparency and integrity in financial reporting, which is essential for stakeholders such as investors, regulators, and the public. Despite its importance, the auditing process faces significant challenges due to the inherent complexity of long financial documents, the intricacies of accounting standards, and the specialized language used in finance.

Financial documents such as annual reports and sustainability disclosures include detailed narratives and extensive numerical tables. They provide crucial insights into a company's financial condition, operational results, strategic direction, and potential risks. An annual financial report typically comprises several key components: the financial statements (including the income statement, balance sheet, and cash flow statement), notes to the financial statements offering additional context and explanations, and the Management's Discussion and Analysis, where leadership discusses financial results, operational achievements, future prospects, and difficulties.

Auditors are tasked with verifying that every relevant requirement from the accounting standards is appropriately reflected in the financial report. Accounting standards such as the German Handelsgesetzbuch[1] (HGB) [24], International Financial Reporting Standards (IFRS) [25], and U.S. Generally Accepted Accounting Principles (GAAP) [26] provide detailed frameworks outlining the required content and presentation of financial reports. They function as extensive checklists, with each item specifying a legal or regulatory requirement that the report must address. However, financial reports are often organized in ways that do not directly align with the structure of these standards. The layout and presentation can vary

[1] German Commercial Code.

greatly between organizations, which complicates auditors' efforts to locate and verify relevant information within the documents [27–29].

Furthermore, auditors must meticulously extract and verify quantitative information to ensure that reported financial figures are accurate and comply with relevant standards [30, 31]. This process involves not only checking the calculations within the financial statements but also tracing figures back to their source documents and, when necessary, checking them against external evidence. Such diligence is critical because errors or misstatements in numerical data can significantly impact stakeholders' decisions and may lead to legal penalties for the organization.

In addition to verifying individual figures, auditors face the challenge of ensuring consistency throughout the document [32]. Discrepancies between different sections, such as conflicting financial figures or inconsistent descriptions of company policies and risks, can undermine the credibility of the entire report. Ensuring consistency requires auditors to thoroughly compare narratives, data tables, and disclosures, which is a meticulous and time-consuming task.

Another significant issue is conducting a comprehensive completeness and compliance check. Auditors must verify that all material requirements from the accounting standards are sufficiently covered and that the report fully complies with these standards [33]. This involves not only confirming the presence of required disclosures but also evaluating the adequacy and accuracy of the information provided. Given the substantial length and complexity of both the standards and the financial reports, the manual process of ensuring completeness and compliance is very cumbersome.

To address these obstacles, this thesis presents several contributions to the field of representation learning and its application to financial document analysis, aiming to assist auditors and enhance the auditing process:

1. **Joint Extraction and Linking of Key Performance Indicators (KPIs).** Financial reports often present KPIs in diverse formats and contexts, which makes their manual extraction and analysis challenging and cumbersome. To tackle this, we develop *KPI-BERT*, a method that jointly extracts and links KPIs from financial documents, leveraging Transformer-based text encoders [19, 34] in conjunction with novel Named Entity Recognition (NER) and Relation Extraction (RE) techniques. This approach, detailed in Chapter 5, increases the efficiency of retrieving quantitative information, and thus, facilitates the automation of subsequent auditing tasks such as consistency checks.

2. **Verifying Numerical Consistency via Contrastive Learning.** Ensuring that semantically equivalent KPIs maintain consistent numerical values throughout financial documents is crucial for reliable financial analysis. We introduce *KPI-Check*, an advanced system that automatically identifies and cross-verifies semantically equivalent KPIs within real-world financial documents to detect numerical inconsistencies. By utilizing contrastive learning to create robust embeddings for KPIs and employing modules such as joint sentence and table encoding, we improve the detection of discrepancies. This contribution is discussed in Chapter 6.

3. **Enhancing Semantic Text Matching with Paragraph-Level Context.** Aligning regulatory requirements with relevant sections of financial reports is challenging due to the complexity of the documents and the lack of structural alignment. To assist auditors in efficiently

matching disclosure requirements to pertinent text passages, we develop *sustain.AI*, a context-aware recommender system for sustainability reports. Furthermore, we address the limitation of processing paragraphs in isolation by introducing a novel pre-training method called *Pointer-Guided Segment Ordering (SO)*. This method enhances language models' ability to understand narrative flow and inter-paragraph relationships, producing contextually rich paragraph embeddings. By incorporating paragraph-level contextual awareness, our approach improves semantic matching between standards and report sections, aiding auditors in locating and verifying compliance-related information. These contributions are detailed in Chapters 7 and 8.

4. **Flexible Compliance Verification Using LLMs and Retrieval-Augmented Generation (RAG).** The dynamic and evolving nature of accounting and compliance standards poses difficulties for traditional models that require retraining when standards change. To provide a scalable and adaptable solution, we develop flexible approaches leveraging advanced text matching techniques and LLMs. We introduce *ZeroShotALI*, which combines BERT-based [34] text matching with an LLM-based re-ranking and compliance check module utilizing GPT-4 [18], enabling zero-shot matching and subsequent compliance assessment between financial reports and unseen legal requirements without retraining (Chapter 9). Additionally, to enhance compliance verification and support in contexts such as risk and quality assurance, we employ RAG techniques [35]. We dynamically integrate external knowledge sources into the LLM answer generation process, which improves the trustworthiness and reliance of model responses, mitigating the hallucination of spurious information (Chapter 10).

By integrating these advanced representation learning methods, we address the key challenges in financial document auditing. Our contributions improve the efficiency and accuracy of the document analytics process, thereby reducing the potential for human error and allowing auditors to focus their expertise on advanced areas that require more nuanced judgment. Ultimately, this work enhances the transparency and reliability of financial reporting, fostering greater trust among stakeholders.

1.3 Thesis Outline

This thesis is structured into three parts.

Part I lays the groundwork by exploring the fundamentals of representation learning in NLP, which are crucial for the applications discussed later. Chapter 2 presents the core concepts and techniques that underpin this thesis, e.g., text classification, text matching, tokenization, and topic modeling. We trace the evolution of word embeddings from static, frequency-based methods to sophisticated semantic embeddings that capture word meanings. In Chapter 3, we bridge the gap between the previously discussed interpretable frequency-based embeddings and non-interpretable semantic embeddings. We introduce a novel DEDICOM-based matrix factorization approach to jointly learn interpretable word embeddings and distinct topics from unlabeled text corpora. Chapter 4 examines language modeling and contextual word embeddings, introducing advanced deep neural network architectures capable of modeling sequential data dependencies. We explore self-supervised learning in the context of language

modeling and present transfer learning paradigms, like supervised fine-tuning and zero-shot learning. Additionally, we discuss retrieval augmented generation, a technique that enhances the trustworthiness and actuality of generative models by integrating relevant external knowledge via retrieval mechanisms. These methodologies form the foundation for the advanced applications covered in the subsequent chapters.

Part II centers on sequential text classification and semantic text matching to enhance the consistency of financial documents within the auditing domain. In Chapter 5, we propose novel methods for the joint extraction and linking of Key Performance Indicators (KPIs) in financial reports, which facilitate subsequent consistency checks. Chapter 6 introduces contrastive learning techniques for numerical consistency checks, improving the semantic matching of KPIs within financial documents.

Part III explores the application of LLMs for ensuring financial document compliance. Chapter 7 presents methods for semantic text classification in the auditing of sustainability reports. In Chapter 8, we enhance LLMs with paragraph-level awareness, refining paragraph-level text representations for sequential text classification. Chapter 9 investigates zero-shot learning for automatic compliance verification and showcases the potential of LLMs in assessing a financial report's compliance and completeness with respect to rigorous accounting standards. Finally, in Chapter 10, we utilize RAG for risk and quality assurance, which illustrates how the combination of hybrid (full-text and vector-based) retrieval mechanisms with LLMs can increase answer performance and reliance for complex user queries.

1.4 Publications

This thesis is based on the following publications (in order of appearance):

1. L. Hillebrand, D. Biesner, C. Bauckhage, and R. Sifa, "Interpretable Topic Extraction and Word Embedding Learning Using Row-Stochastic DEDICOM," *Proc. CD-MAKE*, 2020, DOI: 10.1007/978-3-030-57321-8_22 [36]

2. L. Hillebrand, D. Biesner, C. Bauckhage, and R. Sifa, *Interpretable Topic Extraction and Word Embedding Learning Using Non-Negative Tensor DEDICOM*, Machine Learning and Knowledge Extraction (2021), DOI: 10.3390/make3010007 [37]

3. L. Hillebrand, T. Deußer, T. Dilmaghani, B. Kliem, R. Loitz, C. Bauckhage, and R. Sifa, "KPI-BERT: A Joint Named Entity Recognition and Relation Extraction Model for Financial Reports," *Proc. ICPR*, 2022, DOI: 10.1109/ICPR56361.2022.9956191 [30]

4. L. Hillebrand, T. Deußer, T. Dilmaghani, B. Kliem, R. Loitz, C. Bauckhage, and R. Sifa, "Towards automating Numerical Consistency Checks in Financial Reports," *Proc. BigData*, 2022, DOI: 10.1109/BigData55660.2022.10020308 [31]

5. L. Hillebrand, M. Pielka, D. Leonhard, T. Deußer, T. Dilmaghani, B. Kliem, R. Loitz, M. Morad, C. Temath, T. Bell, R. Stenzel, and R. Sifa, "sustain.AI: a Recommender System to analyze Sustainability Reports," *Proc. ICAIL*, 2023, DOI: 10.1145/3594536.3595131 [29]

6. L. Hillebrand, P. Pradhan, C. Bauckhage, and R. Sifa, "Pointer-Guided Pre-Training: Infusing Large Language Models with Paragraph-Level Contextual Awareness," *Proc. ECML PKDD*, 2024, DOI: 10.1007/978-3-031-70359-1_23 [38]

7. L. Hillebrand, A. Berger, T. Deußer, T. Dilmaghani, M. Khaled, B. Kliem, R. Loitz, M. Pielka, D. Leonhard, C. Bauckhage, et al., "Improving Zero-Shot Text Matching for Financial Auditing with Large Language Models," *Proc. DocEng*, 2023, DOI: 10.1145/3573128.3609344 [39]

8. A. Berger, L. Hillebrand, D. Leonhard, T. Deußer, T. B. F. De Oliveira, T. Dilmaghani, M. Khaled, B. Kliem, R. Loitz, C. Bauckhage, et al., "Towards automated regulatory compliance verification in financial auditing with large language models," *Proc. BigData*, 2023, DOI: 10.1109/BigData59044.2023.10386518 [33]

9. L. Hillebrand, A. Berger, D. Uedelhoven, D. Berghaus, U. Warning, T. Dilmaghani, B. Kliem, T. Schmid, R. Loitz, and R. Sifa, "Advancing Risk and Quality Assurance: A RAG Chatbot for Improved Regulatory Compliance," *Proc. BigData*, 2024, DOI: 10.1109/BigData62323.2024.10825431 [40]

Part I

Foundation of Representation Learning for Document Analysis

CHAPTER 2

Fundamentals of Classification and Embeddings

This chapter provides a comprehensive exploration of the foundational concepts underpinning word embeddings and text classification, which are essential for understanding the more advanced topics discussed in subsequent chapters. We begin by examining text classification, a pivotal task in NLP that involves the categorization of text data into predefined classes. This task serves as a fundamental building block for a wide array of supervised NLP applications, including translation, question answering, sentiment analysis, summarization, and information extraction. Many complex NLP tasks can be distilled into text classification problems, for instance, text generation can be conceptualized as a classification task where the model predicts the next word in a sequence.

Subsequently, we delve into topic modeling, a technique designed to uncover latent topics within a collection of documents. Topic modeling is a crucial unsupervised learning method that aids in understanding the underlying themes in a corpus of text data. Unlike text classification, topic modeling, akin to clustering, operates without the need for labeled data, providing insights into the structure of text data without predefined categories.

A unique challenge in processing text data, as opposed to image or audio data, is its inherently discrete nature. While image data comprises continuous pixel values that can be directly utilized by ML models, text data necessitates transformation into a numerical format. This transformation involves tokenization and the application of word embeddings, which are vital for capturing the semantic and syntactic nuances of words.

We conclude the chapter by discussing the significance of word embeddings in NLP and tracing the evolution of embedding methods over the years, underscoring their transformative impact on the field.

2.1 Text Classification

Text classification is a fundamental task in NLP that involves assigning a text segment, denoted as x, to one or more predefined categories from a set $\mathcal{C}$. These text segments can vary in granularity, encompassing entire documents, paragraphs, sentences, or even individual words. Formally, the task can be defined as learning a function $f : \mathcal{X} \to \mathcal{C}$, where $\mathcal{X}$ represents the space of all possible text segments.

In binary classification, the set $\mathcal{C}$ consists of two labels, typically represented as $\{0, 1\}$. For

multi-class classification, $\mathcal{C}$ contains C distinct categories, and the task is to assign each text segment to exactly one category. Multi-label classification extends this by allowing a text segment to be associated with multiple categories simultaneously, thus $\mathcal{C}$ is a power set of possible categories.

The softmax function is commonly employed in multi-class classification tasks. It transforms the raw output scores of a model into a probability distribution over the C classes. This ensures that the probabilities sum to one, facilitating the assignment of the text segment to the class with the highest probability. In contrast, the sigmoid function is utilized in binary and multi-label classification tasks. It maps the model's output to a probability between 0 and 1 for each class independently. This allows for the independent prediction of each class, making it suitable for scenarios where multiple categories can be assigned to a single text segment. Multitask classification further extends these concepts by simultaneously learning multiple related tasks, each with its own set of categories. This approach leverages shared representations to improve performance across tasks.

A critical aspect of text classification is the conversion of text data into a numerical vector format, known as embeddings. Text data, being inherently discrete, requires this transformation to be processed by ML models. The simplest method, one-hot encoding, assigns a unique integer to each word in the vocabulary, resulting in a high-dimensional, sparse representation. However, this approach lacks semantic and contextual awareness.

Recent advancements in NLP have focused on developing embeddings that capture the semantic and syntactic nuances of words while being contextually aware of surrounding words. These embeddings, which we will explore in Section 2.6 and Chapter 4, have been instrumental in the success of modern NLP applications.

2.2 Sequential Text Classification

Sequential text classification extends text classification to tasks where each element in the input sequence, e.g., words in a sentence or paragraphs in a document, is assigned a label from a predefined set $\mathcal{C}$. Instead of classifying each text segment independently, sequential text classification takes into account the context provided by the entire sequence. This approach is crucial in sequence tagging tasks like Named Entity Recognition (NER) and Part-of-Speech (POS) tagging, where the meaning and function of each word depend heavily on its surrounding words.

Given a sequence of T text segments $\boldsymbol{x} = (x_1, x_2, \ldots, x_T)$, the goal is to predict a corresponding sequence of labels $\boldsymbol{y} = (y_1, y_2, \ldots, y_T)$, where each $y_t \in \mathcal{C}$.

To model the joint probability $\mathbb{P}(\boldsymbol{y} \mid \boldsymbol{x})$ of the label sequence given the input sequence, two main approaches are commonly used. The first is the conditional probability approach, which factorizes the joint probability into a product of conditional probabilities using the chain rule [41]:

$$\mathbb{P}(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{t=1}^{T} \mathbb{P}(y_t \mid \boldsymbol{x}, y_1, y_2, \ldots, y_{t-1}). \tag{2.1}$$

This approach allows each label y_t to depend on the entire input sequence $\boldsymbol{x}$ and all previous

labels, leveraging deep neural architectures like recurrent neural networks (RNNs) and transformers to capture complex long-range label dependencies. However, at inference time it can suffer from error propagation since ground truth labels are not available and each prediction depends on the correctness of the preceding ones.

The second approach is to use structured prediction models like conditional random fields (CRFs) [42], which model the joint probability without explicitly factoring over time steps. The CRF defines the probability as

$$\mathbb{P}(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \prod_{t=1}^{T} \psi(y_{t-1}, y_t, \boldsymbol{x}, t), \tag{2.2}$$

where $\psi(y_{t-1}, y_t, \boldsymbol{x}, t)$ is a potential function that captures the relationship between adjacent labels and the input sequence, and $Z(\boldsymbol{x})$ is a normalization constant ensuring the probabilities sum to one. The CRF approach jointly models the entire label sequence, which mitigates error propagation. However, it typically models only local dependencies between adjacent labels and can be computationally demanding for larger label sets since for each element in the sequence transitions between all possible label pairs have to be considered.

In Chapter 5, we apply the first approach by utilizing an RNN-based label decoding method enhanced with conditional label masking. We compare its performance to CRF decoding in the context of identifying and relating Key Performance Indicators (KPIs) within financial text documents.

2.3 Text Matching

Text matching is a crucial task in NLP that focuses on assessing the semantic similarity between two text segments. Unlike text classification, where a text segment x is assigned to a fixed set of categories $\mathcal{C}$, text matching aims to learn a similarity function capable of producing a similarity score for any pair of text segments. Formally, given two text segments $x, x' \in \mathcal{X}$, the objective is to learn a distance function $d : \mathcal{X} \times \mathcal{X} \to [0, K]$ that quantifies the semantic similarity between x and x' by mapping similar text pairs close to 0 and unrelated pairs to a large value K depending on the form of d.

A common approach to text matching involves using the same neural network model $f_\theta : \mathcal{X} \to \mathbb{R}^d$ with learnable parameters θ that encodes each text segment into a d-dimensional continuous vector representation [43, 44]. This siamese architecture results in comparable embeddings $\boldsymbol{h} = f_\theta(x)$ and $\boldsymbol{h}' = f_\theta(x')$ for the two text segments. The similarity between the text segments is then computed using the distance function $d(\boldsymbol{h}, \boldsymbol{h}')$ that measures the distance of the embeddings in the vector space. Common choices for distance functions are the euclidean distance $d_{\text{euclidean}} \in [0, \infty)$ and cosine distance, often defined as $d_{\text{cosine}} = 1 - \text{cosine similarity} \in [0, 2]$ [45].

To train such a system in a supervised way, a contrastive learning objective can be employed. The aim is to adjust the encoding network f_θ so that semantically similar text segments have low distance scores, while dissimilar segments have high scores. A general form of the contrastive

loss function for a single text pair can be defined as

$$\mathcal{L} = y \cdot L_S(d(\boldsymbol{h}, \boldsymbol{h}')) + (1 - y) \cdot L_D(d(\boldsymbol{h}, \boldsymbol{h}')), \tag{2.3}$$

where $y \in \{0, 1\}$ is the binary label that indicates whether x and x' are similar ($y = 1$) or dissimilar ($y = 0$), and L_S and L_D are the respective partial loss functions for a similar and dissimilar pair [46]. L_S and L_D are designed to penalize similar pairs that are far apart (high distance score) and dissimilar pairs that are close together (low distance score).

Exemplary choices for the partial loss functions as defined by [46] are

$$L_S = \frac{1}{2}\left(d(\boldsymbol{h}, \boldsymbol{h}')\right)^2, \quad L_D = \frac{1}{2}\left(\max\left(0, m - d(\boldsymbol{h}, \boldsymbol{h}')\right)\right)^2, \tag{2.4}$$

where $m > 0$ is a margin hyperparameter that defines the maximum distance for a dissimilar pair still contributing to the loss. The margin m ensures that the model focuses particularly on difficult-to-separate dissimilar pairs.

In Chapter 6, we apply contrastive learning to dynamically match semantically equivalent KPIs in financial reports. By learning models that capture the nuanced semantics of KPIs, we are able to perform consistency checks on their numerical values, ensuring they are equal and consistent throughout financial documents.

2.4 Topic Modeling

Topic modeling is an unsupervised learning technique employed to uncover latent topics within a collection of documents. Unlike text classification, which necessitates labeled data, topic modeling operates without such requirements, making it a versatile tool in NLP for applications such as document clustering, information retrieval, and recommendation systems.

One of the most prominent techniques in topic modeling is Latent Dirichlet Allocation (LDA) [47], a generative probabilistic model that posits each document as a mixture of topics, with each topic being a mixture of words.

Another widely used technique is Non-negative Matrix Factorization (NMF) [48], which factorizes the document-term matrix, containing the individual word frequencies per document, into two non-negative matrices, one representing the topics, while the other represents the topic-associated words.

Generally speaking, the aim of topic modeling is to extract meaningful topics from a corpus and assign these topics to documents based on the word distributions within them. Since the topics are not predefined, qualitative inspection and interpretation are often necessary to understand the extracted topics within the corpus.

In Chapter 3, we explore how topic modeling can be combined with word embedding learning to develop interpretable dense word vectors that capture not only the semantic and syntactic structure of words but can be qualitatively introspected since each embedding entry refers to an extracted corpus topic.

Table 2.1: Example of common Preprocessing methods to standardize text data and reduce the vocabulary size.

Text	`Fraunhofer's 2023 revenue was €~3 billion.`
Lowercasing	`fraunhofer's 2023 revenue was €~3 billion.`
Punctuation-Removal	`fraunhofers 2023 revenue was 3 billion`
Stopword-Removal	`fraunhofers 2023 revenue 3 billion`
Lemmatization	`fraunhofer 2023 revenue 3 billion`

2.5 Tokenization

Tokenization is a critical preprocessing step in the conversion of text data into a numerical representation suitable for input into ML models. This process involves segmenting text into smaller units, known as tokens, which can then be mapped to numerical indices. The most common level of granularity for tokenization is at the word level, where each word in the text is treated as a distinct token.

A notable challenge with word-level tokenization is the large size of the resulting vocabulary and the inevitable issue of unknown words that are absent from a pre-trained vocabulary (see Figure 2.1(c)). To reduce the vocabulary size, various preprocessing steps are employed, e.g., lowercasing, punctuation and stop word removal, and lemmatization or stemming, as illustrated in Table 2.1.

In addition to rare and unknown words, misspellings and numerical data present challenges for word-level tokenization. Given the infinite nature of numbers, it is impractical to maintain a vocabulary that contains all possible numbers without decomposing them into subword units.

Character-level tokenization offers an alternative approach, circumventing the problem of unknown words by utilizing a limited set of characters to construct all possible words. However, models based on character-level tokenization are computationally intensive and require substantially more data to learn the relationships between individual characters and subsequently words (see Figure 2.1(a)).

Subword tokenization strikes a balance between word and character-level tokenization by segmenting words into subword units based on their frequency in the corpus (see Figure 2.1(b)). Popular subword tokenization techniques include Byte Pair Encoding (BPE) [49] and WordPiece [50] tokenization, which are employed in modern language models like BERT [34] and GPT [17]. For a comprehensive analysis and quantitative evaluation of various subword tokenization methods across multilingual corpora, a diverse range of downstream tasks, and different vocabulary sizes, we refer to [51].

2.6 Evolution of Word Embeddings

Building upon the different tokenization methods, this section explores the early progression of word embeddings, which are numerical representations that encapsulate the semantic and syntactic properties of words or documents. We divide this section into two parts:

First, we examine count-based embeddings that rely on word frequency and importance

Input: `Fraunhofer's 2023 revenue was €~3 billion.`

Tokens	`F r a u n h o f e r ' s _ 2 0 2 3 _ r e v e n u e _ w a s _ € ~ 3 _ b i l l i o n .`
Indices	`70 114 97 117 110 104 111 102 101 114 39 115 95 50 48 50 51 95 114 101 118 101 110 117 101 95 119 97 115 95 8364 126 51 95 98 105 108 108 105 111 110 46`

(a) **Character Tokenization.** Each index in the vocabulary represents the unicode integer of the respective character. "_" highlights the whitespace character for better readability.

Tokens	`Fr ##au ##nh ##of ##er ' s 202 ##3 revenue was € ~ 3 billion .`
Indices	`13359 3984 15624 10008 1200 112 188 17881 1495 7143 1108 836 199 124 3775 119`

(b) **Subword Tokenization.** Rare words and numbers not present in the vocabulary are split into subwords to avoid the OOV issue. The "`##`" prefix indicates the continuation of a subword.

Tokens	`[UNK] ' s [UNK] revenue was € ~ 3 billion .`
Indices	`0 39 115 0 21452 4148 8364 126 51 1319 46`

(c) **Word Tokenization.** The sentence is split on whitespace and punctuation marks and subsequently mapped to a large vocabulary. Words or numbers not present in the vocabulary are replaced with the special "`[UNK]`" (unknown) token.

Figure 2.1: Comparison of (a) character-, (b) subword-, and (c) word-level tokenization. Subword tokenization trades off between character- and word-level tokenization by splitting rare words and numbers into subword units to avoid the out-of-vocabulary (OOV) issue while maintaining a relatively short sequence length and a reasonable vocabulary size.

without the need for complex ML models. In the second part, we introduce semantic embeddings, which are generated through ML techniques like Word2Vec [52] and GloVe [53], capturing deeper semantic relationships but lacking interpretability.

In Chapter 3 we address this shortcoming and present a novel matrix factorization approach based on the DEDICOM model [54] that yields semantic but interpretable word embeddings while simultaneously producing distinct topics for a given corpus.

Table 2.2 provides a high-level comparison of these embedding techniques, highlighting their differences, weaknesses, and strengths.

Since the current state-of-the-art, dynamically generated contextual embeddings, is tightly linked to language modeling [55] and deep neural architectures like RNNs and Transformers [19], we cover it in full detail in Chapter 4.

Table 2.2: Comparison of Word Embedding Techniques, categorized by type, vector characteristics, and semantic capabilities.

Method	Type	Vector Type	Interpretable	Semantic
Bag of Words	Word Frequency	Sparse	✓	✗
TF-IDF	Word Importance	Sparse	✓	✗
Word2Vec	Learned	Dense	✗	✓
GloVe	Learned	Dense	✗	✓
DEDICOM	Learned	Dense	✓	✓

2.6.1 Count Statistics-Based Embeddings

Count statistics-based embeddings disregard word order and context, focusing solely on the frequency of words within a corpus. These methods are computationally efficient, straightforward to compute, and highly interpretable but fail to capture the semantic relationships between words.

Bag of Words

The Bag of Words (BoW) model is a fundamental technique for text representation that has been employed for several decades [56]. It is one of the simplest methods for numerically encoding text. In this model, each document d is represented as a vector $\boldsymbol{v}_d \in \mathbb{R}^V$, where V denotes the size of the vocabulary. The i-th element of $\boldsymbol{v}_d$ corresponds to the frequency of the i-th word in the document. Individually, each word is represented as a one-hot encoded vector, with dimensionality equal to the size of the vocabulary. Formally, the BoW representation is defined as $\boldsymbol{v}_d[i] = \text{count}(w_i, d)$, where $\text{count}(w_i, d)$ indicates the number of times word w_i appears in document d. This approach results in a sparse representation that does not capture the context or semantic relationships between words.

To illustrate, consider a simple corpus consisting of two documents

Document 1: "The sith Darth Vader is feared across the galaxy."
Document 2: "The jedi Luke Skywalker is loved across the galaxy."

with the joint vocabulary (after lowercasing and punctuation-removal preprocessing)

{the, sith, darth, vader, is, feared, across, galaxy, jedi, luke, skywalker, loved}.

The BoW vectors, $\boldsymbol{v}_i^{\text{BoW}} \in \mathbb{R}^{12}$, for these documents are constructed based on their respective term occurrences:

$$\boldsymbol{v}_1^{\text{BoW}} = \begin{bmatrix} 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix},$$
$$\boldsymbol{v}_2^{\text{BoW}} = \begin{bmatrix} 2 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

It is evident that common words like “the” and “is” have high occurrences, while more contextually significant words such as “sith” and “jedi” appear less frequently. This underscores a limitation of the BoW model: its inability to capture the relative importance of words within a document in the context of the entire corpus.

TF-IDF

The Term Frequency-Inverse Document Frequency (TF-IDF) model [57] enhances the BoW approach by considering the significance of words in a document relative to the entire corpus. It normalizes the BoW count statistics by the inverse document frequency (IDF) of words, thereby assigning higher weights to words that are frequent in a document but rare across the corpus. This adjustment makes TF-IDF more informative, particularly for distinguishing common words, like “and”, “is” and “the”, from those that are more contextually significant, typically nouns or descriptive verbs and adjectives. The TF-IDF value for a word w_i in document d is calculated as

$$\text{TF-IDF}(w_i, d) = \text{TF}(w_i, d) \cdot \text{IDF}(w_i), \text{ with} \tag{2.5}$$

$$\text{TF}(w_i, d) = \frac{\text{count}(w_i, d)}{\sum_{w \in d} \text{count}(w, d)}, \text{ and} \tag{2.6}$$

$$\text{IDF}(w_i) = \log\left(N/(1 + \text{DF}(w_i))\right), \tag{2.7}$$

where $\text{TF}(w_i, d)$ represents the term frequency normalized by the total number of words in the document, and $\text{IDF}(w_i)$ is the inverse document frequency, with N being the total number of documents and $\text{DF}(w_i)$ the number of documents containing w_i.

Applying TF-IDF to the above toy corpus of two documents, the resulting vectors are:

$$\boldsymbol{v}_1^{\text{TF-IDF}} = \begin{bmatrix} 0.13 & 0.11 & 0.11 & 0.11 & 0.07 & 0.11 & 0.07 & 0.07 & 0.00 & 0.00 & 0.00 & 0.00 \end{bmatrix},$$

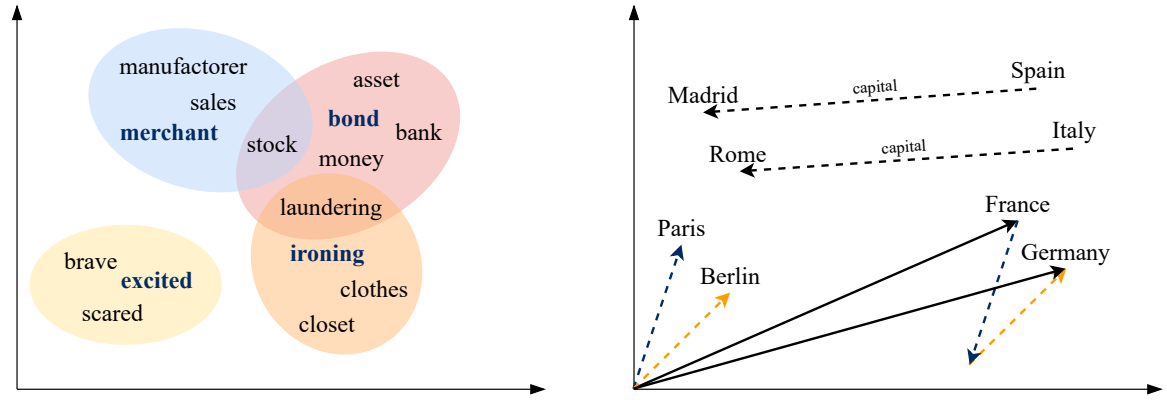
$$\boldsymbol{v}_2^{\text{TF-IDF}} = \begin{bmatrix} 0.13 & 0.00 & 0.00 & 0.00 & 0.07 & 0.00 & 0.07 & 0.07 & 0.11 & 0.11 & 0.11 & 0.11 \end{bmatrix}.$$

In contrast to BoW, TF-IDF scores for common words like “the” and “is” are relatively lower, reflecting their omnipresence across the corpus. Conversely, the scores for informative words like “sith” and “jedi” are relatively higher, indicating their importance within the respective documents.

Despite its simplicity and lack of semantics, TF-IDF remains relevant, particularly in fast hybrid search systems that combine traditional search algorithms like BM25 [58] with modern semantic search methods.

2.6.2 Semantic Embeddings

“You shall know a word by the company it keeps!” [60, p. 11]. This famous quote by John Rupert Firth encapsulates the foundational idea behind semantic embeddings. Unlike traditional methods such as BoW and TF-IDF, which rely on word occurrence counts and fail to capture semantic relationships, semantic embeddings offer a more sophisticated representation. They



(a) (b)

Figure 2.2: (a) An illustrative example showing semantically similar words clustered together in vector space. Homonyms like "stock" and "laundering" appear at the intersections of different clusters. (b) Demonstration of the approximate algebraic properties of semantic word embeddings: the vector for "Germany" can be approximately derived from the equation "France - Paris + Berlin". Similarly, adding the vectors for "Spain" and "Capital" yields a vector very similar to "Madrid". Note for real embeddings, the algebraic operations are not exact but approximate [59].

embed words in a latent vector space that aims to encode both semantic and syntactic language properties.

This section details the two pioneering approaches for learning such embeddings: Word2Vec [52] and GloVe [53]. These methods are trained to position semantically similar words close to each other in vector space, which enables subsequent ML models to better distinguish between different samples in downstream classification tasks.

Figure 2.2 illustrates the algebraic vector space properties of semantic embeddings. In part (a), semantically similar words are clustered together in vector space. The blue boldface words in each cluster exemplify how a semantically structured vector space can be utilized for text classification, as similar words occupy similar regions.

Part (b) of Figure 2.2 demonstrates the algebraic properties of semantic word embeddings [59]. For example, the vector for "Germany" can be approximately derived from the equation "France - Paris + Berlin". Similarly, adding the vectors for "Spain" and "Capital" yields a vector very similar to the word "Madrid". These approximate algebraic operations are possible due to the encoded relationships within the vector space.

The training algorithms of semantic word embeddings focus on neighboring words, often resulting in words with similar POS tags being located near each other. Interestingly, this proximity also extends to synonyms and antonyms within the vector space (see "brave" and "scared" in Figure 2.2(a)).

Despite their advantages, these methods have limitations compared to count-based embeddings. The resulting dense vectors are not easily interpretable, as each index does not correspond to a specific word or semantic concept. To address this limitation, Chapter 3

introduces DEDICOM, a semantic word embedding method that integrates word embedding learning with topic modeling to produce interpretable dense word embeddings.

Word2Vec

Word2Vec [61], introduced in 2013, represents a novel approach to learn dense word embeddings, capturing semantic relationships between words. Unlike frequency-based methods such as TF-IDF, which produce sparse vectors based on word frequency, Word2Vec generates dense vectors by learning from a large corpus of text data using neural networks. The authors of this method proposed two primary architectures for learning word embeddings: Continuous Bag of Words (CBOW) and Skip-Gram.

The CBOW model predicts a target word w_t based on its surrounding context, defined as a window of words. For instance, in the sentence "The Jedi master trains the young Padawan", with a window size of 2, the context words for "master" would be ["The", "Jedi", "trains", "the"]. In this model, the input consists of the context words, each represented as a one-hot encoded vector. These vectors are projected into a shared hidden layer, typically by mean or max pooling the embedding vectors corresponding to the context words. The output layer, a softmax function, predicts the target word by computing probabilities based on the dot product between the embeddings and the softmax weights.

Conversely, the Skip-Gram model uses the target word to predict its context words. Given the same example sentence and the target word "master", the Skip-Gram model aims to predict the context words ["The", "Jedi", "trains", "the"]. Here, the input is only the target word. The output layer consists of multiple softmax units, each predicting a word in the context, and the model outputs a probability distribution for each context position. The objective is to adjust the embeddings to maximize the probability of the correct context words given the target word.

Formally, the loss functions for the CBOW and Skip-Gram models are defined as

$$\mathcal{L}_{\text{CBOW}} = -\frac{1}{T}\sum_{t=1}^{T}\log \mathbb{P}(w_t \mid w_{t-c}, \ldots, w_{t+c}), \tag{2.8}$$

$$\mathcal{L}_{\text{Skip-Gram}} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log \mathbb{P}(w_{t+j} \mid w_t), \tag{2.9}$$

where T is the total number of words in the corpus and c is the context window size.

Word2Vec embeddings are static, which means they do not change at inference time based on context. The choice between CBOW and Skip-Gram depends on the task requirements. CBOW is generally faster and provides slightly better representations for syntactic tasks, while Skip-Gram, though slower, excels in capturing semantic relationships [61].

Overall, Word2Vec's ability to capture semantic relationships through dense word embeddings represented a major milestone in NLP and enabled tasks such as analogy reasoning and semantic similarity measurement for the first time.

GloVe

GloVe (Global Vectors for Word Representation) [53] is an alternative approach to learning dense word embeddings that encode semantics and syntactical structure. Unlike word2vec

which leverages a neural network architecture, GloVe effectively captures linear substructures of the word vector space by employing matrix factorization techniques on the word co-occurrence matrix of a large document corpus. The primary objective of GloVe is to leverage the statistical information of word occurrences in a corpus to produce embeddings where semantically similar words have similar representations.

GloVe constructs a word co-occurrence matrix $\boldsymbol{X} \in \mathbb{R}^{n \times n}$, where each element x_{ij} denotes the frequency with which word i occurs in the context of word j and n is the vocabulary size of the entire corpus. This is similar to the count-based approach of TF-IDF, but GloVe uses a sliding context window to encode relationships between words into these statistics. Instead of modeling the co-occurrence probabilities $\mathbb{P}_{ij} = \mathbb{P}(j \mid i) = x_{ij} / \sum_{j=1}^{n} x_{ij}$ directly, GloVe focuses on the ratios of co-occurrence probabilities, which capture the relative relationships between word pairs in relation to other words. Specifically, for three words i, j, and k, the ratio of probabilities $\mathbb{P}_{ik}/\mathbb{P}_{jk}$ reflects how much more (> 1) or less (< 1) associated word k is with word i than with word j. To represent this in vector space, GloVe encodes these ratios using the differences between word vectors

$$\exp\left((\boldsymbol{w}_i - \boldsymbol{w}_j)^\top \tilde{\boldsymbol{w}}_k\right) = \frac{\mathbb{P}_{ik}}{\mathbb{P}_{jk}} = \frac{\exp\left(\boldsymbol{w}_i^\top \tilde{\boldsymbol{w}}_k\right)}{\exp\left(\boldsymbol{w}_j^\top \tilde{\boldsymbol{w}}_k\right)}, \tag{2.10}$$

where $\boldsymbol{w}_i, \boldsymbol{w}_j \in \mathbb{R}^d$ are d-dimensional word vectors and $\tilde{\boldsymbol{w}}_k \in \mathbb{R}^d$ are context word vectors. By leveraging properties of logarithms and accounting for model symmetry, this relationship can be simplified to

$$\boldsymbol{w}_i^\top \tilde{\boldsymbol{w}}_k = \log \mathbb{P}_{ik} = \log x_{ik} - b_i - \tilde{b}_k, \tag{2.11}$$

where b_i and $\tilde{b}_k$ are bias terms that absorb the logarithmic count of how often any word occurs in the context of word i ($\log \sum_{k=1}^{n} x_{ik}$). The derived model seeks to learn these vectors and biases such that their interactions approximate the logarithm of the co-occurrence counts. The resulting loss function is defined as

$$\mathcal{L}_{\text{GloVe}} = \sum_{i,j=1}^{n} f(x_{ij}) \left(\boldsymbol{w}_i^\top \tilde{\boldsymbol{w}}_j + b_i + \tilde{b}_j - \log x_{ij}\right)^2, \tag{2.12}$$

where $f(x_{ij})$ is a weighting function that reduces the impact of very frequent co-occurrences. For a more detailed derivation of the GloVe model, we refer to the original paper [53].

Both GloVe and Word2Vec yield semantic word embeddings, but their approaches to capturing word relationships differ. As described in Section 2.6.2, Word2Vec utilizes local context information by training a neural network to predict adjacent words in a sentence. In contrast, GloVe constructs an explicit word co-occurrence matrix using global statistical information from the corpus and then applies matrix factorization.

Lack of interpretability is a major problem with both GloVe and Word2Vec. These methods often produce high-dimensional word vectors, typically with dimensions $d \geq 100$. The individual entries of these vectors do not directly correspond to specific semantic meanings or concepts. For instance, when a model that uses these embeddings is applied to a downstream task such as sentiment analysis, it becomes harder to understand the model's decision-making process

and to identify potential errors without interpretable dimensions in the embedding space. In the subsequent chapter, we address this issue by presenting a novel approach for generating interpretable word embeddings. Our method, row-stochastic DEDICOM, involves factorizing a logarithmic and normalized word co-occurrence matrix, allowing us to interpret the resulting word embedding dimensions as distinct topics present in the text corpus.

CHAPTER 3

DEDICOM for Interpretable Word Embeddings and Topic Modeling

Building upon the foundational concepts of word embeddings discussed in Chapter 2, we now turn our attention to a new matrix factorization-based method that enhances the interpretability of word embeddings while simultaneously facilitating topic modeling. As established, word embeddings are crucial for transforming text into numerical vectors that ML models can process. Traditional methods, such as GloVe, utilize matrix factorization to derive these embeddings from word co-occurrence matrices. However, these embeddings often lack interpretability, as the dimensions of the resulting vectors do not correspond to any specific semantic concepts.

In this chapter, we introduce a novel method that addresses this limitation by employing the DEDICOM (DEcomposition into DIrectional COMponents) [54] algorithm, a matrix factorization technique that allows for the creation of interpretable word embeddings. This approach not only retains the semantic richness of traditional embeddings but also provides a clear interpretative framework by associating each dimension of the embedding with distinct topics derived from the corpus.

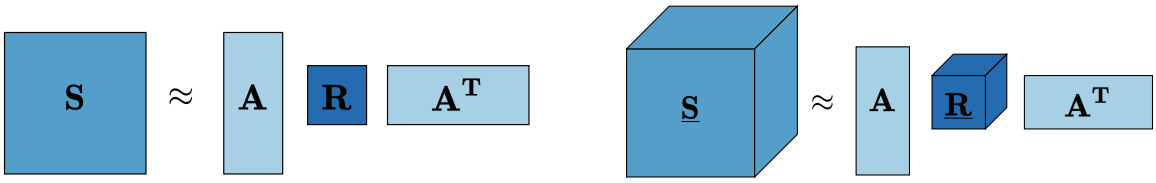
The DEDICOM model factorizes a square matrix $\boldsymbol{S} \in \mathbb{R}^{n \times n}$ into three matrices, expressed as

$$\boldsymbol{S} \approx \boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^\top, \quad \boldsymbol{A} \in \mathbb{R}^{n \times k}, \quad \boldsymbol{R} \in \mathbb{R}^{k \times k}, \tag{3.1}$$

where $\boldsymbol{A}$ is the loading matrix and $\boldsymbol{R}$ is the affinity matrix. By applying this factorization to a pointwise mutual information matrix of word co-occurrence statistics, we derive a matrix $\boldsymbol{A}$ that is row-stochastic, ensuring that each word embedding represents a probability distribution over k topics. This constraint allows us to interpret each word embedding as a mixture of topics, with the significance of each topic indicated by the magnitude of the corresponding entry in the word vector.

The affinity matrix $\boldsymbol{R}$ further enriches this model by capturing the relationships between topics. A high value in r_{bc} suggests a strong co-occurrence likelihood between words associated with topics b and c, thereby revealing semantic relationships between these topics. This interpretative capability is extended through a tensor factorization approach, which accommodates multiple text sources, enabling the analysis of topic relationships across different corpora.

Our research, as detailed in [36] and [37], demonstrates the efficacy of this approach in generating interpretable word embeddings across various text corpora. The DEDICOM model



(a) (b)

Figure 3.1: (a) The DEDICOM algorithm factorizes a square matrix $\boldsymbol{S} \in \mathbb{R}^{n \times n}$ into a loading matrix $\boldsymbol{A} \in \mathbb{R}^{n \times k}$ and an affinity matrix $\boldsymbol{R} \in \mathbb{R}^{k \times k}$. (b) The tensor DEDICOM algorithm factorizes a three-dimensional tensor $\underline{\boldsymbol{S}} \in \mathbb{R}^{t \times n \times n}$ into a loading matrix $\boldsymbol{A} \in \mathbb{R}^{n \times k}$ and a three-dimensional affinity tensor $\underline{\boldsymbol{R}} \in \mathbb{R}^{t \times k \times k}$.

not only highlights logical relationships between topics but also adapts to the dynamic focus of topics, as evidenced in longitudinal analyses of news articles over time.

This chapter is based on the following publications:

- L. Hillebrand, D. Biesner, C. Bauckhage, and R. Sifa, "Interpretable Topic Extraction and Word Embedding Learning Using Row-Stochastic DEDICOM," *Proc. CD-MAKE*, 2020, DOI: 10.1007/978-3-030-57321-8_22 [36],
- L. Hillebrand, D. Biesner, C. Bauckhage, and R. Sifa, *Interpretable Topic Extraction and Word Embedding Learning Using Non-Negative Tensor DEDICOM*, Machine Learning and Knowledge Extraction (2021), DOI: 10.3390/make3010007 [37].

As shared first authors, Lars Hillebrand and David Biesner contributed equally to the research, planning, and conceptualization of the project. Together, they jointly derived the update rules for the matrix formulation of the DEDICOM algorithm. Lars Hillebrand was primarily responsible for implementing the experimental framework, which encompassed data preparation, training with techniques designed to enforce row-stochasticity and matrix scaling for faster convergence, and evaluation. While the experiments and the writing of the papers, including the creation of tables and figures, were collaborative efforts, Lars Hillebrand took the lead in the experiments section, detailing the data processing, training procedures, and findings.

3.1 Introduction

Matrix factorization methods have always been a staple in many NLP tasks. Factorizing a matrix of word co-occurrences can create both low-dimensional representations of the vocabulary, so-called word embeddings [53, 62], that carry semantic and topical meaning within them, as well as representations of meaning that go beyond single words to latent topics.

DEDICOM is a matrix factorization technique that factorizes a square, possibly asymmetric, matrix of relationships between items into a loading matrix of low-dimensional representations of each item and an affinity matrix describing the relationships between the dimensions of the latent representation (see Figure 3.1 for an illustration).

We introduce a modified row-stochastic variation of DEDICOM, which allows for interpretable loading vectors, and apply it to different matrices of word co-occurrence statistics created from

Wikipedia-based semi-artificial text documents. Our algorithm produces low-dimensional word embeddings, where one can interpret each latent factor as a topic that clusters words into meaningful categories. Hence, we show that row-stochastic DEDICOM successfully combines the task of learning interpretable word embeddings and extracting representative topics.

We further derive a similar model for factorization of three-dimensional data tensors, which represent word co-occurrence statistics for text corpora with an intrinsic structure that allows for some separation of the corpus into subsets (e.g., a news corpus structured by time).

An interesting aspect of this type of factorization is the interpretability of the affinity matrix. An entry in the matrix directly describes the relationship between the topics of the respective row and column, and one can therefore use this tool to extract topics that a certain text corpus deals with and analyze how these topics are connected in the given text.

In this chapter, we first describe the aforementioned DEDICOM algorithm and provide details on the modified row-stochasticity constraint and on optimization. We further expand our model to factorize three-dimensional tensors and introduce a multiplicative update rule that facilitates the training procedure. We then present results of various experiments on both semi-artificial text documents (combinations of Wikipedia articles) and real text documents (movie reviews and news articles) that show how our approach is able to capture hidden latent topics within text corpora, cluster words in a meaningful way, and find relationships between these topics within the documents.

3.2 Related Work

Matrix factorization describes the task of compressing the most relevant information from a high-dimensional input matrix into multiple low-dimensional factor matrices, with either approximate or exact input reconstruction (see, for example, [63] for a theoretical overview of common methods and their applications). In this work, we consider the DEDICOM algorithm, which has a long history of providing an interpretable matrix or tensor factorization, mostly for rather low-dimensional tasks.

First described in [54], it since has been applied to analysis of social networks [64], email correspondence [65] and video game player behavior [66, 67]. DEDICOM has also been successfully employed in NLP tasks, such as part-of-speech tagging [68]. However, to the best of our knowledge, we provide the first implementation of DEDICOM for simultaneous word embedding learning and topic modeling.

Many works deal with the task of putting constraints on the factor matrices of the DEDICOM algorithm. In [65, 66], the authors constrain the affinity matrix $\boldsymbol{R}$ to be non-negative, which aids interpretability and improves convergence behavior if the matrix to be factorized is non-negative. However, their approach relies on the Kronecker product between matrices in the update step, solving a linear system of $n^2 \times k^2$, where n denotes the number of items in the input matrix and k is the number of latent factors. These dimensions make the application on text data, where n describes the number of words in the vocabulary, a computationally futile task. Constraints on the loading matrix, $\boldsymbol{A}$, include non-negativity as well (see [65]) or column-orthogonality as in [66].

In contrast, we propose a new modified row-stochasticity constraint on $\boldsymbol{A}$, which is tailored to generate interpretable word embeddings that carry semantic meaning and represent a probability

distribution over latent topics.

The DEDICOM algorithm has previously been applied to tensor data as well, for example, in [69], in which the authors apply the algorithm on general multi-relational data by computing an exact solution for the affinity matrix. Both [64] and [65] explore a slight variation of our tensor DEDICOM approach to analyze relations in email data and [70] apply a similar model on non-square input tensors.

Previous matrix factorization-based methods in the NLP context mostly dealt with either word embedding learning or topic modeling, but not with both tasks combined.

For word embeddings, the GloVe [53] model factorizes an adjusted co-occurrence matrix into two matrices of the same dimension. The work is based on a large text corpus with a vocabulary of $n \approx 400,000$ and produces word embeddings of dimension $k = 300$. To maximize performance on the word analogy task, the authors adjusted the co-occurrence matrix to the logarithmic co-occurrence matrix and added bias terms to the optimization objective.

A model conceived around the same time, word2vec [52], calculates word embeddings not from a co-occurrence matrix but directly from the text corpus using the skip-gram or continuous-bag-of-words approach. More recent work [62] has shown that this construction is equivalent to matrix factorization on the pointwise mutual information (PMI) matrix of the text corpus, which makes it very similar to the glove model described above.

Both models achieve impressive results on word embedding-related tasks like word analogy. However, the large dimensionality of the word embeddings makes interpreting the latent factors of the embeddings impossible.

On the topic modeling side, matrix factorization methods are routinely applied as well. Popular algorithms like Non-negative Matrix Factorization (NMF) [48], Singular Value Decomposition (SVD) [71, 72] and Principal Component Analysis (PCA) [73] compete against the probabilistic Latent Dirichlet Allocation (LDA) [47] to cluster the vocabulary of a word co-occurrence or document-term matrix into latent topics.[1] Yet, we empirically show that the implicitly learned word embeddings of these methods lack semantic meaning in terms of the cosine similarity measure.

We benchmark our approach qualitatively against these methods in Section 3.4.3 and the appendix.

3.3 Constrained DEDICOM Models

In this section, we provide a detailed theoretical view of different constrained DEDICOM algorithms utilized for factorizing word co-occurrence-based positive pointwise mutual information matrices and tensors.

We first consider the case of a two-dimensional input matrix $\boldsymbol{S}$ (see Figure 3.1(a)) in Section 3.3.1. We then present an extension of the algorithm for three-dimension input tensors $\underline{\boldsymbol{S}}$ (see Figure 3.1(b)) in Section 3.3.2. Finally, we derive a multiplicative update rule for non-negative tensor DEDICOM.

[1] More recent expansions of these methods can be found in [74, 75].

3.3.1 The row-stochastic DEDICOM Model for matrices

For a given language corpus consisting of n unique words $X = x_1, \ldots, x_n$ we calculate a co-occurrence matrix $\boldsymbol{X} \in \mathbb{R}^{n \times n}$ by iterating over the corpus on a word token level with a sliding context window of specified size. Then

$$x_{ij} = \#\text{word } i \text{ appears in context of word } j. \tag{3.2}$$

Note that the word context window can be applied symmetrically or asymmetrically around each word. We choose a symmetric context window, which implies a symmetric co-occurrence matrix, $x_{ij} = x_{ji}$.

We then transform the co-occurrence matrix into the pointwise mutual information matrix (PMI), which normalizes the counts to extract meaningful co-occurrences from the matrix. Co-occurrences of words that occur regularly in the corpus are decreased since their appearance together might be nothing more than a statistical phenomenon, the co-occurrence of words that appear less often in the corpus gives us meaningful information about the relations between words and topics. We define the PMI matrix as

$$\text{pmi}_{ij} \coloneqq \log x_{ij} + \log N - \log N_i - \log N_j \tag{3.3}$$

where $N \coloneqq \sum_{ij=1}^{n} x_{ij}$ is the sum of all co-occurrence counts of $\boldsymbol{X}$, $N_i \coloneqq \sum_{j=1}^{n} x_{ij}$ the row sum and $N_j \coloneqq \sum_{i=1}^{n} x_{ij}$ the column sum.

Since the co-occurrence matrix $\boldsymbol{X}$ is symmetrical, the transformed PMI matrix is symmetrical as well. Nevertheless, DEDICOM is able to factorize both symmetrical and non-symmetrical matrices. We expand details on symmetrical and non-symmetrical relationships in Section 3.3.4.

Additionally, we want all entries of the matrix to be non-negative, our final matrix to be factorized is therefore the positive PMI (PPMI)

$$s_{ij} = \text{ppmi}_{ij} = \max\{0, \text{pmi}_{ij}\}. \tag{3.4}$$

Our aim is to decompose this matrix using row-stochastic DEDICOM as

$$\boldsymbol{S} \approx \boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^\top, \quad \text{with} \quad s_{ij} \approx \sum_{b=1}^{k}\sum_{c=1}^{k} a_{ib} r_{bc} a_{jc}, \tag{3.5}$$

where $\boldsymbol{A} \in \mathbb{R}^{n \times k}$, $\boldsymbol{R} \in \mathbb{R}^{k \times k}$, $\boldsymbol{A}^\top$ denotes the transpose of $\boldsymbol{A}$ and $k \ll n$. Literature often refers to $\boldsymbol{A}$ as the loading matrix and $\boldsymbol{R}$ as the affinity matrix. $\boldsymbol{A}$ gives us for each word i in the vocabulary a vector of size k, the number of latent topics we wish to extract. The square matrix $\boldsymbol{R}$ then provides the possibility for interpretation of the relationships between these topics.

Empirical evidence has shown that the algorithm tends to favor columns unevenly, such that a single column receives a lot more weight in its entries than the other columns. We try to balance this behavior by applying a column-wise z-normalization on $\boldsymbol{A}$, such that all columns have zero mean and unit variance.

To aid interpretability, we wish each word embedding to be a distribution over all latent topics, i.e., entry a_{ib} in the word-embedding matrix provides information on how much topic b

describes word i.

To implement these constraints, we therefore apply a row-wise softmax operation over the column-wise z-normalized $\boldsymbol{A}$ matrix by defining $\boldsymbol{A'} \in \mathbb{R}^{n \times k}$ as

$$\begin{aligned} a'_{ib} &\coloneqq \frac{\exp(\bar{a}_{ib})}{\sum_{b'=1}^{k} \exp(\bar{a}_{ib'})}, \quad \bar{a}_{ib} \coloneqq \frac{a_{ib} - \mu_b}{\sigma_b}, \\ \mu_b &\coloneqq \frac{1}{n} \sum_{i=1}^{n} a_{ib}, \quad \sigma_b \coloneqq \sqrt{\frac{1}{n} \sum_{i=1}^{n} (a_{ib} - \mu_b)^2} \end{aligned} \tag{3.6}$$

and optimizing $\boldsymbol{A}$ for the objective

$$\boldsymbol{S} \approx \boldsymbol{A'} \boldsymbol{R} (\boldsymbol{A'})^\top. \tag{3.7}$$

Note that after applying the row-wise softmax operation, all entries of $\boldsymbol{A'}$ are non-negative.

To judge the quality of the approximation (3.7) we apply the Frobenius norm, which measures the difference between $\boldsymbol{S}$ and $\boldsymbol{A'} \boldsymbol{R} \left(\boldsymbol{A'}\right)^\top$. The final loss function we optimize our model for is therefore given by

$$\mathcal{L}(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R}) = \left\| \boldsymbol{S} - \boldsymbol{A'} \boldsymbol{R} (\boldsymbol{A'})^\top \right\|_F^2 \tag{3.8}$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} \left(s_{ij} - \left(\boldsymbol{A'} \boldsymbol{R} (\boldsymbol{A'})^\top \right)_{ij} \right)^2 \tag{3.9}$$

with

$$\left(\boldsymbol{A'} \boldsymbol{R} (\boldsymbol{A'})^\top \right)_{ij} = \sum_{b=1}^{k} \sum_{c=1}^{k} a'_{ib} r_{bc} a'_{jc} \tag{3.10}$$

and $\boldsymbol{A'}$ defined in (3.6).

To optimize the loss function we train both matrices using alternating gradient descent similar to [66]. Within each optimization step, we apply

$$\boldsymbol{A} \leftarrow\!\shortmid \boldsymbol{A} - f_\theta(\nabla_{\boldsymbol{A}}, \eta^{\boldsymbol{A}}), \quad \text{where} \quad \nabla_{\boldsymbol{A}} = \frac{\partial \mathcal{L}(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})}{\partial \boldsymbol{A}} \tag{3.11}$$

$$\boldsymbol{R} \leftarrow\!\shortmid \boldsymbol{R} - f_\theta(\nabla_{\boldsymbol{R}}, \eta^{\boldsymbol{R}}), \quad \text{where} \quad \nabla_{\boldsymbol{R}} = \frac{\partial \mathcal{L}(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})}{\partial \boldsymbol{R}} \tag{3.12}$$

with $\eta^{\boldsymbol{A}}, \eta^{\boldsymbol{R}} > 0$ being individual learning rates for both matrices and $f_\theta(\cdot)$ representing an arbitrary gradient-based update rule with additional hyperparameters θ. For our experiments, we employ automatic differentiation methods. For details on the implementation of the algorithm above, we refer to Section 3.4.2.

3.3.2 The constrained DEDICOM model for tensors

In this section, we extend the model described above to three-dimensional tensors as input data. As above, the input describes the co-occurrences of vocabulary items in a text corpus. However,

we consider additionally structured text: Instead of one matrix describing the entire corpus, we unite multiple $n \times n$ matrices of co-occurrences into one tensor $\underline{\boldsymbol{S}} \in \mathbb{R}^{t \times n \times n}$. Each of the t slices then consists of an adjusted PPMI matrix for a subset of the text corpus. This structure could originate for instance from different data (e.g., different Wikipedia articles), different topical subsets of the data source (e.g., reviews for different articles) or describe time-slices (e.g., news articles for certain time periods).

To construct the PPMI tensor we again take a vocabulary $X = x_1, \ldots, x_n$ over the entire corpus. For each subset l, we then calculate a co-occurrence matrix $\boldsymbol{X}_l \in \mathbb{R}^{n \times n}$ as described above. Stacking these matrices yields the co-occurrence tensor $\underline{\boldsymbol{X}} \in \mathbb{R}^{t \times n \times n}$.

When transforming slice $\boldsymbol{X}_l$ into a PMI matrix we want to apply information from the entire corpus. We therefore do not only calculate the column, the row, and the total sums on the corresponding subset but on the entire text corpus. Therefore,

$$\mathrm{pmi}_{lij} \coloneqq \log x_{lij} + \log N - \log N_i - \log N_j, \tag{3.13}$$

where $N \coloneqq \sum_{l=1}^{k} \sum_{ij=1}^{n} x_{lij}$ is the sum of all co-occurrence counts of $\underline{\boldsymbol{X}}$, $N_i \coloneqq \sum_{l=1}^{k} \sum_{j=1}^{n} x_{lij}$ the row sum and $N_j \coloneqq \sum_{l=1}^{k} \sum_{i=1}^{n} x_{lij}$ the column sum.

Finally, we define the positive pointwise mutual information tensor as

$$s_{lij} = \mathrm{ppmi}_{lij} = \max\{0, \mathrm{pmi}_{lij}\}. \tag{3.14}$$

We decompose this input tensor into a matrix $\boldsymbol{A} \in \mathbb{R}^{n \times k}$ and a tensor $\underline{\boldsymbol{R}} \in \mathbb{R}^{t \times k \times k}$, such that

$$\underline{\boldsymbol{S}} \approx \boldsymbol{A}\underline{\boldsymbol{R}}\boldsymbol{A}^\top \tag{3.15}$$

where we multiply each slice of $\underline{\boldsymbol{R}}$ with $\boldsymbol{A}$ and $\boldsymbol{A}^\top$ to reconstruct the corresponding slice of $\underline{\boldsymbol{S}}$:

$$s_{lij} \approx \sum_{b=1}^{k} \sum_{c=1}^{k} a_{ib} r_{lbc} a_{jc}. \tag{3.16}$$

We keep our naming convention for $\boldsymbol{A}$ as the loading matrix and $\underline{\boldsymbol{R}}$ as the affinity tensor, since again $\boldsymbol{A}$ gives us for each word i in the vocabulary a vector of size k and for each slice l, the square matrix $\boldsymbol{R}_l \coloneqq (r_{lij})_{i,j=1}^{k}$ provides information on the relationships between the topics in the l-th input slice.

Analogous to (3.8) we construct a loss function

$$\mathcal{L}(\underline{\boldsymbol{S}}, \boldsymbol{A}, \underline{\boldsymbol{R}}) = \left\| \underline{\boldsymbol{S}} - \boldsymbol{A}\underline{\boldsymbol{R}}(\boldsymbol{A})^\top \right\|_F^2 \tag{3.17}$$

$$= \sum_{l=1}^{t} \sum_{i=1}^{n} \sum_{j=1}^{n} \left(s_{lij} - \left(\boldsymbol{A}\boldsymbol{R}_l(\boldsymbol{A})^\top \right)_{ij} \right)^2 \tag{3.18}$$

$$= \sum_{l=1}^{t} \mathcal{L}(\boldsymbol{S}_l, \boldsymbol{A}, \boldsymbol{R}_l) \tag{3.19}$$

with

$$\left(\boldsymbol{A}\boldsymbol{R}_l(\boldsymbol{A})^\top\right)_{ij} = \sum_{b=1}^{k}\sum_{c=1}^{k} a_{ib} r_{lbc} a_{jc}. \tag{3.20}$$

Note that in this framework, the DEDICOM algorithm described in the previous section is equivalent to tensor DEDICOM with $t = 1$.

Update steps can then be taken via alternating gradient descent on $\boldsymbol{A}$ and $\underline{\boldsymbol{R}}$. As in the previous section, one can now add additional constraints to $\boldsymbol{A}$ and $\underline{\boldsymbol{R}}$ and calculate the gradients as in (3.11), using automatic differentiation methods. Taking update steps of size $\eta^{\boldsymbol{A}}$ and $\eta^{\underline{\boldsymbol{R}}}$ respectively leads to an eventual convergence to some local or global minimum of the loss (3.17) with respect to the original or constrained $\boldsymbol{A}$ and $\underline{\boldsymbol{R}}$.

Alternatively, constraints can be added to $\boldsymbol{A}$ and $\underline{\boldsymbol{R}}$ by methods like projected gradient descent and the Frank-Wolfe algorithm [76] which either adjust the respective matrix or tensor to be constrained after the gradient step or apply the general gradient step such that the matrix or tensor never leaves the constrained area.

However, empirical results show that automatic differentiation methods lead to slow and unstable training convergence and worse qualitative results when applying the mentioned constraints on the factor matrices and tensors. We therefore derive an alternative method of applying alternating gradient descent to $\boldsymbol{A}$ and $\underline{\boldsymbol{R}}$ based on multiplicative update rules. This does not only improve training stability and convergence behavior but also leads to better qualitative results (see Section 3.4.3 and Figure 3.4).

We derive the gradients for $\boldsymbol{A}$ and $\underline{\boldsymbol{R}}$ analytically and set the learning rates $\eta^{\boldsymbol{A}}$ and $\eta^{\underline{\boldsymbol{R}}}$ individually for each element (i, j) as $\eta^{\boldsymbol{A}}_{ij}$ for matrix $\boldsymbol{A}$ and for each element (l, i, j) as $\eta^{\underline{\boldsymbol{R}}}_{lij}$ for tensor $\underline{\boldsymbol{R}}$, such that the resulting update step is an element-wise multiplication of the respective matrix or tensor.

We derive the updates for the matrix algorithm first and later extend them for the tensor case. For detailed derivations, refer to Appendix Section A.1. For $\boldsymbol{A}$ we derive the gradient analytically as

$$\frac{\partial \mathcal{L}(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})}{\partial \boldsymbol{A}} = -2\left(\boldsymbol{S}\boldsymbol{A}\boldsymbol{R}^\top + \boldsymbol{S}^\top\boldsymbol{A}\boldsymbol{R} - \boldsymbol{A}\left(\boldsymbol{R}\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{R}^\top + \boldsymbol{R}^\top\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{R}\right)\right). \tag{3.21}$$

Therefore, the update step is

$$a_{ij} \leftarrow a_{ij} + \eta^{\boldsymbol{A}}_{ij} 2\left(\left[\boldsymbol{S}\boldsymbol{A}\boldsymbol{R}^\top + \boldsymbol{S}^\top\boldsymbol{A}\boldsymbol{R}\right]_{ij} - \left[\boldsymbol{A}\left(\boldsymbol{R}\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{R}^\top + \boldsymbol{R}^\top\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{R}\right)\right]_{ij}\right). \tag{3.22}$$

If we now chose $\eta^{\boldsymbol{A}}$ as

$$\eta^{\boldsymbol{A}}_{ij} \coloneqq \frac{a_{ij}}{2[\boldsymbol{A}(\boldsymbol{R}^\top\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{R} + \boldsymbol{R}\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{R}^\top)]_{ij}}, \tag{3.23}$$

the update (3.22) becomes

$$a_{ij} \leftarrow a_{ij}\frac{[\boldsymbol{S}^\top\boldsymbol{A}\boldsymbol{R} + \boldsymbol{S}\boldsymbol{A}\boldsymbol{R}^\top]_{ij}}{[\boldsymbol{A}(\boldsymbol{R}^\top\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{R} + \boldsymbol{R}\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{R}^\top)]_{ij}}. \tag{3.24}$$

For $\boldsymbol{R}$ we derive the gradient analytically as

$$\frac{\partial \mathcal{L}(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})}{\partial \boldsymbol{R}} = -2(\boldsymbol{A}^\top \boldsymbol{S}\boldsymbol{A} - \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^\top \boldsymbol{A}). \tag{3.25}$$

Therefore, the update step is

$$r_{ij} = r_{ij} + \eta_{ij}^{\boldsymbol{R}} 2([\boldsymbol{A}^\top \boldsymbol{S}\boldsymbol{A}]_{ij} - [\boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^\top \boldsymbol{A}]_{ij}). \tag{3.26}$$

Choose

$$\eta_{ij}^{\boldsymbol{R}} \coloneqq \frac{r_{ij}}{2[\boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^\top \boldsymbol{A}]_{ij}}, \tag{3.27}$$

and the update (3.26) becomes

$$r_{ij} \leftarrow r_{ij} \frac{[\boldsymbol{A}^\top \boldsymbol{S}\boldsymbol{A}]_{ij}}{[\boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^\top \boldsymbol{A}]_{ij}}. \tag{3.28}$$

Since $s_{ij} \geq 0$ for all i, j, in both (3.24) and (3.28) each element of the multiplier matrix is positive if both $\boldsymbol{A} \geq 0$ and $\boldsymbol{R} \geq 0$ in all entries. Therefore, initializing both matrices with positive values results in an update step that keeps the elements of $\boldsymbol{A}$ and $\boldsymbol{R}$ positive.

To extend this rule to tensor DEDICOM, consider that the analytical derivatives hold for $\underline{\boldsymbol{R}}$ and $\boldsymbol{A}$ by considering each slice $\boldsymbol{S}_l$ and $\boldsymbol{R}_l$ individually:

$$\frac{\partial \mathcal{L}(\boldsymbol{S}_l, \boldsymbol{A}, \boldsymbol{R}_l)}{\partial \boldsymbol{R}_l} = -2(\boldsymbol{A}^\top \boldsymbol{S}_l \boldsymbol{A} - \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{R}_l \boldsymbol{A}^\top \boldsymbol{A}), \tag{3.29}$$

$$\frac{\partial \mathcal{L}(\boldsymbol{S}_l, \boldsymbol{A}, \boldsymbol{R}_l)}{\partial \boldsymbol{A}} = -2(\boldsymbol{S}_l^\top \boldsymbol{A}\boldsymbol{R}_l + \boldsymbol{S}_l \boldsymbol{A}\boldsymbol{R}_l^\top - \boldsymbol{A}(\boldsymbol{R}_l^\top \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{R}_l + \boldsymbol{R}_l \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{R}_l^\top)). \tag{3.30}$$

Since by (3.19) we have $\mathcal{L}(\underline{\boldsymbol{S}}, \boldsymbol{A}, \underline{\boldsymbol{R}}) = \sum_{l=1}^{t} \mathcal{L}(\boldsymbol{S}_l, \boldsymbol{A}, \boldsymbol{R}_l)$ we can derive the full gradients as

$$\frac{\partial \mathcal{L}(\underline{\boldsymbol{S}}, \boldsymbol{A}, \underline{\boldsymbol{R}})}{\partial \underline{\boldsymbol{R}}} = (\boldsymbol{A}^\top \underline{\boldsymbol{S}}\boldsymbol{A} - \boldsymbol{A}^\top \boldsymbol{A}\underline{\boldsymbol{R}}\boldsymbol{A}^\top \boldsymbol{A}), \tag{3.31}$$

$$\frac{\partial \mathcal{L}(\underline{\boldsymbol{S}}, \boldsymbol{A}, \underline{\boldsymbol{R}})}{\partial \boldsymbol{A}} = \sum_{l=1}^{t} -2(\boldsymbol{S}_l^\top \boldsymbol{A}\boldsymbol{R}_l + \boldsymbol{S}_l \boldsymbol{A}\boldsymbol{R}_l^\top - \boldsymbol{A}(\boldsymbol{R}_l^\top \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{R}_l + \boldsymbol{R}_l \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{R}_l^\top)). \tag{3.32}$$

For $\boldsymbol{A}$ we set $\eta^{\boldsymbol{A}}$ as

$$\eta_{ij}^{\boldsymbol{A}} \coloneqq \frac{a_{ij}}{2\sum_{l=1}^{t} [\boldsymbol{A}(\boldsymbol{R}_l^\top \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{R}_l + \boldsymbol{R}_l \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{R}_l^\top)]_{ij}}. \tag{3.33}$$

Then the update step is

$$a_{ij} \leftarrow a_{ij} - \eta_{ij}^{\boldsymbol{A}} \frac{\partial \mathcal{L}(\underline{\boldsymbol{S}}, \boldsymbol{A}, \underline{\boldsymbol{R}})}{\partial \boldsymbol{A}} \tag{3.34}$$

$$= a_{ij} \frac{\sum_{l=1}^{t} [\boldsymbol{S}_l^\top \boldsymbol{A} \boldsymbol{R}_l + \boldsymbol{S}_l \boldsymbol{A} \boldsymbol{R}_l^\top]}{\sum_{l=1}^{t} [\boldsymbol{A}(\boldsymbol{R}_l^\top \boldsymbol{A}^\top \boldsymbol{A} \boldsymbol{R}_l + \boldsymbol{R}_l \boldsymbol{A}^\top \boldsymbol{A} \boldsymbol{R}_l^\top)]_{ij}}. \tag{3.35}$$

For $\underline{\boldsymbol{R}}$ we again set

$$\eta_{lij}^{\underline{\boldsymbol{R}}} := \frac{r_{lij}}{2[\boldsymbol{A}^\top \boldsymbol{A} \boldsymbol{R}_l \boldsymbol{A}^\top \boldsymbol{A}]_{ij}}, \tag{3.36}$$

and the update (3.26) becomes

$$r_{lij} \leftarrow r_{lij} \frac{[\boldsymbol{A}^\top \boldsymbol{S}_l \boldsymbol{A}]_{ij}}{[\boldsymbol{A}^\top \boldsymbol{A} \boldsymbol{R}_l \boldsymbol{A}^\top \boldsymbol{A}]_{ij}}. \tag{3.37}$$

Equations (3.24) and (3.28) provide multiplicative update rules that ensure the non-negativity of $\boldsymbol{A}$ and $\boldsymbol{R}$ without any additional constraints. Equations (3.34) and (3.37) provide the corresponding rules for matrix $\boldsymbol{A}$ and tensor $\underline{\boldsymbol{R}}$ in tensor DEDICOM.

3.3.3 On Symmetry

The DEDICOM algorithm is able to factorize both symmetrical and asymmetrical matrices $\boldsymbol{S}$. For a given matrix $\boldsymbol{A}$, the symmetry of $\boldsymbol{R}$ dictates the symmetry of the product $\boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^\top$, since

$$(\boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^\top)_{ij} = \sum_{b=1}^{k} \sum_{c=1}^{k} a_{ib} r_{bc} a_{jc} = \sum_{b=1}^{k} \sum_{c=1}^{k} a_{ib} r_{cb} a_{jc} \tag{3.38}$$

$$= \sum_{c=1}^{k} \sum_{b=1}^{k} a_{jc} r_{cb} a_{ib} = (\boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^\top)_{ji} \tag{3.39}$$

iff $r_{cb} = r_{bc}$ for all b, c. We therefore expect a symmetric matrix $\boldsymbol{S}$ to be decomposed into $\boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^\top$ with a symmetric $\boldsymbol{R}$, which is confirmed by our experiments. Factorizing a non-symmetric matrix leads to a non-symmetric $\boldsymbol{R}$. The asymmetric relation between items leads to asymmetric relations between the latent factors. The same relations hold for each slice $\boldsymbol{S}_l$ and $\boldsymbol{R}_l$ in tensor DEDICOM.

3.3.4 On Interpretability

We have

$$s_{ij} \approx \sum_{b=1}^{k} \sum_{c=1}^{k} a_{ib} r_{bc} a_{jc}, \tag{3.40}$$

i.e. we can estimate the probability of co-occurrence of two words w_i and w_j from the word embeddings $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$ and the matrix $\boldsymbol{R}$, where $\boldsymbol{a}_i$ denotes the i-th row of $\boldsymbol{A}$.

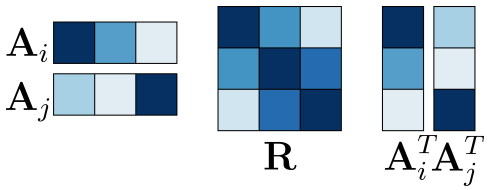


Figure 3.2: The affinity matrix $\boldsymbol{R}$ describes the relationships between the latent factors. Illustrated here are two word embeddings, corresponding to the words w_i and w_j. Darker shades represent larger values. In this example, we predict a large co-occurrence at s_{ii} and s_{jj} because of the large weight on the diagonal of the $\boldsymbol{R}$ matrix. We predict a low co-occurrence at s_{ij} and s_{ji} since the large weights on a_{i1} and a_{j3} interact with low weights on r_{13} and r_{31}.

If we want to predict the co-occurrence between words w_i and w_j we consider the latent topics that make up the word embeddings $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$, and sum up each component from $\boldsymbol{a}_i$ with each component $\boldsymbol{a}_j$ with respect to the relationship weights given in $\boldsymbol{R}$.

Two words are likely to have a high co-occurrence if their word embeddings have larger weights in topics that are positively connected by the $\boldsymbol{R}$ matrix. Likewise, a negative entry r_{bc} makes it less likely for words with high weight in the topics b and c to occur in the same context. See Figure 3.2 for an illustrated example.

Having an interpretable embedding model provides value beyond the analysis of the affinity matrix of a single document. The worth of word embeddings is generally measured in their usefulness for downstream tasks. Given a prediction model based on word embeddings as one of the inputs, further analysis of the model behavior is facilitated when latent input dimensions easily translate to semantic meaning.

In most word embedding models, the embedding vector of a single word is not particularly useful in itself. The information only lies in its relationship (i.e. closeness or cosine similarity) to other embedding vectors. For example, an analysis of the change of word embeddings and therefore the change of word meaning within a document corpus (for example, a news article corpus) can only show how various words form different clusters or drift apart over time. Interpretability of latent dimensions would provide tools to also consider the development of single words within the given topics.

All considerations above hold for the three-dimensional tensor case, in which we analyze a slice $\boldsymbol{R}_l$ together with the common word embedding matrix $\boldsymbol{A}$ to gain insight into the input data slice $\boldsymbol{S}_l$.

3.4 Experiments and Results

In the following section we describe our experimental setup in full detail[2] and present our results on the simultaneous topic (relation) extraction and word embedding learning task. We compare these results against competing matrix and tensor factorization methods for topic

[2] Our `Python` implementation to reproduce the results is available on `https://github.com/LarsHill/text-dedicom-paper`. Additionally, we provide a snapshot of our versions of the applied public datasets (Wikipedia articles and Amazon reviews).

modeling, namely NMF (including a Tucker-2 variation compatible with tensors), LDA, and SVD.

3.4.1 Data

We conduct our experiments on three orthogonal text datasets, which cover different text domains and allow for a thorough empirical analysis of our proposed methods.

The first corpus leverages triplets of individual Wikipedia articles. The articles are retrieved as raw text via the official Wikipedia API using the `wikipedia-api` library. We differentiate between thematically similar (e.g., "Dolphin" and "Whale") and thematically different articles (e.g., "Soccer" and "Donald Trump"). Each article triplet is categorized into one of three classes: All underlying Wikipedia articles are thematically different, two articles are thematically similar and one is different, and all articles are thematically similar. [36] contains an extensive evaluation over 12 triples of articles in the supplementary material. In this chapter, we focus on the three triples described in the previous main paper, namely

1. "Soccer", "Bee", "Johnny Depp",
2. "Dolphin", "Shark", "Whale", and
3. "Soccer", "Tennis", "Rugby".

Depending on whether the article triplets are represented as input matrix or tensor they are processed differently. In the case of a matrix input, all three articles get concatenated to form a new artificially generated document. In the case of a tensor input the articles remain individual documents which later represent slices in the tensor representation.

To analyze the topic extraction capability of constrained DEDICOM also on text which is rather prone to grammatical and syntactical errors we utilize a subset of the Amazon review dataset [77]. In particular, we restrict ourselves to the "movie" product category and create a corpus consisting of six text documents holding the concatenated reviews from the following films respectively, "Toy Story 1", "Toy Story 3", "Frozen", "Monsters, Inc.", "Kung Fu Panda", and "Kung Fu Panda 2". Grouping the reviews by movie affiliation enables us to generate a tensor representation of the corpus, which we factorize via non-negative tensor DEDICOM to analyze topic relations across movies. Table 3.1 lists the number of reviews per movie and shows that based on review count, "Kung Fu Panda 1" is the most popular among the six films.

The third corpus represents a complete collection of New York Times news articles ranging from September 1st, 2019 to August 31st, 2020. The articles are taken from the New York Times website and cover a wide range of sections (see Table 3.3).

Instead of grouping the articles by section, we bin and concatenate them by month, yielding twelve news documents containing monthly information (see Table 3.2 for details on the article count per month). Thereby, the factorization of tensor DEDICOM allows for an analysis of topic relations and their changes over time.

Before transforming the text documents to matrix or tensor representations we apply the following textual preprocessing steps. First, the whole text gets lower-cased. Second, we tokenize the text making use of the word-tokenizer from the `nltk` library and remove common English stop words, including contractions such as "you're" and "we'll". Lastly, we clear the

Table 3.1: Amazon Movie Review corpus grouped by movie and number of reviews per slice of input tensor.

Movie	# Reviews
Toy Story 1	2491
Monsters, Inc.	3203
Kung Fu Panda 1	6708
Toy Story 3	1209
Kung Fu Panda 2	1208
Frozen	1292

Table 3.2: New York Times news corpus grouped by month and number of articles. This corresponds to the number of articles per slice of input tensor.

Month	# Articles
Sep 19	1586
Oct 19	1788
Nov 19	1623
Dec 19	1461
Jan 20	1725
Feb 20	1602
Mar 20	1937
Apr 20	1712
May 20	1713
Jun 20	1828
Jul 20	1814
Aug 20	1886

Table 3.3: New York Times news corpus composition by section and number of articles.

Section	# Articles
Politics	3204
U.S.	2610
Business	1624
New York	1528
Europe	988
Asia Pacific	839
Health	598
Technology	551
Middle East	443
Science	440
Economy	339
Elections	240
Climate	239
World	233
Africa	124
Australia	113
Canada	104

text from all remaining punctuation and delete digits, single characters, and multi-spaces (see Table 3.4 for an overview of corpora statistics after preprocessing).

Next, we utilize all preprocessed documents in a corpus to extract a fixed-size vocabulary of $n = 10\,000$ most frequent tokens. Since our dense input tensor is of dimensionality $t \times n \times n$, a larger vocabulary size leads to a significant increase in memory consumption. Based on the total number of unique corpora words reported in Table 3.4, a maximum vocabulary size of $n = 10\,000$ is reasonable for the three Wikipedia corpora and the Amazon reviews corpus. Only the New York Times dataset could potentially benefit from a larger vocabulary size.

Based on this vocabulary a symmetric word co-occurrence matrix gets calculated for each of the corpus documents. When generating the matrix we only consider context words within a symmetrical window around the base word. Analysis in [53] and [36] shows that the window size in the range of 6 to 10 has little impact on performance. Thus, following our implementation in [36], we choose a window size of 7, the default in the original glove implementation. As in [53], each context word only contributes $1/d$ to the total word pair count, given it is d words apart from the base word. To avoid any bias or prior information from the structure and order of the concatenated Wikipedia articles, reviews, or news articles, we randomly shuffle the vocabulary before creating the co-occurrence matrix. As described in Section 3.3 we then transform the co-occurrence matrix to a positive PMI matrix. If the corpus consists of just one document the generated PPMI matrix functions as input for the row-stochastic DEDICOM algorithm. If

Table 3.4: Overview of word count statistics after preprocessing for all datasets. Columns represent from left to right the total number of words per corpus, the total number of unique words per corpus, the average number of total words per article, the average number of unique words per article, and the cutoff frequency of the 10 000th most common word. Wikipedia article combinations: DSW (Dolphin, Shark, Whale), SBJ (Soccer, Bee, Johnny Depp), STR (Soccer, Tennis, Rugby).

	Total	Unique	Avg. Total	Avg Unique	Cutoff
Amazon Reviews	252 400	15 560	15.2	13.4	1
Wikipedia DSW	14 500	4376	4833.3	2106.0	1
Wikipedia SBJ	10 435	4034	3478.3	1600.3	1
Wikipedia STR	11 501	3224	3833.7	1408.0	1
New York Times	12 043 205	141 591	582.5	366.5	118

the corpus consists of several documents (e.g., one news document per month) the individual PPMI matrices get stacked to a tensor, which in turn represents the input for the non-negative tensor DEDICOM algorithm.

The next section sheds light upon the training process of row-stochastic DEDICOM, non-negative tensor DEDICOM, and the above-mentioned competing matrix and tensor factorization methods, which will be benchmarked against our results in Section 3.4.3 and in the appendix.

3.4.2 Training

As thoroughly outlined in Section 3.3 we train both, row-stochastic DEDICOM and non-negative tensor DEDICOM with the alternating gradient descent paradigm.

In the case of a matrix input and a row-stochasticity constraint on $\boldsymbol{A}$ we utilize automatic differentiation from the `PyTorch` library to perform update steps on $\boldsymbol{A}$ and $\boldsymbol{R}$. First, we initialize the factor matrices $\boldsymbol{A} \in \mathbb{R}^{n \times k}$ and $\boldsymbol{R} \in \mathbb{R}^{k \times k}$, by randomly sampling all elements from a uniform distribution centered around 1, $\mathcal{U}(0, 2)$. Note that after applying the softmax operation on $\boldsymbol{A}$ all rows of $\boldsymbol{A}$ are stochastic. Therefore, scaling $\boldsymbol{R}$ by

$$\bar{s} \coloneqq \frac{1}{n^2} \sum_{ij}^{n} s_{ij}, \tag{3.41}$$

will result in the initial decomposition $\boldsymbol{A}'\boldsymbol{R}(\boldsymbol{A}')^\top$ yielding reconstructed elements in the range of $\bar{s}$, the element mean of the PPMI matrix $\boldsymbol{S}$, and thus, speeding up convergence. Second, $\boldsymbol{A}$ and $\boldsymbol{R}$ get iteratively updated employing the Adam optimizer [78] with constant individual learning rates of $\eta^{\boldsymbol{A}} = 0.001$ and $\eta^{\boldsymbol{R}} = 0.01$ and hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1 \times 10^{-8}$. Both learning rates were identified through an exhaustive grid search. We train for $\texttt{num\_epochs} = 15,000$ until convergence, where each epoch consists of an alternating gradient update with respect to $\boldsymbol{A}$ and $\boldsymbol{R}$. Algorithm 1 illustrates the just-described training procedure.

In the case of a tensor input and an additional non-negativity constraint on $\underline{\boldsymbol{R}}$ we noticed an inferior training performance with automatic differentiation methods. Hence, due to faster

Algorithm 1 The row-stochastic DEDICOM algorithm

initialize $\boldsymbol{A}, \boldsymbol{R} \leftarrow U(0,2) \cdot \bar{s}$ ▷ See Equation (3.41) for the definition of $\bar{s}$
initialize β_1, β_2, ϵ ▷ Adam algorithm hyperparameters
initialize $\eta^{\boldsymbol{A}}$, $\eta^{\boldsymbol{R}}$ ▷ Individual learning rates

for i in $1, \ldots,$ `num_epochs` **do**
    Calculate loss $\mathcal{L} = \mathcal{L}(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R})$ ▷ See Equation (3.8)
    $\boldsymbol{A} \leftarrow\!\shortmid \boldsymbol{A} - \mathrm{Adam}_{\beta_1,\beta_2,\epsilon}(\nabla_{\boldsymbol{A}}, \eta^{\boldsymbol{A}})$, where $\nabla_{\boldsymbol{A}} = \dfrac{\partial \mathcal{L}}{\partial \boldsymbol{A}}$
    $\boldsymbol{R} \leftarrow\!\shortmid \boldsymbol{R} - \mathrm{Adam}_{\beta_1,\beta_2,\epsilon}(\nabla_{\boldsymbol{R}}, \eta^{\boldsymbol{R}})$, where $\nabla_{\boldsymbol{R}} = \dfrac{\partial \mathcal{L}}{\partial \boldsymbol{R}}$
return $\boldsymbol{A}'$ and $\boldsymbol{R}$, where $\boldsymbol{A}' = \mathrm{row\_softmax}\left(\mathrm{col\_norm}\left(\boldsymbol{A}\right)\right)$ ▷ See Equation (3.6)

Algorithm 2 The non-negative tensor DEDICOM algorithm

initialize $\boldsymbol{A}, \underline{\boldsymbol{R}} \leftarrow U(0,2)$
scale $\boldsymbol{A}$ by $\bar{\alpha}$ and $\boldsymbol{R}_l$ by α_l ▷ See Equation (3.42) for the definitions of $\bar{\alpha}$ and α_l

for i in $1, \ldots,$ `num_epochs` **do**
    Calculate loss $\mathcal{L} = \mathcal{L}(\underline{\boldsymbol{S}}, \boldsymbol{A}, \underline{\boldsymbol{R}})$ ▷ See Equation (3.18)
    $a_{ij} \leftarrow\!\shortmid a_{ij} \dfrac{\left[\sum_{l=1}^{t} \left(\boldsymbol{S}_l \boldsymbol{A} \boldsymbol{R}_l^\top + \boldsymbol{S}_l^\top \boldsymbol{A} \boldsymbol{R}_l\right)\right]_{ij}}{\left[\boldsymbol{A} \sum_{l=1}^{t} \left(\boldsymbol{R}_l \boldsymbol{A}^\top \boldsymbol{A} \boldsymbol{R}_l^\top + \boldsymbol{R}_l^\top \boldsymbol{A}^\top \boldsymbol{A} \boldsymbol{R}_l\right)\right]_{ij}}$
    $r_{lij} \leftarrow\!\shortmid r_{lij} \dfrac{\left[\boldsymbol{A}^\top \boldsymbol{S}_l \boldsymbol{A}\right]_{ij}}{\left[\boldsymbol{A}^\top \boldsymbol{A} \boldsymbol{R}_l \boldsymbol{A}^\top \boldsymbol{A}\right]_{ij}}$
return $\boldsymbol{A}$ and $\underline{\boldsymbol{R}}$

and more stable training convergence and improved qualitative results, we update $\boldsymbol{A}$ and $\underline{\boldsymbol{R}}$ iteratively via derived multiplicative update rules enforcing non-negativity. Again, we initialize $\boldsymbol{A} \in \mathbb{R}^{n \times k}$ and $\underline{\boldsymbol{R}} \in \mathbb{R}^{t \times k \times k}$ by randomly sampling all elements from a uniform distribution centered around 1, $\mathcal{U}(0,2)$. To ensure that the initialized components yield a reconstructed tensor whose elements are in the same range of the input, we calculate an appropriate scaling factor for each tensor slice $\boldsymbol{S}_l$ as

$$\alpha_l \coloneqq \left(\frac{\bar{s}_l}{k^2}\right)^{\frac{1}{3}}, \quad \text{where} \quad \bar{s}_l \coloneqq \frac{1}{n^2} \sum_{ij}^{n} s_{lij}. \tag{3.42}$$

Next, we scale $\boldsymbol{A}$ by $\bar{\alpha} = \frac{1}{t} \sum_{l=1}^{t} \alpha_l$ and each slice $\boldsymbol{R}_l$ by α_l before starting the alternating multiplicative update steps for `num_epochs` $= 300$. The detailed derivation of the update rules is found in Section 3.3.2 and their iterative application in the training process is described in Algorithm 2.

We implement NMF, LDA, and SVD using the `sklearn` library. In all cases, the learnable factor matrices are initialized randomly and default hyperparameters are applied during training.

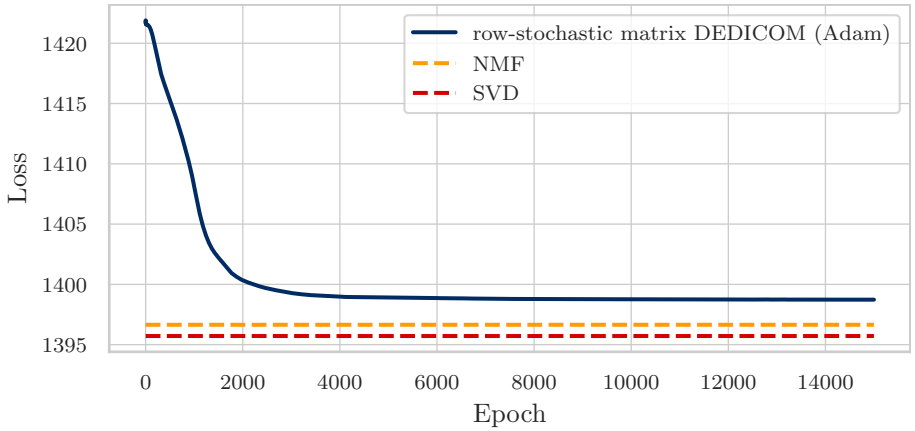


Figure 3.3: Reconstruction loss development during matrix factorization training. The x-axis plots the number of epochs, and the y-axis plots the corresponding reconstruction error for each method.

For NMF the multiplicative update rule from [48] is utilized.

Figure 3.3 shows the convergence behavior of the row-stochastic matrix DEDICOM training process and the final loss of NMF and SVD. Note that LDA optimizes a different loss function, which is why the calculated loss is not comparable and therefore excluded. We see that the final loss of DEDICOM is located just above the other losses, which is reasonable when considering the row stochasticity constraint on $\boldsymbol{A}$ and the reduced parameter amount of $nk + k^2$ compared to NMF ($2nk$) and SVD ($2nk + k^2$).

To also have a benchmark model for our constrained tensor DEDICOM methods to compare against, we implement a Tucker-2 variation of NMF, named tensor NMF (TNMF), which factorizes the input tensor $\underline{\boldsymbol{S}}$ as

$$\boldsymbol{S}_l \approx \underline{\boldsymbol{W}} \boldsymbol{h}_l. \tag{3.43}$$

Its training procedure follows closely the above-described alternating gradient descent approach for non-negative tensor DEDICOM. However, due to the two-way factorization (three-way for DEDICOM) the scaling factor α_l to properly initialize $\underline{\boldsymbol{W}}$ and $\boldsymbol{H}$ has to be adapted to

$$\alpha_l := \left(\frac{\bar{s}_l}{k}\right)^{\frac{1}{2}}, \quad \text{where} \quad \bar{s}_l := \frac{1}{n^2} \sum_{ij}^{n} s_{lij}. \tag{3.44}$$

Analogous to Figure 3.3 we compare the training stability and convergence speed of our implemented tensor factorization methods. In particular, Figure 3.4 visualizes the reconstruction loss development for non-negative tensor DEDICOM trained via multiplicative update rules, row-stochastic tensor DEDICOM trained with automatic differentiation and the Adam optimizer, and tensor NMF. It can be observed that row-stochastic tensor DEDICOM converges much slower than the other two models trained with multiplicative update rules (learning rates are implicit here and don't have to be tuned).

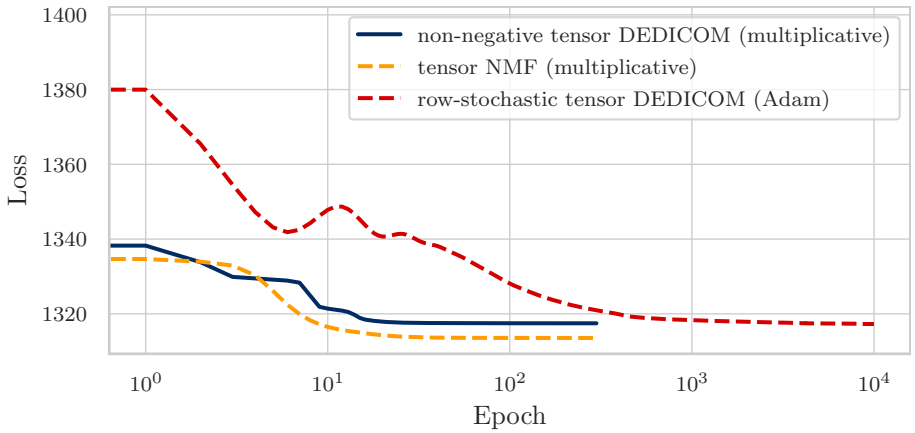


Figure 3.4: Reconstruction loss development during tensor factorization training. The x-axis plots the number of epochs on a logarithmic scale, and the y-axis plots the corresponding reconstruction error for each method.

3.4.3 Results

In the following, we present our results of training the above-mentioned constrained DEDICOM factorizations on different text corpora to simultaneously learn interpretable word embeddings and meaningful topic clusters and their relations.

First, we focus our analysis on row-stochastic matrix DEDICOM applied to the synthetic Wikipedia text documents described in Section 3.4.1. For compactness reasons we primarily consider the document, "Soccer, Bee and Johnny Depp", set the number of topics to $k = 6$ and refer to Appendix A.2.1 for the other article combinations and competing matrix factorization results. Second, we extend our evaluation to the tensor representation of the Wikipedia documents ($t = 3$, one article per tensor slice) and compare the performance of non-negative (multiplicative updates) and row-stochastic (Adam updates) tensor DEDICOM. Lastly, we apply non-negative tensor DEDICOM to the binned Amazon movie and New York Times news corpora to investigate topic relations across movies and over time. We again point the interested reader to Appendix A.2 for additional results and the comparison to tensor NMF.

In the first step, we evaluate the quality of the learned latent topics by assigning each word embedding $\boldsymbol{a}_i' \in \mathbb{R}^{1\times k}$ to the latent topic dimension that represents the maximum value in $\boldsymbol{a}_i'$, e.g.,

$$\boldsymbol{a}_i' = \begin{bmatrix} 0.05 & 0.03 & 0.02 & 0.14 & 0.70 & 0.06 \end{bmatrix}, \quad \operatorname{argmax}\left(\boldsymbol{a}_i'\right) = 5, \tag{3.45}$$

and thus, $\boldsymbol{a}_i'$ gets matched to Topic 5. Next, we decreasingly sort the words within each topic based on their matched topic probability. Table 3.5 shows the overall number of allocated words and the resulting top 10 words per topic together with each matched probability.

As indicated by the high assignment probabilities, one can see that columns 1, 2, 4, 5, and 6 represent distinct topics, which easily can be interpreted. Topic 1 and 4 are related to soccer, where 1 focuses on the game mechanics and 4 on the organizational and professional aspects of the game. Topic 2 and 6 refer to Johnny Depp, where 2 focuses on his acting career and 6

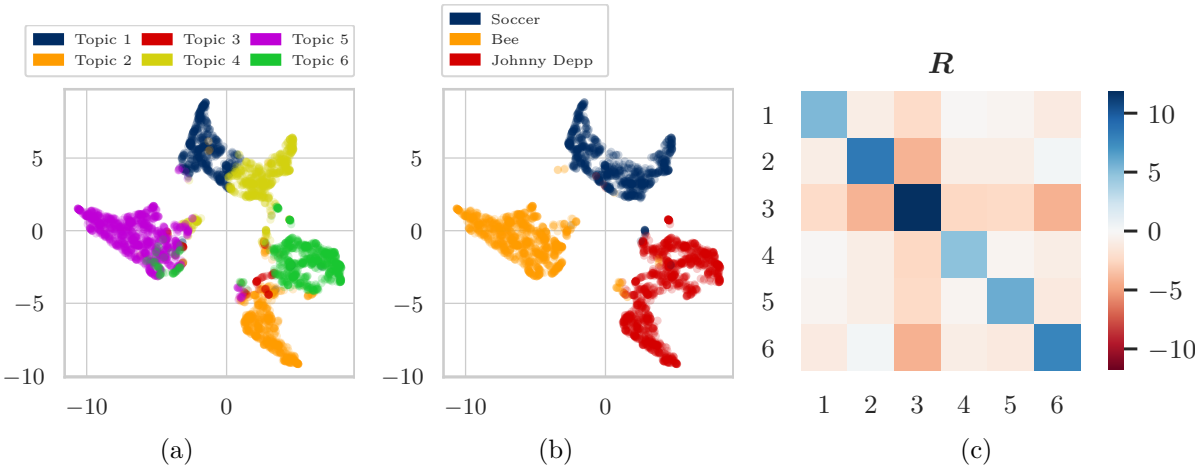


Figure 3.5: (a) 2-dimensional representation of word embeddings $\boldsymbol{A}'$ colored by topic assignment. (b) 2-dimensional representation of word embeddings $\boldsymbol{A}'$ colored by original Wikipedia article assignment (words that occur in more than one article are excluded). (c) Colored heatmap of affinity matrix $\boldsymbol{R}$.

on his difficult relationship with Amber Heard. The fifth topic obviously relates to the insect "bee". In contrast, Topic 3 does not allow for any interpretation and all assignment probabilities are significantly lower than for the other topics.

Further, we analyze the relations between the topics by visualizing the trained $\boldsymbol{R}$ matrix as a heatmap (see Figure 3.5(c)).

One thing to note is the symmetry of $\boldsymbol{R}$ which is a first indicator of a successful reconstruction $\boldsymbol{A}'\boldsymbol{R}(\boldsymbol{A}')^\top$ (see Section 3.3.3). Also, the main diagonal elements are consistently blue (positive), which suggests a high distinction between the topics. Although not very strong one can still see a connection between Topic 2 and 6 indicated by the light blue entry $r_{26} = r_{62}$. While the suggested relation between Topic 1 and 4 is not clearly visible, element $r_{14} = r_{41}$ is the least negative one for Topic 1. To visualize the topic cluster quality we utilize UMAP (Uniform Manifold Approximation and Projection) [79] to map the k-dimensional word embeddings to a 2-dimensional space. Figure 3.5(a) illustrates this low-dimensional representation of $\boldsymbol{A}'$, where each word is colored based on the above-described word-to-topic assignment. In conjunction with Table 3.5 one can nicely see that Topics 2 and 6 (Johnny Depp) and Topics 1 and 4 (Soccer) are close to each other. Hence, Figure 3.5(a) implicitly shows the learned topic relations as well.

As an additional benchmark, Figure 3.5(b) plots the same 2-dimensional representation, but now each word is colored based on the original Wikipedia article it belonged to. Words that occur in more than one article are not considered in this plot.

Directly comparing Figure 3.5(b) and 3.5(a) shows that row-stochastic DEDICOM does not only recover the original articles but also finds entirely new topics, which in this case represent subtopics of the articles. Let us emphasize that for all thematically similar article combinations, the found topics are usually not subtopics of a single article, but rather novel topics that might span across multiple Wikipedia articles (see, for example, Table A.2 in the appendix). As mentioned at the top of this section, we are not only interested in learning meaningful topic

Table 3.5: The top 10 representative words per dimension of the basis matrix $\boldsymbol{A}'$, trained on the wikipedia data as input matrix using automatic gradient methods.

	Topic 1 #619	Topic 2 #1238	Topic 3 #628	Topic 4 #595	Topic 5 #612	Topic 6 #389
1	ball (0.77)	film (0.857)	salazar (0.201)	cup (0.792)	bees (0.851)	heard (0.738)
2	penalty (0.708)	starred (0.613)	geoffrey (0.2)	football (0.745)	species (0.771)	court (0.512)
3	may (0.703)	role (0.577)	rush (0.2)	fifa (0.731)	bee (0.753)	depp (0.505)
4	referee (0.667)	series (0.504)	brenton (0.199)	world (0.713)	pollen (0.658)	divorce (0.454)
5	goal (0.66)	burton (0.492)	hardwicke (0.198)	national (0.639)	honey (0.602)	alcohol (0.435)
6	team (0.651)	character (0.465)	thwaites (0.198)	uefa (0.623)	insects (0.576)	paradis (0.42)
7	players (0.643)	played (0.451)	catherine (0.198)	continental (0.582)	food (0.536)	relationship (0.419)
8	player (0.639)	director (0.45)	kaya (0.198)	teams (0.576)	nests (0.529)	abuse (0.41)
9	play (0.606)	success (0.438)	melfi (0.198)	european (0.57)	solitary (0.513)	stating (0.408)
10	game (0.591)	jack (0.434)	raimi (0.198)	association (0.563)	eusocial (0.505)	stated (0.402)

Table 3.6: For the most significant two words per topic, the four nearest neighbors based on cosine similarity are listed. Matrix $\boldsymbol{A}'$ trained on the wikipedia data as input matrix using automatic gradient methods.

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6
0	ball (1.0)	film (1.0)	salazar (1.0)	cup (1.0)	bees (1.0)	heard (1.0)
1	penalty (0.994)	starred (0.978)	geoffrey (1.0)	fifa (0.995)	bee (0.996)	court (0.966)
2	referee (0.992)	role (0.964)	rush (1.0)	national (0.991)	species (0.995)	divorce (0.944)
3	may (0.989)	burton (0.937)	bardem (1.0)	world (0.988)	pollen (0.986)	alcohol (0.933)
4	goal (0.986)	series (0.935)	brenton (1.0)	uefa (0.987)	honey (0.971)	abuse (0.914)
0	penalty (1.0)	starred (1.0)	geoffrey (1.0)	football (1.0)	species (1.0)	court (1.0)
1	referee (0.999)	role (0.994)	rush (1.0)	fifa (0.994)	bees (0.995)	divorce (0.995)
2	goal (0.998)	series (0.985)	salazar (1.0)	national (0.983)	bee (0.99)	alcohol (0.987)
3	player (0.997)	burton (0.981)	brenton (1.0)	cup (0.983)	pollen (0.99)	abuse (0.982)
4	ball (0.994)	film (0.978)	thwaites (1.0)	world (0.982)	insects (0.977)	settlement (0.978)

clusters but also in training interpretable word embeddings that capture semantic meaning.

Hence, we select within each topic the two most representative words and calculate the cosine similarity between their word embeddings and all other word embeddings stored in $\boldsymbol{A}'$. Table 3.6 shows the 4 nearest neighbors based on cosine similarity for the top 2 words in each topic. We observe a high thematic similarity between words with large cosine similarity, indicating the usefulness of the rows of $\boldsymbol{A}'$ as word embeddings.

In comparison to DEDICOM, other matrix factorization methods also provide a useful clustering of words into topics, with varying degrees of granularity and clarity. However, the application of these methods as word embedding algorithms mostly fails on the word similarity task, with words close in cosine similarity seldom sharing the same thematic similarity we have seen in DEDICOM. This can be seen in Table A.1, which shows for each method, NMF, LDA, and SVD, the resulting word-to-topic clustering and the cosine nearest neighbors of the top two word embeddings per topic. While the individual topics extracted by NMF look very reasonable, its word embeddings do not seem to carry any semantic meaning based on cosine similarity; e.g., the four nearest neighbors of "ball" are "invoke", "replaced", "scores", and "subdivided". A similar nonsensical picture can be observed for the other main topic words. LDA and SVD perform slightly better on the similar word task, although not all similar words appear to be sensible, e.g., "children", "detective", "crime", "magazine" and "barber". Also, some topics cannot be clearly defined due to mixed word assignments, e.g., Topic 4 for LDA and Topic 1 for SVD.

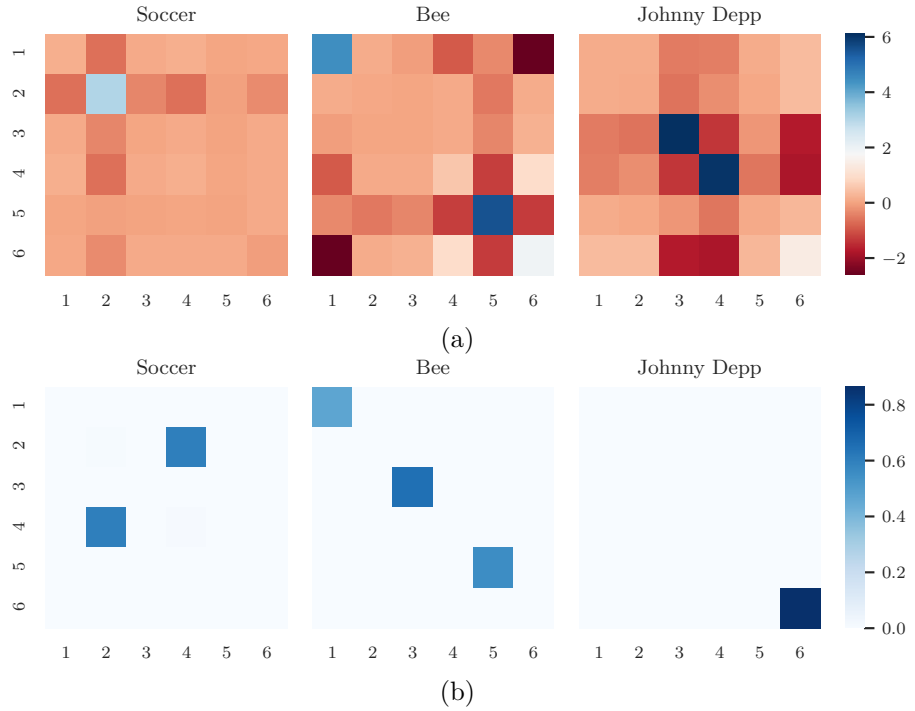


Figure 3.6: (a) and (b) show colored heatmaps of the affinity tensor $\underline{\boldsymbol{R}}$, trained on the Wikipedia data represented as input tensor using different methods: (a) automatic gradient methods, and (b) non-negative multiplicative update rules.

Before shifting our analysis to the Amazon movie review and the New York Times news corpus, we investigate factorizing the tensor representation of the "Soccer, Bee, and Johnny Depp" Wikipedia documents. In particular, we compare the qualitative factorization results of row-stochastic and non-negative tensor DEDICOM trained with automatic differentiation and multiplicative update rules, respectively. Tables 3.7 and 3.8 in conjunction with Figures 3.6(a) and 3.6(b) show the extracted topics and their relations for both methods.

It can be seen that non-negative tensor DEDICOM yields a more interpretable affinity tensor $\underline{\boldsymbol{R}}$ (Figure 3.6(b)) due to its enforced non-negativity. For example, it highlights the bee-related Topics 1, 3, and 5 in the affinity tensor slice corresponding to the article "Bee". Moreover, all extracted topics in Table 3.8 are distinct, and their relations are well represented in the individual slices of $\underline{\boldsymbol{R}}$. In contrast, Topic 6 in Table 3.7 does not represent a meaningful topic, which is also indicated by the low probability scores of the ranked topic words. Although the results of the similar word evaluation are arguably better for row-stochastic tensor DEDICOM (see Tables A.6 and A.7) we prioritize topic extraction and relation quality. That is why in the further analysis of the Amazon review (see Appendix A.2.3) and New York Times news corpus, we restrict our evaluation to non-negative tensor DEDICOM.

Figure 3.7 and Table 3.9 refer to our experimental results on the dataset of New York Times news articles. We see a diverse array of topics extracted from the text corpus, ranging from US politics (Topics 4, 6, 7) to natural disasters (Topic 8), Hollywood sexual assault allegations (Topic 10), and the COVID epidemic both from a medical view (Topic 3) and a view on resulting

Table 3.7: Top 10 representative words per dimension of the basis matrix $\boldsymbol{A}'$, trained on the wikipedia data as input tensor using automatic gradient methods.

	Topic 1 #481	Topic 2 #661	Topic 3 #414	Topic 4 #457	Topic 5 #316	Topic 6 #1711
1	hind (0.646)	game (0.83)	film (0.941)	heard (0.844)	bees (0.922)	disorder (0.291)
2	segments (0.572)	football (0.828)	starred (0.684)	court (0.566)	bee (0.868)	collapse (0.29)
3	bacteria (0.563)	players (0.782)	role (0.624)	divorce (0.51)	honey (0.756)	attrition (0.285)
4	legs (0.562)	ball (0.777)	series (0.562)	depp (0.508)	insects (0.68)	losses (0.284)
5	antennae (0.555)	team (0.771)	burton (0.547)	sued (0.48)	food (0.634)	invertebrate (0.283)
6	females (0.549)	may (0.696)	character (0.499)	stating (0.45)	species (0.599)	rate (0.283)
7	wings (0.547)	play (0.692)	success (0.494)	alcohol (0.449)	nests (0.596)	businesses (0.282)
8	small (0.538)	competitions (0.677)	played (0.483)	paradis (0.446)	flowers (0.571)	virgil (0.282)
9	groups (0.527)	match (0.672)	films (0.482)	alleged (0.445)	pollen (0.56)	iridescent (0.282)
10	males (0.518)	penalty (0.664)	box (0.465)	stated (0.444)	larvae (0.529)	detail (0.281)

Table 3.8: Top 10 representative words per dimension of the basis matrix $\boldsymbol{A}$, trained on the wikipedia data as input tensor using multiplicative update rules.

	Topic 1 #521	Topic 2 #249	Topic 3 #485	Topic 4 #871	Topic 5 #445	Topic 6 #1469
1	species (3.105)	game (3.26)	honey (2.946)	allow (0.668)	insects (2.794)	depp (2.419)
2	eusocial (2.524)	football (3.05)	bee (2.01)	organised (0.662)	pollen (2.239)	film (2.115)
3	solitary (2.279)	players (2.699)	beekeeping (1.933)	winner (0.632)	flowers (2.019)	role (1.32)
4	nest (2.118)	ball (2.475)	bees (1.704)	officially (0.626)	nectar (1.656)	starred (1.3)
5	females (1.993)	may (2.447)	increased (1.589)	wins (0.617)	wasps (1.602)	actor (1.155)
6	workers (1.797)	team (2.424)	humans (1.515)	level (0.613)	wings (1.588)	series (1.126)
7	nests (1.787)	association (1.92)	wild (1.415)	free (0.6)	many (1.588)	burton (1.112)
8	colonies (1.722)	play (1.834)	mites (1.4)	constitute (0.596)	hind (1.577)	played (1.068)
9	egg (1.692)	referee (1.809)	colony (1.35)	regulation (0.595)	hairs (1.484)	heard (1.005)
10	males (1.664)	laws (1.792)	beekeepers (1.332)	prestigious (0.594)	pollinating (1.467)	success (0.981)

restrictions to businesses (Topic 9).

The corresponding heatmap allows us to infer when certain topics were most relevant in the last year. While the entries relating to the COVID pandemic remain light blue for the first half of the heatmap we see the articles picking up on the topic around March 2020, when the effects of the Coronavirus started hitting the US. Even comparatively smaller events like the conviction of Harvey Weinstein and the death of George Floyd triggering the racism debate in the US can be recognized in the heatmap, with a large deviation of Topic 10 around February 2020 and Topic 4 around June 2020.

Further empirical results on the Amazon review and New York Times news corpora, such as two-dimensional UMAP representations of the embedding matrix $\boldsymbol{A}$ and extracted topics from tensor NMF, can be found in Appendix A.2.3 and A.2.4, respectively. For example, Table A.23 shows that the tensor NMF factorization also extracts high-quality topics but lacks the interpretable affinity tensor $\underline{\boldsymbol{R}}$ which is crucial to properly comprehend a topic development over time.

3.5 Conclusion and Outlook

We propose a constrained version of the DEDICOM algorithm that is able to factorize the pointwise mutual information matrices of text documents into meaningful topic clusters all the while providing interpretable word embeddings for each vocabulary item. Our study on semi-artificial data from Wikipedia articles has shown that this method recovers the underlying structure of the text corpus and provides topics with thematic granularity, meaning the extracted latent topics are more specific than a simple clustering of articles. Comparison to related matrix factorization methods has shown that the combination of relation-aware topic modeling and

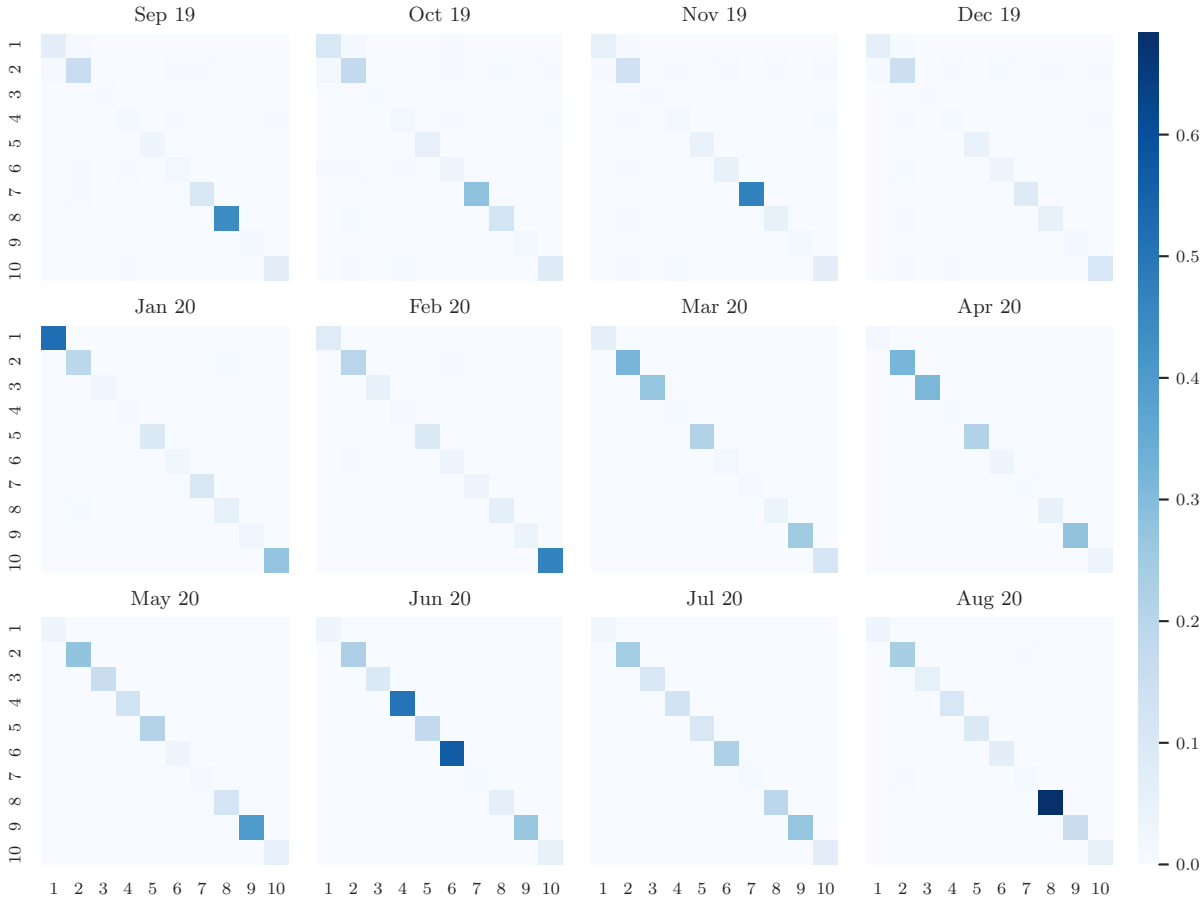


Figure 3.7: Colored heatmap of affinity tensor $\underline{\boldsymbol{R}}$, trained on the New York Times news article data represented as input tensor using multiplicative update rules.

interpretable word embedding learning given by our algorithm is unique in its class.

Extending this algorithm to factorize three-dimensional input tensors allows for the study of changes in the relations between topics across subsets of a structured text corpus, e.g., news articles grouped by time period. Algorithmically, this can be solved via alternating gradient descent by either automatic gradient methods or by applying multiplicative update rules which decrease training time drastically and enhance training stability.

Due to memory constraints from matrix multiplications of high-dimensional dense tensors, our proposed approach is currently limited in vocabulary size or time dimension.

In further work, we aim to develop algorithms capable of leveraging sparse matrix multiplications to avoid the above-mentioned memory constraints. In addition, we plan to expand on the possibilities of constraining the factor matrices and tensors when applying a multiplicative update rule and further analyze the behavior of the factor tensors, for example, by utilizing time series analysis to discover temporal relations between extracted topics and to potentially identify trends. Finally, further analysis may include additional quantitative evaluations of the topic modeling performance of our proposed methods with competing approaches.

Table 3.9: Top 10 representative words per dimension of the basis matrix $\boldsymbol{A}$, trained on the New York Times news article data as input tensor using multiplicative update rules.

	Topic 1 #454	Topic 2 #5984	Topic 3 #567	Topic 4 #562	Topic 5 #424	Topic 6 #330	Topic 7 #515	Topic 8 #297	Topic 9 #431	Topic 10 #436
1	suleimani (2.812)	loans (0.618)	masks (3.261)	floyd (3.376)	contributed (4.565)	confederate (3.226)	ukraine (3.191)	storm (2.76)	restaurants (2.948)	weinstein (3.442)
2	iran (2.593)	university (0.551)	protective (2.823)	minneapolis (2.551)	reporting (2.788)	statue (2.649)	sondland (2.881)	hurricane (2.622)	bars (2.021)	sexual (2.71)
3	iraq (2.453)	oil (0.549)	gloves (2.516)	police (2.255)	michael (2.707)	statues (2.416)	zelensky (2.133)	winds (1.715)	reopen (1.684)	rape (2.102)
4	iranian (2.408)	billion (0.54)	ventilators (2.22)	george (2.088)	katie (2.324)	monuments (1.815)	ambassador (1.976)	tropical (1.606)	gyms (1.654)	assault (1.861)
5	iraqi (1.799)	loan (0.468)	surgical (2.032)	protests (1.936)	alan (2.292)	monument (1.375)	ukrainian (1.789)	storms (1.439)	stores (1.638)	jury (1.513)
6	baghdad (1.604)	bonds (0.466)	gowns (1.965)	brutality (1.765)	emily (2.165)	flag (1.206)	giuliani (1.755)	coast (1.415)	theaters (1.627)	charges (1.409)
7	qassim (1.599)	payments (0.456)	equipment (1.86)	racism (1.579)	nicholas (2.096)	richmond (1.109)	volker (1.754)	laura (1.259)	salons (1.438)	predatory (1.387)
8	strike (1.597)	edited (0.452)	supplies (1.816)	knee (1.435)	cochrane (1.934)	symbols (1.089)	investigations (1.602)	isaias (1.217)	closed (1.424)	harvey (1.35)
9	gen (1.513)	trillion (0.451)	gear (1.742)	killing (1.429)	rappeport (1.613)	remove (1.058)	testified (1.584)	category (1.192)	shops (1.325)	guilty (1.312)
10	maj (1.504)	graduated (0.449)	mask (1.502)	officers (1.405)	maggie (1.529)	removal (1.003)	testimony (1.558)	landfall (1.106)	indoor (1.247)	sex (1.3)

CHAPTER 4

Language Modeling and Contextual Embeddings

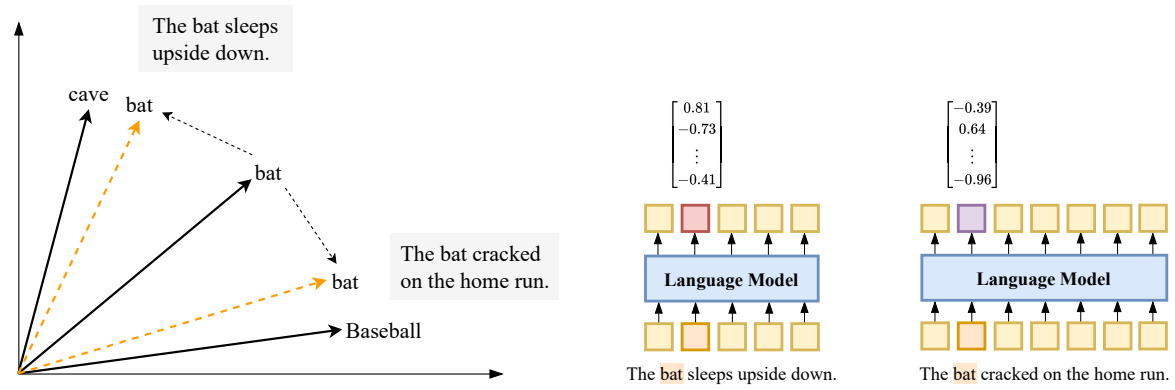
In the previous chapters, we established semantic word embeddings and foundational concepts in NLP, such as text classification, text matching, and topic modeling. Notably, the previous chapter introduced a novel method, row-stochastic DEDICOM, which combines word embedding learning and topic modeling to enhance the interpretability of word embeddings.

However, a significant limitation of the embedding methods so far discussed is their static nature, where each word in the corpus vocabulary is represented by a fixed vector, independent of the surrounding context. This approach fails to account for polysemous words, words with multiple meanings, and those whose meanings are highly context-dependent. For instance, the homonym "bat" can either refer to the flying animal or the equipment used to hit a ball in sports like Baseball, depending on the context.

This chapter addresses these issues by introducing *contextual word embeddings*, which dynamically generate word representations based on their context within a sentence or document. These embeddings are closely linked to *language modeling* [55], where deep neural network architectures capable of sequence modeling are trained on large corpora to learn the relationships, syntactic intricacies, and semantic dependencies of natural language.

Language modeling is a fundamental task in NLP that involves predicting the likelihood of sequences of words. It serves as the cornerstone for developing contextual embeddings, which are crucial for understanding and generating human language. Language models are designed to capture the sequential dependencies of words, which can be intricate and span long distances within a text.

There are various approaches to language modeling, each with its own methodology and applications. *Autoregressive models*, such as those used in RNNs and decoder-only Transformers like GPT (Generative Pre-trained Transformer) [17], focus on predicting the next token in a sequence, thereby modeling the text in a unidirectional manner. In contrast, *denoising* or *bidirectional models*, exemplified by BERT (Bidirectional Encoder Representations from Transformers) [34], employ masked language modeling to predict missing words within a sequence, allowing them to capture context from both directions. These advancements in language modeling have paved the way for the inherent development and improvement of contextual embeddings.



(a) (b)

Figure 4.1: (a) Contextual embeddings resolve the homonymy and polysemy issue. Depending on the surrounding context, the embedding vector for "bat" is closer to the vector for "cave" or "Baseball". (b) Large Language Models (LLMs) output contextual embeddings dynamically at inference time, taking the surrounding context into account.

As illustrated in Figure 4.1, contextual embeddings dynamically adjust based on the context in which a word appears. We see how the word "bat" can have different embedding vectors depending on its usage: when used in the context of "animal" or "cave" its embedding is closer to those words, whereas when used in the context of "Baseball", its embedding aligns more closely with sports-related terms. This dynamic adjustment effectively resolves issues such as homonymy and polysemy.

The remaining structure of this chapter is designed to provide a comprehensive overview of the key concepts and deep neural network architectures used in language modeling, which underpin the creation of contextual embeddings. We begin with an exploration of RNNs, which laid the groundwork for sequential data processing. Following this, we delve into Transformers, which have revolutionized the field with their ability to handle long sequences and parallelize training. Building on these foundational architectures, we introduce the concepts of self-supervised pre-training and transfer learning. These paradigms have become essential in leveraging large-scale datasets to capture the statistics and intricacies of language, enhancing model performance and facilitating domain adaptation.

Throughout these sections, we also present the most influential and prominent architectures for language modeling, namely BERT [34] and GPT [17], which exemplify the bidirectional and autoregressive approaches, respectively. These models have set new benchmarks in various NLP tasks and have been instrumental in advancing the capabilities of language understanding and generation.

Finally, we conclude with a discussion on Retrieval-Augmented Generation (RAG) [35], a cutting-edge approach that combines retrieval mechanisms rooted in text matching (see Section 2.3) with generative models like GPT to produce more reliable and trustworthy responses. This approach addresses common issues such as hallucination [80] in generative language models.

4.1 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) [8] are a class of neural networks designed to process sequential data by maintaining an internal state that evolves over time. This architecture is particularly suited for tasks where the order and context of input data are crucial, such as language modeling and sequence prediction. The fundamental operation of an RNN can be described by the following equations

$$\boldsymbol{h}_t = \sigma\left(\boldsymbol{U}\boldsymbol{h}_{t-1} + \boldsymbol{W}\boldsymbol{x}_t + \boldsymbol{b}_h\right), \tag{4.1}$$

$$\boldsymbol{y}_t = \phi\left(\boldsymbol{V}\boldsymbol{h}_t + \boldsymbol{b}_y\right), \tag{4.2}$$

where $\boldsymbol{h}_t$ is the hidden state at time step t, $\boldsymbol{x}_t$ is the input at time t, $\boldsymbol{U}$, $\boldsymbol{W}$, and $\boldsymbol{V}$ are weight matrices, $\boldsymbol{b}_h$ and $\boldsymbol{b}_y$ are bias vectors, and σ and ϕ are activation functions. The hidden state $\boldsymbol{h}_t$ captures information from all previous inputs, allowing the RNN to model dependencies across time steps.

In the context of textual data, the input $\boldsymbol{x}_t$ represents the embedding of the word at position t. Initially, this embedding can either be randomly initialized or derived from pre-trained static word embeddings. In contrast to the static input representation $\boldsymbol{x}_t$, the hidden state $\boldsymbol{h}_t$ is a dynamically computed contextual representation of the word at time t, capturing the semantic dependencies of all previous words. The input $\boldsymbol{x}_t$ introduces new information at each time step, while the hidden state $\boldsymbol{h}_{t-1}$ retains information from the preceding sequence. The RNN is typically initialized with a zero or random hidden state $\boldsymbol{h}_0$.

One significant shortcoming of this unidirectional approach is that the contextual word embedding at time step t only captures context from the preceding text. To incorporate context from both past and future words, *Bidirectional RNNs* (BiRNNs) are employed. In a BiRNN, two separate RNNs process the sequence: one in the forward direction (from left to right) and another in the backward direction (from right to left). The hidden states from both RNNs are then combined, often by concatenation, to form the final embedding at each time step. This approach enriches the hidden state representation by capturing context from both directions, leading to more informative embeddings.

Despite their ability to model sequential data, standard RNNs suffer from the *vanishing gradient* problem [81], which hampers their ability to learn long-term dependencies. To address this issue, variants such as Long Short-Term Memory (LSTM) networks [82] and Gated Recurrent Units (GRUs) [83] have been developed. These architectures introduce gating mechanisms, e.g., input, output, and forget gates in LSTMs, to control the flow of information, enabling the network to retain or forget information as needed. This allows them to capture dependencies over longer sequences more effectively.

In Chapter 5, we leverage a GRU for sequential text classification, specifically for named entity recognition, to extract KPIs from financial reports.

Since all RNNs are inherently sequential, they process one token at a time. This sequential nature limits their parallelizability and results in computational inefficiency for long sequences. Training RNNs on modern hardware like GPUs, which are optimized for parallel computations, becomes less efficient due to this constraint.

Additionally, RNNs can struggle with capturing long-range dependencies in sequences, even with gating mechanisms, especially in very long texts. These limitations motivate the

exploration of alternative architectures that can handle long sequences more efficiently and capture dependencies over varying lengths without being constrained by sequential processing.

In the following section, we present the Transformer architecture, a state-of-the-art neural network model that addresses many of the limitations of RNNs by utilizing self-attention mechanisms. Transformers enable efficient parallelization during training and effectively capture long-range dependencies, making them highly suitable for language modeling and contextual embedding generation.

4.2 Transformers

Transformers [19] have fundamentally transformed NLP by introducing an architecture that replaces the sequential nature of RNNs with a mechanism known as *self-attention*. This allows the model to process input data in parallel and capture dependencies regardless of their distance in the sequence. Unlike RNNs, which process tokens sequentially and maintain an evolving hidden state, Transformers compute representations of the entire sequence simultaneously.

The core component of the Transformer architecture is the self-attention mechanism, which enables the model to weigh the significance of different tokens when encoding a particular word or symbol. This mechanism relies on transforming the input sequence into queries ($\boldsymbol{Q}$), keys ($\boldsymbol{K}$), and values ($\boldsymbol{V}$). Given an input sequence of token embeddings represented as $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, where n is the sequence length and d is the embedding dimension, the projection layer computes

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}^Q \in \mathbb{R}^{n \times d_k}, \quad \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}^K \in \mathbb{R}^{n \times d_k}, \quad \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}^V \in \mathbb{R}^{n \times d_v}, \tag{4.3}$$

where $\boldsymbol{W}^Q \in \mathbb{R}^{d \times d_k}$, $\boldsymbol{W}^K \in \mathbb{R}^{d \times d_k}$, and $\boldsymbol{W}^V \in \mathbb{R}^{d \times d_v}$ are learnable projection matrices, d_k is the dimension of the keys (and queries) and d_v is the dimension of the values. The attention weights are calculated using the scaled dot-product of the queries and keys which are subsequently projected on the value matrix

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}}\right)\boldsymbol{V} \in \mathbb{R}^{n \times d_v}, \tag{4.4}$$

where d_k is the dimension of the keys (and queries). The scaling factor $1/\sqrt{d_k}$ helps to mitigate the effect of large dot-product values on the softmax function [19].

To enhance the model's ability to focus on different positions and representation subspaces, Transformers employ *multi-head attention*. Instead of utilizing a single attention function, the model computes attention multiple times, known as *heads*, each with its own set of projection matrices $\boldsymbol{W}_i^Q, \boldsymbol{W}_i^K, \boldsymbol{W}_i^V$. The outputs of these h heads are concatenated and projected back to obtain the final multi-head attention result

$$\text{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = [\text{head}_1; \ldots; \text{head}_h]\,\boldsymbol{W}^O \in \mathbb{R}^{n \times d}, \tag{4.5}$$

where each head is defined as $\text{head}_i = \text{Attention}(\boldsymbol{X}\boldsymbol{W}_i^Q, \boldsymbol{X}\boldsymbol{W}_i^K, \boldsymbol{X}\boldsymbol{W}_i^V)$, and $\boldsymbol{W}^O \in \mathbb{R}^{hd_v \times d}$ is a learned projection matrix.

Because Transformers lack the inherent sequential structure of RNNs, positional information is not encoded automatically. To address this, *positional encodings* are added to the input

embeddings to provide the model with information about the position of each token in the sequence. These encodings can be learned or predefined, such as using sinusoidal functions that allow the model to generalize to longer sequences than those seen during training [19].

The original Transformer architecture employs an *encoder-decoder* structure, where both the encoder and decoder are composed of stacked layers containing multi-head attention mechanisms and position-wise feedforward networks. The encoder processes the input sequence to generate contextualized representations, while the decoder generates the output sequence, attending both to the previous outputs and the encoder's representations.

Several variations of the Transformer architecture have been developed to suit specific tasks. One of the most influential *encoder-only* models is BERT [34], which revolutionized the field of NLP. BERT focuses solely on the encoder part of the Transformer, allowing it to learn deep bidirectional representations capturing context from both left and right simultaneously. By leveraging multitask self-supervised pre-training (see Section 4.3), BERT learns rich token-level and sentence-level representations. This enables efficient fine-tuning for a variety of downstream tasks, setting new benchmarks in NLP.

In contrast, *decoder-only* models, like GPT [17, 84], utilize only the decoder part and are geared towards text generation tasks. GPT employs autoregressive next token prediction during pre-training, modeling the probability of a token given its preceding context. This unidirectional approach makes GPT highly effective for generative tasks, and its ability to perform zero-shot and few-shot learning via prompting (see Section 4.4) has further expanded its applicability.

While Transformers excel in capturing long-range dependencies and allow for parallelization during training, they have computational and memory demands that grow quadratically with the sequence length due to the self-attention mechanism's $O(n^2)$ complexity. This limits their scalability for very long sequences. To address this, various approaches aim to reduce the computational burden while maintaining performance to enable longer input sequence lengths. Notable examples include the Longformer [85], which uses local windowed attention mechanisms to reduce complexity, and BigBird [86], which introduces sparse attention patterns.

Furthermore, efficient attention mechanisms like FlashAttention [87] optimize the computation and memory usage of the attention operation itself. Research continues into alternative architectures and methods, such as linear time selective state-space models like Mamba [88], which aim to handle long sequences more effectively.

Transformers have become the foundation of state-of-the-art language models, including large generative models like GPT-4 [18], Mixtral [89], and Gemini [90]. These models demonstrate remarkable capability in generating coherent and contextually appropriate text, answering questions, and performing complex language tasks.

Throughout the remainder of this thesis, we employ Transformer-based models to leverage their strengths in handling sequential data and capturing contextual relationships. Specifically, we utilize BERT-based architectures for sequential text classification and text matching in Chapters 5, 6, 7, and 8. Additionally, we utilize GPT-based architectures for text generation tasks in Chapters 9 and 10.

4.3 Self-Supervised Pre-Training

Self-supervised pre-training has become a cornerstone of modern NLP, enabling models to learn powerful representations of language without the need for extensive labeled datasets. The vast availability of unlabeled textual data on the internet, combined with advancements in computational resources such as powerful GPUs and large-scale memory, has facilitated the training of massive deep neural networks capable of capturing complex linguistic patterns.

The key idea behind self-supervised learning is to generate supervisory signals directly from the data itself through carefully designed pre-training tasks. This approach circumvents the need for manual annotation, which is both time-consuming and expensive, thus allowing models to leverage enormous amounts of data to learn semantic, syntactic, and contextual nuances of language.

One of the pioneering methods in self-supervised pre-training is *Next Token Prediction* (NTP), utilized in generative language models like GPT [17, 18, 91]. In NTP, the model learns to predict the next token in a sequence, which inherently requires understanding the prior context to generate plausible continuations. For instance:

"Luke Skywalker ignites his lightsaber and prepares to face Darth [redacted] ..."

Formally, given a sequence of T text tokens $\mathbf{x} = (x_1, x_2, \ldots, x_T)$, where each $x_t \in \mathcal{V}$ and $\mathcal{V}$ represents the vocabulary (commonly ranging from 50 000 to 100 000 tokens), the goal is to model the joint probability $\mathbb{P}(\boldsymbol{x})$. This is achieved by factorizing the joint probability into a product of conditional probabilities using the chain rule:

$$\mathbb{P}(\boldsymbol{x}) = \prod_{t=1}^{T} \mathbb{P}(x_t \mid x_{1:t-1}). \tag{4.6}$$

The model is trained to maximize the likelihood of the observed sequence, which corresponds to minimizing the negative log-likelihood loss

$$\mathcal{L}_{\text{NTP}} = -\sum_{t=1}^{T} \log \mathbb{P}_\theta(x_t \mid x_{1:t-1}). \tag{4.7}$$

In practice, $\mathbb{P}_\theta(x_t \mid x_{1:t-1})$ is commonly modeled with deep neural network architectures like Transformers that consist of billions of learnable parameters θ. By learning to predict each token based on its preceding tokens, the model captures the sequential dependencies necessary for coherent text generation.

Another prominent approach is *Masked Language Modeling* (MLM), utilized by models like BERT [34]. In MLM, a proportion of the input tokens are randomly masked, and the model is tasked with predicting the original tokens based on the surrounding context. This encourages the model to develop a deep understanding of the language structure and the relationships between tokens. For example, given the sentence:

"Luke [redacted] ignites his [redacted] and prepares to face [redacted] Vader."

the model must predict the missing words *Skywalker*, *lightsaber*, and *Darth*, relying on the context provided by the unmasked tokens.

Formally, for each training sequence, a masking function $M(\boldsymbol{x})$ randomly replaces some tokens in $\boldsymbol{x}$ with a special token [MASK], producing a corrupted sequence $\tilde{\boldsymbol{x}}$. The model is then trained to predict the original tokens at the masked positions. The loss function for MLM is defined as

$$\mathcal{L}_{\text{MLM}} = -\sum_{t \in \mathcal{M}} \log \mathbb{P}_\theta(x_t \mid \tilde{\boldsymbol{x}}), \tag{4.8}$$

where $\mathcal{M}$ is the set of masked token positions and θ denotes the model parameters. By predicting the masked tokens, the model learns bidirectional context representations, capturing information from both left and right of the masked positions.

Self-supervised pre-training can also be leveraged to incorporate higher-level contextual understanding, such as paragraph-level coherence. BERT introduced the *Next Sentence Prediction* (NSP) task [34], where the model learns to predict whether a given pair of sentences are consecutive in the original text. This is formulated as a binary classification problem, enhancing the model's ability to understand the relationship between sentences. For example:

✓ **A**: Luke Skywalker is the last Jedi. **B**: He ignites his lightsaber and prepares to face Darth Vader.

✗ **A**: Luke Skywalker is the last Jedi. **B**: Harry Potter fights Lord Voldemort.

In the first pair, Sentence B logically follows Sentence A, whereas in the second pair, the sentences are unrelated. While NSP aims to foster an understanding of discourse coherence, subsequent research has indicated that it might not significantly contribute to downstream task performance and could be too simplistic, potentially allowing the model to exploit superficial cues [92].

To address these limitations, alternative pre-training objectives have been explored. In Chapter 8, we introduce a novel self-supervised pre-training task designed to enhance paragraph-level contextual awareness in language models. This task aims to provide a more robust training signal that encourages models to capture deeper semantic relationships and improve performance on tasks requiring a comprehensive understanding of longer text spans.

Overall, self-supervised pre-training has enabled the development of models that achieve state-of-the-art results across a wide range of NLP tasks by effectively utilizing large-scale unlabeled data to learn rich linguistic representations.

4.4 Transfer Learning: Fine-Tuning and Zero-Shot Learning

Building upon self-supervised pre-training, *transfer learning* has become a pivotal technique in NLP, enabling models to leverage knowledge acquired from large-scale pre-training and adapt it to specific downstream tasks. Transfer learning allows models to generalize to new tasks or domains with limited labeled data, enhancing their applicability and performance across diverse linguistic challenges.

One common approach to transfer learning is *fine-tuning*, where a pre-trained language model is further trained on a target task using labeled data specific to that task. The model adjusts

its parameters to minimize the loss associated with the new task while retaining the language understanding acquired during pre-training. Fine-tuning is particularly effective because it requires significantly less data and computational resources compared to training a model from scratch.

A powerful example of fine-tuning is *instruction tuning*, where foundational LLMs pre-trained on massive text corpora are further fine-tuned to follow instructions and comply with user requests [93]. This process enables models like GPT [91] to better understand and execute a wide range of tasks specified by natural language prompts, enhancing their ability to perform complex tasks based solely on user instructions.

For instance, after instruction tuning, a model can perform tasks such as summarization, translation, or question answering when provided with appropriate instructions. This capability extends to understanding tasks formulated entirely in language, allowing the model to generalize to new tasks and domains without explicit task-specific training data.

In *few-shot learning*, the model is given a limited number of input-output examples (shots) within the prompt to demonstrate the desired task. The model leverages these examples to infer the task's pattern and apply it to new inputs. For example:

Instruction: Identify the character who said the following quotes in the Star Wars universe.

Examples:
Quote: "Do or do not, there is no try."
Character: Yoda

Quote: "It's a trap!"
Character: Admiral Ackbar

Quote: "I find your lack of faith disturbing."
Character:

The model is expected to continue the pattern and provide the character name for the next quote, namely *Darth Vader*, demonstrating its ability to learn from minimal examples.

In *zero-shot learning*, the model is instructed to perform a task solely through natural language prompts without any task-specific examples. This leverages the model's pre-trained knowledge and its ability to follow instructions embedded in the prompt. For example:

Instruction: Identify all the characters mentioned in the following sentence.

Sentence: "Luke Skywalker and Han Solo board the Millennium Falcon."

The model, understanding the task from the instruction, should output: "Luke Skywalker, Han Solo."

These capabilities stem from the extensive pre-training on diverse data and the instruction-following fine-tuning process, which together enable models to perform new tasks by interpreting instructions provided in natural language.

We successfully leverage the fine-tuning paradigm in Chapter 6 of this thesis to adapt a BERT model for text matching tasks. By fine-tuning, we enable the model to assess semantic similarity between pairs of texts effectively. In Chapter 8, we further fine-tune a pre-trained

BERT model for sequential text classification, enhancing its ability to label sequences of tokens for tasks such as named entity recognition.

In the final chapters of Part III, we utilize zero-shot learning by prompting GPT-based models to perform compliance checks on financial text documents and to generate high-quality, trustworthy answers to detailed user queries related to risk and quality assessments. By formulating the tasks through appropriate prompts, we harness the model's capability to generalize to new domains without additional task-specific training.

Overall, transfer learning, through fine-tuning and zero-shot learning, empowers language models to adapt to a wide array of tasks and domains, maximizing the utility of pre-trained models and minimizing the need for extensive labeled data.

4.5 Retrieval-Augmented Generation

While LLMs like GPT have demonstrated remarkable capabilities in generating coherent and contextually relevant text, they are not without limitations. One significant challenge is the phenomenon of *hallucination*, where the model generates plausible-sounding but factually incorrect or nonsensical content [80]. This issue arises because LLMs learn statistical patterns in text data rather than grounded truths, leading them to produce information that may not align with reality, especially when dealing with specific or specialized knowledge.

Another limitation of LLMs is the knowledge cutoff inherent in their training data. Models like GPT-3 and GPT-4 are trained on vast datasets that encompass information available up to a certain point in time. Consequently, they lack awareness of events, data, or developments that occur after their last training update. For instance, an LLM trained up to 2019 would be unaware of events like the COVID-19 pandemic or the latest technological advancements.

Furthermore, many organizations possess extensive proprietary knowledge bases containing confidential or specialized information not available in public datasets. This information is critical for tasks requiring domain-specific expertise, such as legal advisories, financial analyses, or technical support. Since LLMs cannot be trained on private data they have not seen, they are unable to access or utilize this information during their standard operation.

To overcome these limitations and enhance the reliability and accuracy of language models, *Retrieval-Augmented Generation* (RAG) has emerged as a powerful approach [35]. RAG combines the generative capabilities of LLMs with information retrieval systems, enabling models to access and incorporate external knowledge sources at inference time. This integration allows the model to provide up-to-date and contextually relevant information, grounded in the retrieved data, which mitigates hallucination and addresses the knowledge cutoff problem.

In the RAG framework, when a user provides a query or prompt, the system first uses a retrieval component to search over a large corpus of documents, such as databases, knowledge bases, or the internet, to find relevant information. The retrieved documents or passages are then fed into the generative model along with the original query. The model generates a response that is conditioned not only on its pre-trained parameters but also on the supplementary information provided by the retrieval component. This process effectively injects real-time, context-specific knowledge into the model's output.

Consider the following example from the auditing domain where an employee asks:

> "Have there been any significant changes in the company's internal control processes or risk management strategies in 2024 that could impact the accuracy and reliability of financial reporting?"

A standard LLM with a knowledge cutoff before 2024 would be unable to provide the most recent guidelines. Even if it attempts to answer, it might hallucinate or provide outdated information, especially considering that the relevant information is only available in internal sources. However, using RAG, the system can retrieve the latest documents from the company's internal document knowledge base. By incorporating this retrieved information, the language model can generate an accurate, up-to-date, and trustworthy response.

In Chapter 10, we leverage RAG to develop a reliable and accurate chatbot designed for complex queries related to risk and quality assessment within PricewaterhouseCoopers (PwC), a leading global auditing and consulting firm.

Part II

Named Entity Recognition and Text Matching for Financial Document Consistency

CHAPTER 5

Joint KPI-Extraction and Linking for Financial Reports

Building upon the foundational concepts of representation learning in NLP introduced in Part I, we now transition to the specialized domain of financial document analysis. The previous chapters provided a comprehensive exploration of text classification, the evolution of word embeddings, from static to contextual, and advanced architectures such as RNNs and Transformers [19], which underpin today's state-of-the-art NLP models. Notably, we examined how models like BERT [34] and GPT [17] have revolutionized the field by leveraging transfer learning paradigms, enabling applications across a wide range of NLP tasks.

In the following parts of the thesis, we focus on the application of these advanced NLP techniques to improve the analysis of financial documents. Specifically, this chapter introduces a novel approach for the joint extraction and linking of Key Performance Indicators (KPIs) from financial reports. KPIs are quantifiable measures that reflect the critical success factors of an organization, providing insights into financial health, operational efficiency, and strategic progress. Examples of KPIs include revenue growth, net profit margin, return on investment, and earnings per share. Extracting and accurately linking these indicators within and across documents is essential for analysts, investors, and regulatory bodies to assess performance and ensure transparency.

Financial reports are complex documents that often contain KPIs expressed in varied linguistic forms and embedded within intricate narrative contexts. Moreover, semantically equivalent KPIs can be presented differently across sections or even within the same section, posing challenges for automated extraction and consistency verification. Linking these KPIs involves identifying relationships between them, such as hierarchical dependencies, which is crucial for tasks like trend analysis and anomaly detection.

In this chapter, we present *KPI-BERT* a novel methodology that addresses these challenges by leveraging the power of contextual embeddings and multitask learning. Our approach employs a BERT-based encoder model augmented with two classification heads that are trained jointly:

- **Named Entity Recognition Decoder:** Utilizing an RNN-based GRU [83] decoder with conditional label masking, this component efficiently extracts KPIs from sentences by capturing contextual dependencies and ensuring label consistency.

- **Relation Extraction Classifier:** This head predicts the relationships between the extracted KPIs, facilitating the linking process by identifying how different KPIs relate to each other within the document.

The model architecture is designed for end-to-end training, allowing the shared encoder to learn representations that are optimal for both tasks simultaneously. By fine-tuning a pre-trained BERT checkpoint, we leverage transfer learning to adapt the model to the financial domain, enhancing its ability to comprehend domain-specific terminology and structures.

Our research demonstrates that jointly training the extraction and linking tasks improves performance in both areas, as the shared encoder benefits from the combined learning objectives. This approach not only enhances the accuracy of KPI extraction but also improves the reliability of KPI linking, which is vital for subsequent analyses such as consistency checks. In the following chapter, we build upon this work to perform numeric consistency verification, ensuring that semantically equivalent KPIs exhibit consistent numerical values throughout the documents.

This chapter is based on the following publication:

- L. Hillebrand, T. Deußer, T. Dilmaghani, B. Kliem, R. Loitz, C. Bauckhage, and R. Sifa, "KPI-BERT: A Joint Named Entity Recognition and Relation Extraction Model for Financial Reports," *Proc. ICPR*, 2022, DOI: `10.1109/ICPR56361.2022.9956191` [30].

As first author, Lars Hillebrand was responsible for the idea and methodological design of the joint extraction and linking model. He implemented the majority of the codebase, including the training routines using `PyTorch` [94], and led the development of the model architecture. The experiments were conducted collaboratively with Tobias Deußer, with both contributing to the evaluation of results. The writing of the paper was a joint effort between Lars Hillebrand and Tobias Deußer, with Lars Hillebrand focusing on articulating the methodological innovations and experimental findings.

5.1 Introduction

In the context of business administration, KPIs are defined as quantitative measures about structural entities and are usually utilized for facilitating descriptive, comparative, and predictive analysis as well as for informed decision-making [95, 96]. Considering the latter, (semi-)automatically extracting information (e.g., in the form of values or relationships) related to such indicators can give companies competitive advantages due to the time efficiency practitioners gain, especially when analyzing large amounts of data. Recently NLP and ML-based approaches have been deployed to extract KPI-related information from unstructured data, such as financial documents. These approaches have also been used to support financial auditors with certain elementary processes related to analyzing and comparing information from single as well as multiple documents [27]. Although being successfully deployed, these concepts often suffer from either being rule-based and inflexible [97], only considering structured data (i.e. tables) [96], or focusing exclusively on numerical cross-checking [98].

To alleviate these challenges, we present KPI-BERT, an automated system which leverages new methods of Named Entity Recognition (NER) and Relation Extraction (RE) to detect KPIs and their relationships in real-world German financial documents. The described system

is currently being integrated in the auditing process of a major auditing company and promises to achieve significant efficiency gains.

Given the following sentence from a financial statement,

> "In 2021 the revenue$_{\text{kpi}}$ increased to \$100$_{\text{cy}}$ million (prior year: \$80$_{\text{py}}$ million) while the total costs$_{\text{kpi}}$ decreased to \$50$_{\text{cy}}$ million (prior year: \$70$_{\text{py}}$ million)."

it automatically recognizes and classifies the highlighted named entities and links their relations:

> revenue$_{\text{kpi}}$ – 100$_{\text{cy}}$, revenue$_{\text{kpi}}$ – 80$_{\text{py}}$, total costs$_{\text{kpi}}$ – 50$_{\text{cy}}$, total costs$_{\text{kpi}}$ – 70$_{\text{py}}$

where *kpi*, *cy* (current year value), and *py* (prior year value) are defined entity classes explained in Table 5.2. In particular, the system utilizes a BERT-based [34] architecture that novelly combines an RNN with conditional label masking to sequentially tag the above-emphasized entities before it classifies the linked relations. We further improve the setup by employing trainable RNN-based pooling layers, which outperform the established mean- and max-pooling counterparts. The model also incorporates domain expert knowledge into the process. First, it filters impossible relation candidates prior to their classification since not all entity pair combinations are allowed to be linked (see Table 5.1). Second, we post-process the predicted relations by removing overlapping ones based on their prediction probability.

We benchmark our approach against multiple strong baselines, which also build on BERT-encoded word embeddings but utilize different entity tagging schemes, namely state-of-the-art span-based tagging [99], sequential Conditional Random Field (CRF) tagging [100] and standard linear tagging [101]. In addition, we thoroughly investigate the impact of various parameter ablations, including the usage of different word pooling functions. We find that our system outperforms the competing architectures in robustly extracting and relating KPIs within financial reports.

In summary, our contributions are twofold:

- We present a novel system that automatically extracts and links KPIs from financial documents and is actively integrated into the auditing process of a major auditing firm.
- We introduce a new BERT-based architecture that employs a GRU coupled with trainable pooling layers and conditional label masking to successfully address the sequential nature of the KPI extraction task.

In the following, we first review related work and recent advances in NER and RE. Next, Section 5.3 introduces our model, competing baselines, and the corresponding training process. In Section 5.4, we describe our real-world dataset of financial documents and present the experimental setup along with the performance results of all models. We close with concluding remarks and an outlook into conceivable future work.

5.2 Related Work

In this work, we focus on our specific setup of token-level entity tagging combined with conditional label masking to jointly extract entities and relations on a novel corpus of German financial documents and contrast the results with various ablation studies. However, many recent studies have investigated the task of NER [102–105], RE [106–108], and the joint combination of both [109–113][1].

Much effort has been spent on the task of separately recognizing named entities and extracting relations in the past, whose results were then hierarchically combined in a pipeline [114, 115]. The previously mentioned studies showed that learning these tasks jointly can improve performance immensely and thus suggest that insightful information from one task can be exploited by the other. Furthermore, most contemporary models have their foundation in modern pre-trained natural language models [17, 34, 84, 92].

Highlighting a few of these contemporary studies, [99] introduced a model called SpERT and reported state-of-the-art results on various datasets designated for this task. [116] leveraged BERT at its core to implement an end-to-end model on the token-level with feed forward layers for each task, achieving comparable results to [99]. [109] lessened the required annotations during the NER task by introducing a self-training approach and [117] focused on diminishing the computational complexity by utilizing more compact entity embeddings.

Looking at our specific task of retrieving information from financial reports using ML methods, [118] employed a multi-layer perceptron (MLP) to capture interpretable structures similar to accounting ratios. However, the inputs for their MLP were already extracted and transformed accounting variables. A step in the direction of automatically extracting these variables has been made by [97], who developed a NER model with a rule-based approach. The most up-to-date work in this specific field is [98], which leveraged a joint entity and relation extraction approach to cross-check various financial formulas.

With respect to our domain of processing German accounting and financial documents, [119] leveraged contextualized NLP methods to recognize named entities in the context of anonymization. Besides, [27] presented a recommender-based tool that greatly simplifies and to a large extent automates the auditing of financial statements.

5.3 Methodology

Our proposed model comprises three stacked components that we train jointly in an end-to-end fashion via gradient descent. First, a BERT-based encoder embeds the sentence into latent space. Second, a GRU-based named entity NER decoder sequentially classifies entities using conditional label masking along with the prior tagging history. Third, a RE decoder links the predicted entities.

5.3.1 BERT-based Sentence Encoder

Given a WordPiece [120] tokenized input sentence of n subwords we use a pre-trained BERT [34] model to obtain a sequence of $n+1$ encoded token embeddings, $(\boldsymbol{c}, \boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_n)$, where

[1] [101] wrote a more comprehensive article on recent developments in the field of RE and compared their results in depth.

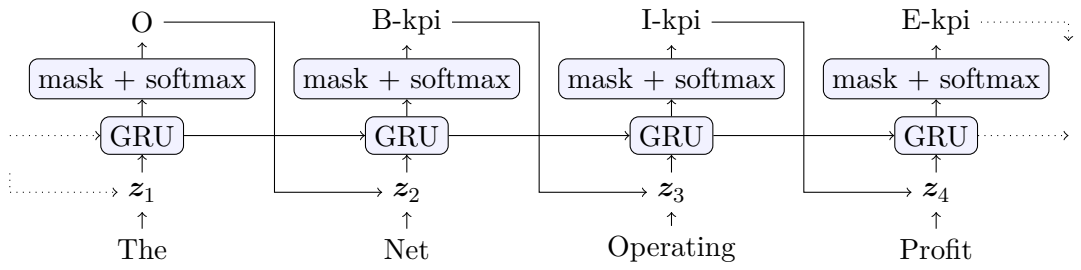


Figure 5.1: Sequential IOBES tagging (each entity class is prepended with the prefixes *I-* (inside), *B-* (begin), *E-* (end) or *S-* (single), while *O* (outside) represents the *none* class) leveraging a Gated Recurrent Unit (GRU) and conditional label masking. $\boldsymbol{z}_j$ represents the concatenation of the previously predicted label embedding with the word embedding at position j. Along with the previous hidden state vector, it gets passed to a GRU that, followed by label masking and a softmax layer, predicts the next IOBES tag.

$\boldsymbol{c} \in \mathbb{R}^d$ represents the context embedding for the whole sentence and $\boldsymbol{t}_i \in \mathbb{R}^d$ represents the token embedding at position i. To easily utilize our word-level entity annotations and to reduce model complexity, we apply a pooling function, $\text{pool}(\cdot)$, which creates word representations by combining their individual subword embeddings. Specifically, the j-th word, consisting of k subwords, is represented as

$$\boldsymbol{e}_j := \text{pool}(\boldsymbol{t}_i, \boldsymbol{t}_{i+1}, \ldots, \boldsymbol{t}_{i+k-1}), \quad \boldsymbol{e}_j \in \mathbb{R}^d. \tag{5.1}$$

While also evaluating max- and mean-pooling we employ a more sophisticated trainable RNN-pooling mechanism building on a bidirectional GRU.

In particular, the subword embedding sequence $(\boldsymbol{t}_i, \boldsymbol{t}_{i+1}, \ldots, \boldsymbol{t}_{i+k-1})$ is passed bidirectionally through a forward and backward GRU, yielding the final hidden states

$$\boldsymbol{h}_j^f = \text{GRU}^f\left(\boldsymbol{t}_i, \boldsymbol{t}_{i+1}, \ldots, \boldsymbol{t}_{i+k-1}\right), \tag{5.2}$$

$$\boldsymbol{h}_j^b = \text{GRU}^b\left(\boldsymbol{t}_{i+k-1}, \ldots, \boldsymbol{t}_{i+1}, \boldsymbol{t}_i\right), \tag{5.3}$$

where $\boldsymbol{h}_j^f, \boldsymbol{h}_j^b \in \mathbb{R}^{d/2}$ and the superscripts $\cdot^f$ and $\cdot^b$ refer to the forward and backward model, respectively. Next, we simply concatenate $\boldsymbol{h}_j^f$ and $\boldsymbol{h}_j^b$ to obtain

$$\boldsymbol{e}_j = \left[\boldsymbol{h}_j^f; \boldsymbol{h}_j^b\right]. \tag{5.4}$$

5.3.2 NER Decoder

Utilizing the BERT-encoded and pooled word embedding sequence $(\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_m)$, a NER decoder module classifies named entities within the sentence. Specifically, we introduce a sequential GRU-based tagger with conditional label masking, which builds on the IOBES annotation scheme. IOBES tagging refers to classifying named entities on a word level by prepending all entity classes with the prefixes *I-* (inside), *B-* (begin), *E-* (end) and *S-* (single). Additionally, *O* (outside) represents the *none* type in this annotation scheme. If we denote $\mathcal{E}$

as the set of possible entity types described in Table 5.2 including the *none* class, the actual number of IOBES entity tags increases to $|\mathcal{E}_{\text{IOBES}}| = 4\,(|\mathcal{E}| - 1) + 1$.

Given the sentence "The Net Operating Profit increased to $ 1.2 million in 2020 .", IOBES tagging aims to predict the following label sequence: "O, B-kpi, I-kpi, E-kpi, O, O, O, S-cy, O, O, O, O".

To take the sequential nature of entity tagging into account we employ a GRU in combination with conditional label masking to sequentially predict IOBES tags considering the past predictions. Figure 5.1 visualizes this decoding scheme, which is described in more detail in the following paragraphs.

First, we define an embedding matrix $\boldsymbol{W}_{\text{label}} \in \mathbb{R}^{|\mathcal{E}_{\text{IOBES}}| \times u}$ that holds learnable u-dimensional embeddings of all IOBES entity types.

Second, we concatenate $\boldsymbol{e}_j$ with $\boldsymbol{w}_{j-1}^{\text{label}}$, which yields the decoding input representation of word j, $\boldsymbol{z}_j = \left[\boldsymbol{e}_j; \boldsymbol{w}_{j-1}^{\text{label}}\right]$, where $\boldsymbol{w}_{j-1}^{\text{label}} \in \mathbb{R}^u$ represents the embedding of the previously predicted IOBES tag. We define $\boldsymbol{w}_0^{\text{label}}$ as the *O* label embedding since using a dedicated begin-of-sequence embedding did not yield improved empirical results.

Third, we feed $\boldsymbol{z}_j$ alongside the previous hidden state $\boldsymbol{h}_{j-1}$ into a GRU, yielding

$$\boldsymbol{h}_j = \text{GRU}\left(\boldsymbol{z}_j, \boldsymbol{h}_{j-1}\right). \tag{5.5}$$

To get IOBES tag posteriors for word j we linearly transform $\boldsymbol{h}_j$ followed by masking out impossible tag predictions and applying softmax:

$$\hat{\boldsymbol{y}}_j = \text{softmax}\left(\text{mask}\left(\boldsymbol{W}_{\text{seq}}\boldsymbol{h}_j + \boldsymbol{b}_{\text{seq}}\right)\right). \tag{5.6}$$

Note that masking is applied conditional on the last predicted tag $\hat{\boldsymbol{y}}_{j-1}$. Specifically, if $\arg\max(\hat{\boldsymbol{y}}_{j-1})$ equals *O* or has prefix *S* or *E* we mask out all entity types with prefix *I* and *E*. Likewise, if $\arg\max(\hat{\boldsymbol{y}}_{j-1})$ starts with *B* or *I* we know the next predicted tag has to be of the same entity type with prefix *I* or *E*. Hence, all other entity types are masked out, which effectively reduces the tagging decision to a binary classification problem.

Next, we convert all predicted IOBES tags and their word embeddings $\boldsymbol{e}_j$ to the entity level by applying the same pooling function as in Equation (5.1). Finally, we concatenate this pooled entity representation with a span size embedding $\boldsymbol{w}_k^{\text{width}}$. It is taken from a dedicated embedding matrix $\boldsymbol{W}_{\text{width}} \in \mathbb{R}^{l \times v}$ holding fixed-size embeddings of dimensionality v for each span length from 1 to l and is learned during training to let the model include a prior over span widths. This gives us the embedding for each entity s

$$\boldsymbol{e}(s) := \left[\text{pool}(\boldsymbol{e}_j, \boldsymbol{e}_{j+1}, \ldots, \boldsymbol{e}_{j+k-1}); \boldsymbol{w}_k^{\text{width}}\right], \quad \boldsymbol{e}(s) \in \mathbb{R}^{d+v}. \tag{5.7}$$

5.3.3 RE Decoder

We only allow for a single relation type between two entities, namely the *matches* relation. This relation is symmetric, as it does not matter whether a KPI is matched to its value or the reverse case of a value being matched to its KPI. Additionally, we refine the entity sampling process to only allow for valid entity pairs. The *relation matrix* shown in Table 5.1 specifies which entity combinations are allowed. Finally, we enforce the uniqueness conditions specified in said

Table 5.1: Comprehensive overview of all allowed relations and their uniqueness. "1:1": One entity of type 1 can only be linked to one entity of type 2, "1:n": One entity of type 1 can be linked to many entities of type 2. "-": No relation possible.

	kpi	cy	py	increase	decrease	davon	davon-cy	davon-py
kpi	-	1:1	1:1	1:1	1:1	1:n	-	-
cy	1:1	-	-	-	-	-	-	-
py	1:1	-	-	-	-	-	-	-
increase	1:1	-	-	-	-	-	-	-
decrease	1:1	-	-	-	-	-	-	-
davon	n:1	-	-	-	-	-	1:1	1:1
davon-cy	-	-	-	-	-	1:1	-	-
davon-py	-	-	-	-	-	1:1	-	-

table after the model has processed the input data. This prunes the results by eliminating relations of two entities if the same combination is predicted with a higher score in the same input sequence and such a combination is labeled as unique, i.e. a 1:1 relation. For instance, if two KPI entities are linked to a singular current year value, we only keep the relation with the higher score and discard the other.

Similar to other studies, we sample candidate pairs s_1 and s_2 from the pool $S \times S$ representing all *allowed* entity combinations in the sentence. Given two entities, we concatenate their respective representations (see Equation (5.7)) with a localized context embedding $\boldsymbol{c}_{\text{loc}}$. Different from the global sentence context $\boldsymbol{c}$, $\boldsymbol{c}_{\text{loc}}$ is defined as the pooled representation[2] of BERT-encoded word embeddings located between s_1 and s_2. As in [99] we find that this localized context embedding is better suited for the relation classification task than the BERT context token $\boldsymbol{c}$. Hence, we define

$$\boldsymbol{x}_r(s_1, s_2) := [\boldsymbol{e}(s_1); \boldsymbol{c}_{\text{loc}}(s_1, s_2); \boldsymbol{e}(s_2)] \tag{5.8}$$

as input for the relation classifier, where $\boldsymbol{x}_r(s_1, s_2) \in \mathbb{R}^{3d+2v}$. Due to our relation type being symmetric, we do not have to classify the inverse $\boldsymbol{x}_r(s_2, s_1)$.

The relation classifier is then defined as

$$\hat{y}_r = \text{sigmoid}\left(\boldsymbol{w}_{\text{rel}}^T \boldsymbol{x}_r(s_1, s_2) + b_{\text{rel}}\right), \tag{5.9}$$

where $\boldsymbol{w}_{\text{rel}} \in \mathbb{R}^{3d+2v}$ and $b_{\text{rel}} \in \mathbb{R}$. If the output of Equation (5.9) exceeds a confidence threshold α, we consider entity s_1 and entity s_2 to match.

5.3.4 Training

We train the above-described model architecture end-to-end, including fine-tuning BERT, by minimizing the joint entity and relation classification loss defined as $\mathcal{L} = \mathcal{L}_{\text{ner}} + \mathcal{L}_{\text{rel}}$, where $\mathcal{L}_{\text{ner}}$

[2] The same pooling function as in Equation (5.1) is applied.

Table 5.2: Description and support of all entity types in the complete dataset, excluding the *none* type.

Entity	Support	Description
kpi	16 849	Key Performance Indicators expressible in numerical and monetary value, e.g. revenue or net sales.
cy	11 498	Current Year monetary value of a KPI .
py	5057	Prior Year monetary value of a KPI.
increase	356	Increase of a KPI from the previous year to the current year.
decrease	230	Decrease of a KPI from the previous year to the current year.
davon	8827	Davon, German for thereof, represents a *subordinate* KPI, i.e. if a KPI is part of another, broader KPI.
davon-cy	8443	Current Year value of a thereof KPI.
davon-py	4382	Prior Year value of a thereof KPI.

denotes the categorical cross-entropy loss over IOBES-tagged entity classes and $\mathcal{L}_{\text{rel}}$ denotes the binary cross-entropy loss over the relation prediction.

For the GRU-based NER decoder, we utilize teacher forcing to foster training convergence and stability. Thus, we embed the annotated ground truth tag and use it to condition the label masking instead of the previously predicted tag.

For the relation classifier, we utilize annotated ground truth relations as positive examples as well and randomly sample N_{rel} negative examples from allowed ground truth entity pairs $\mathcal{S}_{\text{gt}} \times \mathcal{S}_{\text{gt}}$ that don't constitute a labeled relation.

5.4 Experiments

In the following sections, we introduce our custom dataset, describe the training setup and model selection process, and evaluate the results. All experiments are conducted on four Nvidia Tesla V100 GPUs and the model plus training code is implemented in `PyTorch` [94].

5.4.1 Data

Our dataset[3] comprises 500 manually annotated financial documents containing a total of 15 394 sentences and was sourced from the *Bundesanzeiger*[4], a platform hosted by the German department of Justice where companies publish their legally mandated documents.

In the first pre-processing step, the reports are tokenized on a sentence level and subsequently on a word level using the `syntok` `Python` library. Second, we tag monetary numbers and extract their scale (e.g., million) and unit (e.g., $) applying rule-based string matching heuristics. Third, we filter each tokenized report for sentences containing said monetary numbers because our only interest lies in matching KPI entities with their monetary values. Next, we manually generate

[3] We are currently unable to publish the dataset and the accompanying `Python` code because both are developed and used in the context of an industrial project.

[4] `https://www.bundesanzeiger.de/`.

token and span-level annotations that are composed of an entity and a relation part, where the entity annotation signals the type of each span in a sentence and the relation annotation which entity spans are linked together.

The manual annotations were done by a group of six qualified auditors, led by a senior auditing expert. In consultation with them, we defined the entity classes outlined in Table 5.2, which also shows the overall support of each class in the dataset. Throughout the annotation process, the exact entity class definitions were refined in several iterations to adjust for variation and edge cases in the data. Most notably, we paid special attention to distinguish *kpi* and *davon* entities which proved difficult depending on the sentence context. After completing the annotations, the aforementioned senior auditing expert reviewed 50 randomly sampled documents and verified their quality. Due to budget and time constraints, each document was annotated just once by a single auditor. Hence, no inter-annotator agreement metrics can be provided. Although not being entirely free of mistakes, we are confident of the overall annotation quality of the dataset.

We randomly split the pre-processed dataset on a document level into a training, validation, and test set, encompassing 13 835, 821, and 738 sentences each.

5.4.2 Baselines

We compare our proposed model with three competing architectures, which all build on the BERT-based sentence encoder outlined in Section 5.3.1, ensuring a level playing field.

First, we replace the GRU-based NER decoder with a fully connected linear layer that classifies named entities in parallel using the BERT-encoded word embeddings as input. The resulting model was introduced by [101] and functions as a straightforward baseline since it neglects inter-label dependencies when classifying entity tags.

Second, we integrate SpERT [99] in our training framework by utilizing its span-based NER decoder. Span-level entity tagging does not make use of the IOBES annotation scheme but classifies entire word spans at once. For further details, we refer the interested reader to [99]. Our implementation closely follows the original code[5] except for extending the hyperparameter search to our novel Bi-GRU pooling function and including the options to filter overlapping and impossible relations.

Third, we implement a Conditional Random Field (CRF) [42] leveraging Viterbi decoding [121] to classify named entities, which is a popular choice for NER due to its ability to model label dependencies. To ensure a fair comparison with our model, we also incorporate the IOBES label constraints from Section 5.3.2 in the CRF by masking out invalid class transitions in the trainable transition matrix.

5.4.3 Training Setup and Hyperparameter Selection

To determine the best hyperparameter setup for each model we conduct an extensive grid search evaluating various parameter combinations based on the validation set relation classification F_1-score. A relation is considered correct if the spans and the types of both related entities are predicted correctly. Table 5.3 shows all tuned model parameters with their respective ranges of values. The overall best-performing setup on the validation set is highlighted in boldface.

[5] `https://github.com/lavis-nlp/spert`.

Table 5.3: Hyperparameter configurations evaluated by grid search. The best configuration on the validation set is highlighted in boldface. $_{\text{LM}}$ indicates the model using conditional label masking.

Hyperparameter	Configurations
Word-, entity- and context pooling (pool)	**Bi-GRU**, Min, Max
NER decoding	$\mathbf{GRU_{LM}}$, CRF_{LM}, Span, Linear
Conditional label masking (LM)	**True**, False
Dropout	0.0, **0.1**, 0.2, 0.3
Confidence threshold (α)	0.4, **0.5**, 0.6
Filtering impossible relations	**True**, False
Removing overlapping relations	**True**, False
Batch size	**2**, 4, 8
Learning rate	5×10^{-6}, $\mathbf{1 \times 10^{-5}}$, 5×10^{-5}
Weight decay	None, **0.01**, 0.1
Gradient normalization	None, **1.0**

Also, note that the "NER decoding" row effectively discriminates KPI-BERT (GRU_{LM} – GRU with conditional label masking) from the other baselines. For all models, we employ the cased $\text{BERT}_{\text{BASE}}$ sentence encoder, published by the MDZ Digital Library team (dbmdz)[6], which has the same architectural setup as the English $\text{BERT}_{\text{BASE}}$ counterpart: 12 multi-head attention layers with 12 attention heads per layer and 768-dimensional output embeddings. We initialize all trainable parameters randomly from a normal distribution $\mathcal{N}(0, 0.02)$, fix the same random seed of 42 for all training runs, and utilize the AdamW [122] optimizer with a linear warm-up of 10% and a linearly decaying learning rate schedule. Further, we set the width embedding dimension v to 25, the label embedding dimension u to 128 (where applicable), and sample a maximum of $N_{\text{rel}} = 100$ negative relation examples per sentence. In line with Table 5.3 we also evaluate different levels of dropout regularization [123] before the entity and relation classifier and apply weight decay and gradient normalization. In addition, the models train with varying peak learning rates, batch sizes, and prediction thresholds (α). We train each model variation for 20 epochs and determine its best checkpoint via early stopping[7].

5.4.4 Ablation Study

In the process of hyperparameter selection, we paid special attention to certain parameter ablations of KPI-BERT, which are described in Table 5.4. First, we find that conditional label masking boosts its validation set relation F_1-score by 1.07 percentage points, which shows the beneficial influence of including prior knowledge in the form of label dependencies in the classification process. Second, we thoroughly investigate the impact of employing different pooling functions. We find that trainable bidirectional GRU-pooling layers outperform the standard mean- and max-pooling significantly by 1.16 percentage points. Third, we quantify

[6] `https://huggingface.co/dbmdz/bert-base-german-cased`.
[7] KPI-BERT's best validation set relation F_1-score is achieved in epoch 18.

Table 5.4: Ablation study of our tuned KPI-BERT model, applying different pooling functions and removing filtering heuristics and conditional label masking. F_1-scores are reported on the validation set since the model ablations are part of a broader grid search.

Configuration/Ablations	Relation F_1 in %
KPI-BERT	**70.32**
No conditional label masking	69.25
No filtering overlapping relations	69.73
No filtering impossible relations	69.47
No filtering impossible & overlapping relations	69.04
KPI-BERT$_{\text{max pooling}}$	69.16
KPI-BERT$_{\text{mean pooling}}$	69.34

how much our modifications of filtering overlapping and impossible relations improve the model's performance. While both heuristics enhance the relation extraction F_1-score, filtering impossible relations leads to a larger improvement, which is expected considering the simplified relation task depicted by the sparsity of Table 5.1.

5.4.5 Results

We retrain the fine-tuned configurations of KPI-BERT and all baselines on the combined training and validation set. To control for a model's susceptibility to random weight initialization, we execute each retraining process 10 times with different seeds. Thereafter, we evaluate all models on the previously specified hold-out test set based on the classification results of the joint NER and RE task. Table 5.5 reports the mean and standard deviation of our metrics based on the 10 seed-varying training runs. We see that KPI-BERT outperforms the other architectures on both the entity and relation classification objective, yielding respective F_1-scores of 81.08 and 70.88 percentage points. Noticeably, SpERT and the linearly NER decoding model show a significantly lower mean classification performance on both tasks, which likely originates from neglecting label dependencies when decoding NER tags. The CRF-based extraction model with conditional label masking (CRF_{LM}) takes label dependencies into account but still achieves lower scores while suffering from a higher standard deviation across differently seeded runs indicating a worse model robustness compared to KPI-BERT.

Table 5.6 showcases several test set sentence examples where KPI-BERT predicts valid relations that either have not been annotated correctly or deviate only slightly from the ground truth entity spans, but still contain valuable information. For instance, in Sentence 3 the annotators did not add the word "Raw" to the entity annotation of "Raw, auxiliary and operating materials". The model predicted that word, and thus the entity span prediction as well as the relation classifications in this sentence were evaluated as mistakes, although the actual model predictions are arguably correct. Further, Sentence 5 shows that the model is able to detect long-distance relations between entities that even have been wrongly annotated.

Taking the above findings into account we see that KPI-BERT handles noise in the annotations adequately and is capable of extracting valuable KPI relations from complex sentence structures.

Table 5.5: Test set evaluation of the joint named entity and relation classification task, reporting mean (standard deviation) Precision-, Recall- and F_1-scores of 10 identical training runs with varying seeds. Our model, KPI-BERT, outperforms the competing state-of-the-art architectures in both entity extraction and relation linking.

in %		Entity			Relation		
Name	Architecture	Precision*	Recall*	F_1^*	Precision	Recall	F_1
–	BERT + Linear + RE [101]	76.81 ±1.00	81.57 ±0.59	79.12 ±0.72	66.95 ±1.51	69.34 ±1.13	68.12 ±1.26
SpERT	BERT + Span + RE [99]	75.67 ±0.63	**83.45** ±0.46	79.37 ±0.47	67.00 ±0.84	69.48 ±0.63	68.22 ±0.61
–	BERT + CRF_{LM} + RE	79.80 ±0.63	82.35 ±0.51	81.05 ±0.51	**70.68** ±0.81	70.62 ±0.93	70.65 ±0.83
KPI-BERT	BERT + GRU_{LM} + RE	**79.87** ±0.55	82.31 ±0.55	**81.08** ±0.53	70.33 ±0.55	**71.43** ±0.60	**70.88** ±0.55

* = micro average, $_{LM}$ = conditional label masking

5.5 Conclusion and Future Work

In this chapter, we introduce KPI-BERT, an automated system that utilizes new methods of Named Entity Recognition (NER) and Relation Extraction (RE) to jointly extract and relate KPIs and their values from real-world German financial reports. Our system leverages a BERT-based architecture that novelly employs an RNN coupled with conditional label masking to sequentially predict KPI tags. In contrast to several other studies, this setup successfully models label dependencies and takes the sequential nature of entity tagging into account. We further integrate a trainable RNN-based pooling layer, which significantly improves upon classic methods like mean and max pooling.

We compare KPI-BERT with three strong relation extraction models, which equally build on BERT embeddings, but differ in their NER capabilities. Our system outperforms all competing setups in both KPI extraction and entity linking performance, especially surpassing state-of-the-art span-based entity decoders such as SpERT [99]. Ultimately, our results illustrate KPI-BERT's capability to correctly learn and identify long term relations, despite the complexity of the prediction task.

In future work, we plan to evaluate KPI-BERT on public datasets, potentially outside the financial accounting domain. Additionally, we intend to investigate cross-attention-based transformer architectures coupled with conditional label masking to sequentially tag entities and classify their relations. Further, current state-of-the-art language models like BERT lack numerical reasoning capabilities and are mainly limited to representing plain text. Since we aim to expand the entity and relation extraction task to structured data, e.g., financial tables, a future direction of research will be to replace BERT with a tailored language model, better capable of numerical reasoning and representing tables.

Table 5.6: Several example sentences from the test set with NER and RE results. Green, blue and red represent "true positive", "false positive", and "false negative" entity and relation classifications, respectively.

	Sentence with predicted Entities	Relations
(a)	Correct relation predictions and annotations.	
1	The [other assets]$_{\text{kpi}}$ include "[earmarked loans]$_{\text{davon}}$" amounting to TEUR [25]$_{\text{davon-cy}}$ (prior year: TEUR [25]$_{\text{davon-py}}$), which were given to franchise partners in fiscal year 2004 to 2007.	kpi – davon davon – davon-cy davon – davon-py
2	The [members of the supervisory board]$_{\text{attr}}$ received T€ [368]$_{\text{cy}}$ (prior year T€ [242]$_{\text{py}}$) as [total compensation]$_{\text{kpi}}$ in fiscal year 2007.	kpi – attr kpi – cy kpi – py
(b)	Minor differences between ground truth annotations and model predictions. Arguably, the model predictions are correct, but an annotation mismatch leads to a sentence F_1-score below 1.	
3	[Raw, [auxiliary and operating materials]$_{\text{kpi kpi}}$] in the amount of [56,9]$_{\text{cy}}$ M. € (prior year [50,3]$_{\text{py}}$ M. €) are mainly allocated to medical requirements.	kpi – cy kpi – py kpi – cy kpi – py
4	In 2011, [the [performance-related share]$_{\text{kpi kpi}}$] is composed of [short term components]$_{\text{davon}}$ in the amount of TEUR [150]$_{\text{davon-cy}}$ (corresponding to 75% of the performance-related share), which will be paid off during the following fiscal year, and [long term components]$_{\text{davon}}$ in the amount of TEUR [48]$_{\text{davon-cy}}$ (corresponding to 25% of the performance-related share).	kpi – davon kpi – davon kpi – davon kpi – davon davon – davon-cy davon – davon-cy
(c)	Wrong ground truth annotations, but arguably correct model predictions.	
5	[These]$_{\text{kpi-coref}}$ include [[revenues from not yet billed service contracts]$_{\text{davon kpi}}$] in the amount of T€ [[6.848]$_{\text{davon-cy cy}}$] (prior year T€ [[3.950]$_{\text{davon-py py}}$]), which are realised in accordance to the stage of completion.	kpi-coref – davon davon – davon-cy davon – davon-py kpi – cy kpi – py
6	The [interest expenses]$_{\text{kpi}}$ include TEUR [[848]$_{\text{davon-cy cy}}$] (prior year TEUR [[816]$_{\text{davon-py py}}$]) for the [compounding of long term accruals]$_{\text{davon}}$, in particular for retirements, partial retirements and anniversaries and TEUR [317]$_{\text{davon-cy}}$ (prior year TEUR [432]$_{\text{davon-cy}}$) [for [remunerations of factoring services]$_{\text{davon davon}}$].	kpi – davon kpi – davon davon-cy – davon davon-py – davon davon-cy – davon davon-py – davon kpi – cy kpi – py kpi – cy kpi – py

CHAPTER 6

Contrastive Learning for Numerical Consistency Checks

Building upon the KPI-BERT model introduced in the previous chapter, which effectively extracts and links Key Performance Indicators (KPIs) from financial documents, we now address the critical task of verifying the numerical consistency of these KPIs within a report. Ensuring that semantically equivalent KPIs maintain consistent numerical values throughout a document is essential for analysts, investors, and auditors who rely on accurate financial information to make informed decisions.

In this chapter, we present *KPI-Check*, an advanced system that automatically identifies and cross-checks semantically equivalent KPIs within real-world German financial documents. By focusing on the detection of numerical inconsistencies, KPI-Check enhances the reliability and integrity of financial reports, contributing to improved transparency and trust in corporate disclosures.

To achieve this, we augment the capabilities of KPI-BERT with additional modules and methodologies:

- **Joint Sentence and Table Encoding Module:** We use a fine-tuned BERT model to jointly encode processed sentences and tables. We extract context-aware embeddings for the KPIs within these texts, utilizing the known positions of the individual KPI words and aggregating them to a single embedding via max-pooling.
- **Contrastive Autoencoder (CAE) Classification Module:** We introduce a binary prediction module that leverages a CAE to classify the previously embedded KPI pairs. By employing weighted over- and under-sampling, we expose the model to related pairs more frequently during training, enhancing its ability to learn semantic similarities despite data imbalance. The CAE minimizes the distance between embeddings of related KPI pairs while maximizing it for unrelated pairs, effectively distinguishing between them.

One of the key challenges in KPI matching is the extreme data imbalance caused by the vast number of unrelated KPI pair combinations. To address this, we introduce a *Filtering Module* that acts as a pre-filtering step prior to the linking process. This module employs a fine-tuned BERT model with cross-attention mechanisms to jointly encode candidate sentence-table pairs.

Based on this representation it is fine-tuned to keep semantically relevant pairs and to discard unrelated ones, effectively simplifying the matching task and improving the system's overall performance.

Recognizing the impracticality of manually annotating thousands of financial documents, we implement an *Automated Annotation Process* leveraging informed number matching. By accurately linking each KPI to its numerical quantities, units, and scales, and accounting for potential rounding errors, we match KPIs based on their numerical values. This process minimizes false positives and negatives, providing reliable training data without manual intervention. Our results show that despite the presence of some false positive and negative annotations, KPI-Check generalizes effectively, learning the correct similarity patterns and ultimately outperforming the annotation process itself.

To validate the effectiveness of KPI-Check, we conduct comprehensive experiments comparing our system with several baselines, including fuzzy string matching, an MLP, and a siamese network architecture trained via contrastive learning. Additionally, we perform ablation studies by removing the filtering module to assess its impact on performance. Our results demonstrate that KPI-Check outperforms these baselines, highlighting the importance of each component, particularly the filtering module and the contrastive learning approach in handling data imbalance and improving classification accuracy.

Our complete system achieves a final test set micro F_1-score of 73.00%, demonstrating its capability to learn semantic similarities and identify numerical inconsistencies effectively.

This chapter is based on the following publication:

- L. Hillebrand, T. Deußer, T. Dilmaghani, B. Kliem, R. Loitz, C. Bauckhage, and R. Sifa, "Towards automating Numerical Consistency Checks in Financial Reports," *Proc. BigData*, 2022, DOI: 10.1109/BigData55660.2022.10020308 [31].

As the primary contributor, Lars Hillebrand conceived and developed the methodology, implemented the codebase, designed and conducted the experiments, processed the data, evaluated the results, and authored the paper. Co-authors assisted in revising the final version of the work.

6.1 Introduction

Corporate disclosure documents like annual financial statements, management reports, or initial public offering (IPO) prospectuses play a vital role in informing the public about a company's economic state of affairs. They contain large amounts of numerical facts that convey detailed information about profitability, financial strength, and operational efficiency. These KPIs greatly affect investment decisions of outside investors and in return impact the company's future development. Hence, their authenticity, factual correctness, and consistency within the report are of immense importance, which is reflected in strict reporting standards, e.g., IFRS (International Financial Reporting Standards), whose compliance is regularly validated by external auditors.

Companies themselves and external auditors spend a considerable amount of time manually cross-checking these numerical facts and financial indicators, which occur in tables and are further explained and referred to in various sections of text across the entire document. Due

to their large volume and tedious report generation process, which often involves multiple authors, frequent updates, and copy-pasting, they are prone to numeric inconsistencies. These errors frequently persist even after official publishing, which negatively affects the reader's impression of the firm and thus, harms its reputation and integrity. Several studies have quantified the negative effect of accounting errors on investment decisions and the resulting economic consequences. For example, [124] and [125] show that individual investors prefer to invest in companies with accessible and transparent disclosures. [126] and [127] go a step further and find that accounting errors are negatively associated with share returns and cause market participants to react less to earnings surprises. Therefore, reducing the amount of numerical errors in disclosure documents, while at the same time speeding up the tedious cross-checking process, is in the best interest of companies as well as auditing firms.

To tackle these objectives, we introduce KPI-Check, a sophisticated system that automatically identifies and cross-checks semantically equivalent KPIs in real-world German financial documents. Figure 6.1 shows an excerpt of such a document, in which textual KPIs and their numeric values are successfully identified across the document and matched with their balance sheet counterparts (equal color). Subsequently, the related pairs can be validated for numerical consistency by simply comparing their monetary values taking the scale (e.g., million) and unit (e.g., €) into account.

The balance sheet and profit & loss statement arguably represent the most important sources of information within a financial report. Together, they can be used to assess the year-to-year consistency, performance, and organizational direction of a company. That is why KPI-Check focuses on matching retrieved textual KPIs to these table types.

To achieve the non-trivial linking of synonymous KPIs, our tool consists of three dependent building blocks.

First, we leverage KPI-BERT [30] (introduced in the previous Chapter 5), a novel NER and RE model tailored to the financial domain, which jointly retrieves KPIs from sentences and relates them to their numeric values. Given the following example sentence,

> "In 2021 the revenues$_{\text{kpi}}$ increased from \$76$_{\text{py}}$ million to \$112$_{\text{cy}}$ million while the total costs$_{\text{kpi}}$ decreased to \$47$_{\text{cy}}$ million (prior year: \$66$_{\text{py}}$ million)."
>
> revenue$_{\text{kpi}}$ − 112$_{\text{cy}}$, revenue$_{\text{kpi}}$ − 76$_{\text{py}}$, total costs$_{\text{kpi}}$ − 47$_{\text{cy}}$, total costs$_{\text{kpi}}$ − 66$_{\text{py}}$

it automatically recognizes and classifies the highlighted financial indicators and links their numeric relations, where *kpi*, *cy* (current year value), and *py* (prior year value) are part of previously defined entity classes.

Second, we integrate a joint sentence- and table encoding module which utilizes BERT [34] to find relevant sentence/table pairs within a financial report. One of the key challenges of linking semantically equivalent KPIs is the extreme data imbalance since the large majority of KPI pair combinations are unrelated. Hence, this filtering module substantially simplifies the matching task, which positively impacts the system's performance.

Lastly, we introduce a binary prediction module using a CAE to classify the remaining KPI pairs employing weighted sampling techniques to expose related pairs more frequently during training.

Balance sheet as of December 31, 2011

Assets

	31.12.2011 EUR	31.12.2010 EUR
A. Fixed assets		
I. Property, plant and equipment		
Office furniture and equipment	1,00	867,64
II. Financial assets		
1. Shares in affiliated companies	73.800.962,67	36.000.962,67
2. Loans to affiliated companies	61.100.000,00	98.900.000,00
	134.900.963,67	134.901.830,31
B. Current assets		
I. Receivables and other assets		
1. Receivables from affiliated companies	24.074.137,40	26.067.468,52
2. Other assets	2.840,00	26.751,35
	24.076.977,40	26.094.219,87
II. Cash on hand, bank balances	645.050,74	3.048.351,92
	24.722.028,14	29.142.571,79
C. Prepaid expenses	73.331,32	66.018,45
	159.696.323,13	164.110.420,55

...

D. Notes to the balance sheet

1. Fixed assets

The development of fixed assets is shown in the appendix to these notes. Shares in affiliated companies increased from EUR 36.0 million to EUR 73.8 million. …
Accordingly, loans to affiliated companies decreased from EUR 98.9 million to EUR 61.1 million.
...

2. Net assets, financial position and results of operations

...
Cash and cash equivalents amount to EUR 0.6 million (2010: EUR 3.0 million). The decrease is mainly due to the dividend payment for the previous year, which amounted to EUR 5.5 million. In the reporting year, 14,000 shares were acquired as part of the share buyback program. The difference of EUR 0.328 million resulting from the acquisition of treasury shares was offset by EUR 0.130 million against other revenue reserves and the remaining amount of EUR 0.198 million against retained earnings. As of the reporting date, the Company held 143,900 no-par value shares in treasury.

Figure 6.1: A screenshot of selected parts of a German financial statement (translated to English via DeepL, `https://www.deepl.com/`) showcasing the successful linking of semantically equivalent Key Performance Indicators (KPIs) that occur in the balance sheet and at different places across the document text. KPIs colored equally refer to the same fact and thus, have to be numerically consistent.

The complete system achieves a final test set micro F_1-score of 73.00%, which shows the model's capability to learn semantic similarities despite the task's difficulty and the aforementioned imbalance challenge.

KPI-Check is currently being deployed for a major auditing firm as a separate component of an AI-based auditing tool for financial statements. It adds a convenient method to automatically retrieve and highlight identical KPIs and detect numerical inconsistencies in financial documents. First user tests have already revealed significant efficiency gains and continuous use in production will further improve the system's performance due to the integration of human feedback, e.g., in the form of error corrections.

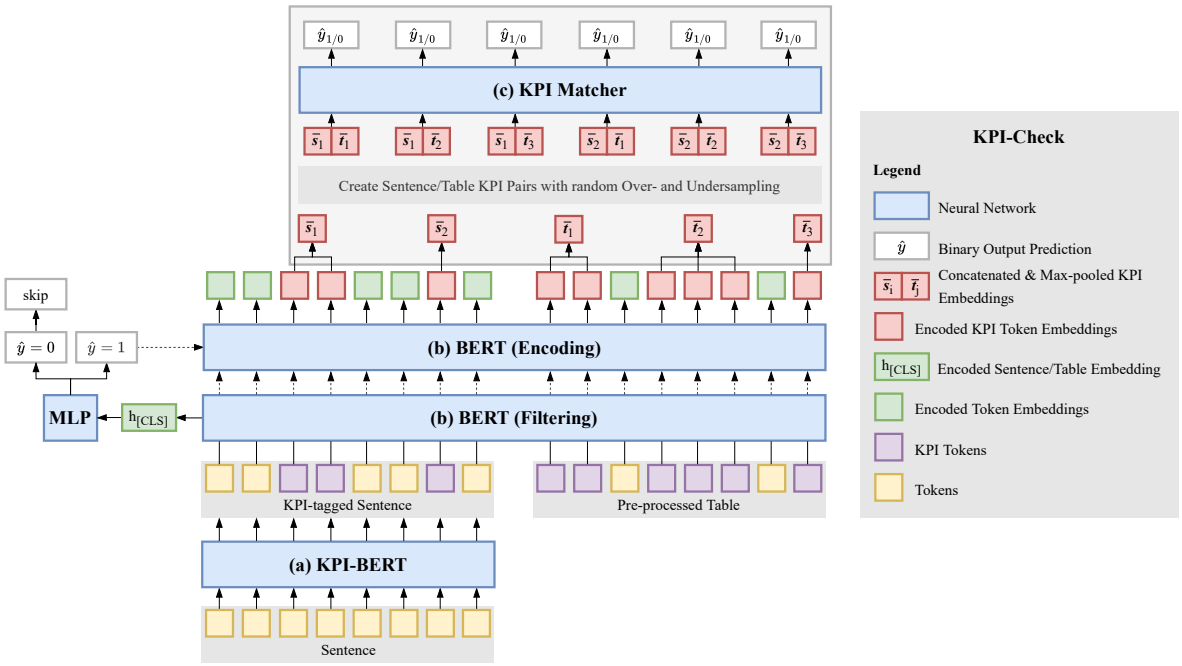


Figure 6.2: Schematic visualization of our system, KPI-Check, to automatically identify and link semantically equivalent Key Performance Indicators (KPIs) in sentences and tables within financial documents. First, (a) sentences are passed through KPI-BERT, a joint named entity and relation extraction model, to retrieve KPIs and their numeric values. Second, (b) a BERT-based filtering model using cross-attention with a multi-layer perceptron (MLP) classification head classifies a KPI-tagged sentence and pre-processed table pair to either match (contain equivalent KPIs) or not. If a match is predicted, a separate BERT-based encoding module utilizing max-pooling creates encoded sentence- and table KPI embeddings, $\bar{\boldsymbol{s}}$ and $\bar{\boldsymbol{t}}$. Finally, (c) a contrastive autoencoder (CAE) classifies each sentence/table KPI pair to be synonymous or not.

6.2 Related Work

In today's digitized world with an ever-growing amount of freely available information claim-checking becomes more and more important to guarantee factual correctness and consistency. Over the past years, it has sparked much research interest across multiple disciplines. For example, [128] introduce StatCheck, a rule-based tool that detects inconsistencies during significance testing in academic psychology papers. [129] develop a fact-checking platform called ClaimBuster that utilizes NLP and supervised learning to identify important factual claims in political discourses. Additionally, [130] and [131] present end-to-end fact-checking systems that predict the factuality of historically given claims.

The majority of these systems rely on pre-trained natural language models [17, 34, 92] and utilize NER [104, 105, 132–134] and RE [99, 110–113, 135], of which KPI-Check is no exception since it makes use of KPI-BERT [30], a joint named entity and relation extraction model tailored to the financial domain (see Chapter 5).

Turning to our concrete task of analyzing and cross-checking financial documents, [136] suggest Jura, an ML-based compliance tool to improve the efficiency of reviewing annual financial reports submitted to the Hong Kong Exchanges and Clearing (HKEX). Similarly, [27] and [28] propose and improve ALI (Automated List Inspector), a recommender-based tool that greatly simplifies and to a large extent automates the auditing of financial statements.

However, the previously named systems lack our focus on the numerical consistency of KPIs. The closest research in this regard is probably the studies by [98] and [137]. The former extracts formulas from verbal descriptions of numerical claims while leaving the actual claim-linking task for future work. Also, they extract financial indicators using a whitelist, which might not be general enough depending on the variety of KPIs. The latter study analyzes how well different KPIs in tables can be cross-checked in Chinese IPO prospectuses and auditing reports. The authors achieve great results in identifying semantically equivalent table cells, but we consider it problematic that they leak numerical information in the matching process, which they claim is purely based on semantics.

Since identifying semantically equivalent KPI pairs in financial reports suffers from an extremely high data imbalance KPI-Check's task qualifies as an outlier/anomaly detection problem. In the past many architectures and training schemes have been proposed to tackle such problems. However, autoencoders [138–142] stand out in popularity, both in supervised and unsupervised settings. KPI-Check, being no exception, leverages a CAE that uses contrastive learning on the reconstruction loss to separate equivalent from unrelated KPI pairs.

6.3 Methodology

In this section, we briefly formulate the problem and motivate our modeling approach before turning to the in-depth analysis of our proposed architecture which is visualized in Figure 6.2.

6.3.1 Problem Formulation and Modeling Approach

Given a corporate financial report containing a list of tabular KPIs, $\mathcal{T}$, depicted in the balance sheet and profit & loss statement, and a list of sentence KPIs, $\mathcal{S}$, occurring across the entire document, we identify all semantically equivalent pairs from the combined Cartesian product $\mathcal{T} \times \mathcal{S}$. Once we succeed in this task, we can automatically cross-check their monetary values and thus, verify numerical consistency.

Effectively, the above-described objective can be divided into three sub-problems.

First, we extract the originally unknown performance indicators, $\mathcal{T}$ and $\mathcal{S}$, along with their numerical quantities from the document. In the case of $\mathcal{T}$ this can be done rule-based, due to the known structure of the balance sheet and profit & loss statement. On the contrary, retrieving $\mathcal{S}$ is more difficult which is why we utilize KPI-BERT [30], a dedicated named entity and relation extraction model that is trained to detect and link KPIs within unstructured sentences (see Figure 6.2 (a)). KPI-Check uses the extracted and linked KPIs subsequently in the following two tasks.

Second, we create vector representations (embeddings) for all extracted KPIs in $\mathcal{T}$ and $\mathcal{S}$ that capture their semantics and contextual information (see Figure 6.2 (b)). It is important to note that we explicitly exclude the KPI's numeric quantity from the embedding process so that our classification module is forced to only learn semantic and contextual similarities instead of

focusing on numerical equivalence. Enabling the latter would open up the possibility of linking KPIs solely based on their monetary values, which contradicts our idea of matching identical KPIs to find numerical inconsistencies.

Third and finally, we classify each KPI pair $(t, s) \in \mathcal{T} \times \mathcal{S}$ to either *match*, $+$, or *not match*, $-$ (see Figure 6.2 (c)). Due to the huge discrepancy in the amount of synonymous (small) and unrelated (large) KPIs, a key challenge is the extremely high classification imbalance of $\frac{|\mathcal{T}\times\mathcal{S}|^-}{|\mathcal{T}\times\mathcal{S}|^+} \approx 500:1$ which effectively qualifies this task as outlier detection problem. To reduce this issue, we introduce a separate filtering module that is trained to remove irrelevant sentence/table pairs not containing any matching KPIs before performing the actual KPI linking step. In addition, we incorporate class-weighted sampling in the training process to expose the minority class of matching KPI pairs more frequently. Lastly, we perform the final KPI linking step by introducing a contrastive autoencoder (CAE) that utilizes contrastive learning to robustly differentiate between inliers (unrelated KPIs) and outliers (synonymous KPIs).

The following sections describe our solutions to these sub-tasks in more detail.

6.3.2 Entity Extraction and Relation Linking (KPI-BERT)

To retrieve and connect all textual KPIs, $\mathcal{S}$, and their numeric quantities from unstructured sentences in a financial report, we leverage a named entity and relation extraction model, called KPI-BERT [30]. The model consists of three stacked components that are trained jointly in an end-to-end fashion via gradient descent. The entire architecture and its training process are described in full detail in the previous Chapter 5.

6.3.3 Entity Encoding

After KPI-BERT [30] successfully extracts all textual KPIs in $\mathcal{S}$ and links their numeric quantities within sentences, we turn to the second sub-task of encoding sentence- and table KPIs into vector space while preserving their semantics and context.

Pre-Processing

For the balance sheet and profit & loss statement, we only regard their first column, which comprises the entirety of tabular KPIs, $\mathcal{T}$. The current and prior year numeric values, present in the remaining table columns, are deliberately discarded in the embedding process. Including them would potentially enable the downstream classification model to base its predictions mainly on the numerical equivalence of two KPIs. This would be problematic since semantically identical KPIs with inconsistent numerical values, e.g., caused by human error, might not be detected.

We flatten and pre-preprocess each table, which is exemplarily depicted in Figure 6.3. First, we use rule-based heuristics and regular expressions to remove any hierarchical prefix, i.e. converting "1. Gross profit" to "Gross profit". Second, we include two special tokens, <row> and <nan> to separate new rows and tag empty KPIs, respectively.

To prepare the previously extracted sentence KPIs for encoding, we process each sentence by enclosing all tagged KPIs with HTML-like special tokens, i.e.

in $	2019/2020	2018/2019
1. Gross profit	24 059 512.21	22 051 698.38
2. Personnel expenses		
a) Wages and salaries	15 675 943.67	13 231 237.73
b) Social security contributions	1 375 421.49	1 865 432.63
...	...	...

(a) Excerpt of a profit & loss statement.

Gross profit <row> <nan> <row> Personnel expenses <row> Wages and salaries <row> Social security contributions <row> ...

(b) Flattened and pre-processed profit & loss statement.

Figure 6.3: Example of a profit & loss statement (a) and its pre-processed and flattened Key Performance Indicators (KPI) sequence (b).

In 2022, the <kpi> revenue </kpi> increased to $1.5 million.

Filtering

As visualized in Figure 6.2 (b), a filtering module decides prior to linking KPIs whether a candidate sentence/table pair contains synonymous KPIs. Concretely, we employ a pre-trained and in the process fine-tuned BERT model which jointly encodes the processed sentence and table using cross-attention to learn a combined sentence/table representation

$$\boldsymbol{h}_{[\mathrm{CLS}]} = \mathrm{BERT}_{\mathrm{filter}}\left([\mathrm{CLS}]\ \text{sentence}\ [\mathrm{SEP}]\ \text{table}\ [\mathrm{SEP}]\right), \tag{6.1}$$

where [CLS] and [SEP] denote BERT-specific special tokens used for input classification and separation. Subsequently, we classify the pair's relevance by passing $\boldsymbol{h}_{[\mathrm{CLS}]}$ to a simple MLP consisting of a fully connected layer followed by dropout [123] and a sigmoidal activation function:

$$\hat{y}_{\mathrm{filter}} = \mathrm{MLP}_{\mathrm{filter}}\left(\boldsymbol{h}_{[\mathrm{CLS}]}\right). \tag{6.2}$$

If $\hat{y}_{\mathrm{filter}} \in [0, 1]$ is below a pre-defined confidence threshold α_1, we discard the sentence/table pair to reduce the aforementioned data imbalance problem and thus, increase the final KPI matching performance.

Encoding

We use a dedicated pre-trained BERT model to separately encode pre-processed sentences and tables that succeeded the previous filtering process. Specifically, given a KPI-tagged sentence

of n tokens and a flattened table of m tokens, we obtain context-aware sub-word embeddings:

$$\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n = \text{BERT}_{\text{encode}}\left([\text{CLS}]\ \text{sentence}\ [\text{SEP}]\right), \tag{6.3}$$

$$\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_m = \text{BERT}_{\text{encode}}\left([\text{CLS}]\ \text{table}\ [\text{SEP}]\right). \tag{6.4}$$

Next, we utilize the known positions of our KPIs within the embedding sequences and apply max-pooling to create KPI embeddings for both, tables and sentences. Similar to Equation 5.7, an arbitrary KPI in a sentence containing k sub-words is represented as

$$\bar{\boldsymbol{s}} = \left[\text{max-pool}(\boldsymbol{s}_i, \boldsymbol{s}_{i+1}, \ldots, \boldsymbol{s}_{i+k-1}); \boldsymbol{w}_k^{\text{width}}\right], \tag{6.5}$$

where $\boldsymbol{w}_k^{\text{width}}$ again denotes a unique size embedding. $\bar{\boldsymbol{t}}$ follows the same approach for tables.

6.3.4 Entity Pair Classification

We cast the entity pair classification task as an outlier detection problem to further tackle the imbalance of relatively few synonymous KPI pairs compared to many unrelated KPI pairs. Specifically, we implement the KPI matching network depicted in Figure 6.2 (c) as a CAE leveraging contrastive learning on the reconstruction loss to distinguish synonymous from unrelated pairs. The autoencoder is defined as

$$\varphi(\boldsymbol{x}) \coloneqq \text{dec}(\text{enc}(\boldsymbol{x})), \qquad \boldsymbol{x} = \left[\bar{\boldsymbol{s}}; \bar{\boldsymbol{t}}\right], \tag{6.6}$$

where $\boldsymbol{x}$ represents the concatenation of the sentence- and table KPI embeddings $\bar{\boldsymbol{s}}$ and $\bar{\boldsymbol{t}}$ that are randomly sampled from the pool of pairs $\mathcal{T} \times \mathcal{S}$. The encoder (enc) and decoder (dec) networks are MLPs with two fully connected layers each enclosed by ReLU [143] activation functions and dropout [123]. Following the standard design of autoencoders the hidden dimension imposes an information bottleneck ($\dim_{\text{hidden}} \ll \dim_{\text{input}} = \dim_{\text{output}}$), which enforces meaningful representation learning for correct input reconstruction.

Given the reconstructed input, $\hat{\boldsymbol{x}} = \varphi(\boldsymbol{x})$, the original input $\boldsymbol{x}$ and the ground truth label $y \in \{0, 1\}$ we train the CAE to minimize the reconstruction loss for unrelated KPI pairs, $y = 0$, while maximizing it for semantically equivalent pairs, $y = 1$. Concretely, we define the combined contrastive loss as

$$\mathcal{L}_{\text{contrastive}}(\hat{\boldsymbol{x}}, \boldsymbol{x}, y) = (1 - y) \cdot \mathcal{L}_{\text{MSE}}(\hat{\boldsymbol{x}}, \boldsymbol{x}) + y \cdot \max(0, m - \mathcal{L}_{\text{MSE}}(\hat{\boldsymbol{x}}, \boldsymbol{x})), \tag{6.7}$$

where $\mathcal{L}_{\text{MSE}}$ denotes the mean-squared error loss function and m denotes the margin parameter enforcing that the model focuses particularly on difficult-to-reconstruct samples during optimization.

During inference, we normalize the resulting mean-squared error loss with a sigmoid layer such that

$$\hat{y} = \text{sigmoid}(\mathcal{L}_{\text{MSE}}(\hat{\boldsymbol{x}}, \boldsymbol{x}))) \in [0, 1]. \tag{6.8}$$

If $\hat{y}$ is above the threshold α_2, we consider the KPI pair (t, s) to be semantically equivalent.

6.3.5 Training

We decouple the training process of KPI-Check and train each sub-module independently due to the significantly different objectives in their sub-tasks and a more effective usage of tailored random over and undersampling (ROUS) per task.

First, our named entity and relation extraction model for sentences, KPI-BERT, trains on a smaller subset of 500 manually annotated financial reports[1], to extract and link KPIs and their numeric quantities from each sentence. The model jointly optimizes the named entity (categorical cross-entropy) and relation extraction (binary cross-entropy) loss. Further details about the training and hyperparameter tuning process can be again taken from [30]. In this work we leverage the best-performing version of KPI-BERT as reported in [30].

Second, we train the components of the filtering module, $\text{BERT}_{\text{filter}}$ and $\text{MLP}_{\text{filter}}$, end-to-end while fine-tuning $\text{BERT}_{\text{filter}}$ explicitly on the task of identifying relevant sentence/table pairs. We optimize the binary cross-entropy loss over related and unrelated pairs. Since most sentences and tables are unrelated and only a few share semantically equivalent KPIs, we employ weighted random over- and undersampling (ROUS) with replacement to show relevant pairs more often during training. Specifically, the originally uniform sampling probability of each sentence/table pair is altered to the normalized inverse frequency of the pair's class occurrence in the training set. For example, a training set of four pairs $(+, -, -, -)$ receives weighted sampling probabilities of $\left(\frac{1}{2}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}\right)$.

Third, we train the actual KPI-matching model, the CAE, which takes the encoded ($\text{BERT}_{\text{encode}}$) and pooled KPI representations as input, provided their sentence/table pair overcame the previous filtering process. We employ contrastive learning (see Equation (6.7)) to learn a robust decision boundary, which effectively distinguishes between unrelated and synonymous KPI pairs. In addition, we leverage teacher forcing (training only) by also utilizing positively annotated KPI pairs that were falsely filtered out.

6.4 Experiments

In the following sections, we introduce our custom dataset of German financial reports, describe the automated annotation process of synonymous KPIs, discuss the overall training setup including hyperparameter optimization, and evaluate results.

6.4.1 Data

Our dataset[2] comprises 7548 real-world corporate annual reports sourced from the *Bundesanzeiger*[3], a platform hosted by the DuMont media group where German companies publish their legally required documents. A subset of 500 reports (part of the training set) was utilized in [30] to train KPI-BERT, our joint named entity and relation extraction model that identifies and links KPIs in sentences. For the remainder of this paper, we take a trained KPI-BERT

[1] The reports are part of the training split for the other KPI-Check modules.

[2] We are currently unable to publish the dataset and the accompanying `Python` code because both are developed and used in the context of an ongoing industrial project.

[3] `https://www.bundesanzeiger.de/`

Table 6.1: Dataset statistics about Key Performance Indicators (KPI) and sentence/table pairs, highlighting their respective class imbalances of positive (+) and negative (−) pairs.

	Training	Validation	Testing
Documents	6032	764	752
KPI Pair Statistics			
Total			
Pairs +	48 736	5893	5431
Pairs −	17 871 922	2 192 954	2 145 354
Imbalance	367:1	372:1	395:1
Average per Document			
Pairs +	8	8	7
Pairs −	2963	2870	2853
Imbalance	510:1	492:1	560:1
Sentence/Table Pair Statistics			
Total			
Pairs +	42 912	5218	4812
Pairs −	503 150	64 423	60 543
Imbalance	12:1	12:1	13:1
Average per Document			
Pairs +	7	7	6
Pairs −	83	84	81
Imbalance	17:1	16:1	17:1

model as a given and only consider it in the context of KPI-Check. For detailed information about annotations, model training, and performance results we again refer to [30].

We pre-process each report by first tokenizing on a sentence level and subsequently on a word level using the `syntok` `Python` library. Second, we tag monetary numbers and extract their scale (e.g., million) and unit (e.g., $) using regular expressions. Similarly, we utilize rule-based string matching heuristics and trigger words to identify the balance sheet and profit & loss statement within each document. Third, we ignore all other tables and discard sentences not containing any numeric quantities because our only interest lies in linking KPI entities to check their numerical consistency.

For model training, tuning, and evaluation purposes we randomly divide the dataset into a fixed split of 6032 training, 764 validation, and 752 testing documents. Table 6.1 presents detailed KPI- and sentence/table pair statistics for all splits. Each document contains on average around 8 semantically equivalent and 2900 unrelated KPI pairs leading to a per-document imbalance of around $500 : 1$. The overall imbalance across the entire training set still amounts to 367 negative pairs for each positive one, which emphasizes the need for a filtering approach. The corresponding model, $\text{BERT}_{\text{filter}}$, requires sentence/table inputs, whose class ratio of negative (−) to positive (+) pairs is significantly lower equaling $12 : 1$.

6.4.2 Automated Annotation Process

Manually cross-checking over 7000 financial documents to get ground truth annotations for semantically equivalent KPI pairs would be extremely time-consuming and is practically infeasible in the scope of our project. Hence, we introduce an automated annotation logic which leverages informed number matching to efficiently annotate synonymous KPIs while keeping the amount of false positive and false negative annotations to a minimum.

As described in Section 6.3.2 we identify and link each KPI within a sentence to its numeric quantities, i.e. current year (cy) and prior year (py) values. In addition, the value's unit (e.g., €) and scale (e.g., million) are accurately extracted employing regular expressions and rule-based heuristics. The same is true for KPIs occurring in the balance sheet and profit & loss statement. They can be automatically aligned with their quantities due to the known tabular structure.

Utilizing the accurate linking of each KPI to its numerical quantity, we apply informed number matching taking a number's unit, scale, and potential rounding into account. For example, our matching heuristic recognizes that the tabular value of "EUR 73 800 962.67" and the textual value of "EUR 73.8 million" match (see Figure 6.1).

Of course, relying on number matching for linking semantically equivalent KPIs is not perfect. In particular, two sources of noise might dilute the annotation quality. First, all equivalent KPIs whose numeric quantities do not match, i.e. caused by human error due to wrong rounding or copy-pasting, remain undetected (false negative annotations). Second, two numbers can match by chance, although their KPIs are completely unrelated (false positive annotations).

The first issue cannot be avoided but occurs presumably quite rarely. The second issue can be mitigated by applying the following rules.

If two candidate KPIs are respectively linked to current year and prior year values (cy and py), we numerically compare both quantities. In case of two matches, we confidently label the KPI pair positively. If one of the candidate KPIs is linked to only a single numeric quantity (either cy or py), we can only focus on comparing this quantity. In case of a numerical match, we additionally confirm whether the matched number x is "common" by validating if the decade logarithm of the absolute value of x returns a natural number, i.e. $\log_{10}(|x|) \in \mathbb{N}$. If the previous expression evaluates to `False` (not common) we directly label the KPI pair positively. If it evaluates to `True` we add an extra safety cushion by applying fuzzy string matching on the raw strings of both candidate KPIs. Concretely, we use the Levenshtein distance [144] based weighted ratio function `WRatio()` of the `rapidfuzz Python` library to check whether the character similarity of both KPIs is below or above a strict threshold of 0.9 (similarity is bound between 0 and 1). Only if the textual similarity is ≥ 0.9 we label the candidate KPI pair positively. In all other cases, pairs are labeled negatively.

We extend this logic to the sentence/table level, by annotating a corresponding pair positively if at least two KPIs within the pair match.

A qualitative analysis of randomly sampled annotations supports our hypothesis of high labeling accuracy. Out of 100 positively labeled samples, 97 were actually correct. Also, a small amount of annotation noise is acceptable due to the large amount of data and the proven robustness of deep neural networks [145]. In addition, employing the available numerical information for automated labeling is fully decoupled from the actual modeling process. As described in Section 6.3.3, KPI-Check discards all numerical information and solely uses

semantics and context to learn useful representations for synonymous KPI matching.

In Section 6.4.6 we see that KPI-Check generalizes well and is indeed able to predict correct KPI matches that were missed during the automated annotation process.

6.4.3 Evaluation Metrics

We quantitatively evaluate our system's performance by calculating precision, recall, and F_1 scores. For a single document d and given sets of predicted- and ground truth KPI pairs, $\hat{\mathcal{Y}}$ and $\mathcal{Y}^*$, the three metrics are defined as

$$\text{Precision} = \frac{|\hat{\mathcal{Y}} \cap \mathcal{Y}^*|}{|\hat{\mathcal{Y}}|}, \quad \text{Recall} = \frac{|\hat{\mathcal{Y}} \cap \mathcal{Y}^*|}{|\mathcal{Y}^*|}, \quad \text{F}_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{6.9}$$

For N documents $(d_1, d_2, \ldots, d_N)$ we calculate macro scores by averaging the document-level metrics and micro scores by aggregating $\hat{\mathcal{Y}}$ and $\mathcal{Y}^*$ across all documents. For example, macro and micro recall are respectively defined as

$$\text{Recall}_{\text{macro}} = \frac{\sum_{i=1}^{N} \text{Recall}_i}{N}, \quad \text{Recall}_{\text{micro}} = \frac{\sum_{i=1}^{N} |\hat{\mathcal{Y}}_i \cap \mathcal{Y}_i^*|}{\sum_{i=1}^{N} |\mathcal{Y}_i^*|}, \tag{6.10}$$

where the subscript i refers to the i-th document. Macro and micro metrics for precision and F_1 score are calculated analogously.

6.4.4 Training Setup

In this section, we shed light on the training setup and hyperparameter optimization of the individual components of KPI-Check (except KPI-BERT [30], which is thoroughly described in Chapter 5).

We determine the best hyperparameter setup for each sub-module by conducting an extensive grid search analyzing various parameter combinations based on their validation set micro F_1-score. Table 6.2 shows all tuned model parameters with their respective ranges of values. The best-performing parameter setup on the validation set is highlighted in boldface.

Each sub-module employs the cased $\text{BERT}_{\text{BASE}}$ encoder, published by the MDZ Digital Library team (dbmdz)[4], which has the same architectural setup as the English $\text{BERT}_{\text{BASE}}$ counterpart[5] and is pre-trained on a large corpus of German news reports, books and Wikipedia articles. We initialize all of KPI-Check's trainable parameters randomly from a normal distribution $\mathcal{N}(0, 0.02)$ and fix the same random seed of 42 for all training runs. In addition, we utilize the AdamW [122] optimizer with a linear warmup of 10% and a linearly decaying learning rate schedule. Further, we apply weight decay of 0.01, clip gradients by normalizing their length to 1 and set the width embedding dimension of $\boldsymbol{w}^{\text{width}}$ to 25. In line with Table 6.2 we also evaluate different levels of dropout regularization [123], various peak learning rates, batch sizes, and hidden dimensions.

Our experiments are conducted on four Nvidia Tesla V100 GPUs and the model plus training code is implemented in `PyTorch` [94]. We train the BERT-based filtering model for 10 epochs

[4] `https://huggingface.co/dbmdz/bert-base-german-cased`.

[5] 12 multi-head attention layers with 12 attention heads per layer and 768-dimensional output embeddings.

Table 6.2: Evaluated hyperparameter configurations of KPI-Check's sub-modules, the filtering component, and the contrastive autoencoder (CAE) classification head. The best configuration on the validation set is highlighted in boldface. The classification thresholds α_1 and α_2 are tuned in the $[0, 1]$ interval based on the best validation set micro F_1-score performance. For details on KPI-BERT, we refer to the previous Chapter 5.

Sub-Module	Hyperparameter	Configurations
$\text{BERT}_{\text{filter}}$ + $\text{MLP}_{\text{filter}}$	Batch size	2, **4**, 8
	Learning rate	1×10^{-6}, $\mathbf{1 \times 10^{-5}}$, 1×10^{-4}
	Dropout	0.0, **0.1**, 0.2, 0.3
	Confidence threshold (α_1)	$[0, \mathbf{0.0035}, 1]$
	MLP hidden dimensions	**no**, $(1024, 128)$, $(2048, 256)$, $(2048, 1024, 256)$
CAE	Batch size	32, **64**, 128
	Learning rate	1×10^{-6}, $\mathbf{1 \times 10^{-5}}$, 1×10^{-4}
	Dropout	0.0, **0.1**, 0.2, 0.3
	Confidence threshold (α_2)	$[0, \mathbf{0.7099}, 1]$
	Hidden dimensions (enc/dec)	$(1024, 128)$, $\mathbf{(2048, 256)}$, $(2048, 256, 64)$
	margin (m)	0.5, **1.0**

and find the best-performing model on the validation set after epoch 9 using early stopping. The total training time amounted to 52 hours and 14 minutes. The final KPI matching classification network was trained for 15 epochs until convergence with a training time of 7 hours and 3 minutes.

6.4.5 Baseline and Ablations

We compare the fine-tuned setup of KPI-Check with three baselines and an additional variation discarding the filtering module. Each baseline makes use of KPI-BERT to first extract and link relevant KPIs and their numerical values from sentences. Also, all competing methods are fine-tuned individually on the validation set and only their respective best setup is evaluated on the hold-out test set.

First, we establish a simple baseline, which we denote Fuzzy String Matching, that drops all learnable components and solely employs fuzzy string matching to link semantically equivalent KPIs. Concretely, it utilizes the extracted KPI strings from sentences (KPI-BERT) and tables and links them by applying the weighted string similarity function `WRatio()` from the `rapidfuzz` `Python` library which is based on the Levenshtein distance [144]. If the similarity exceeds a fixed threshold which is tuned on the validation set, we classify two candidate KPIs as semantically equivalent.

Second, we benchmark KPI-Check and its CAE classification layer depicted in Figure 6.2c against an MLP with two fully connected layers. It is enclosed by ReLU activation functions [143] and dropout [123] and is followed by a sigmoidal output layer. Formally, the MLP is

Table 6.3: Test set results of the sentence/table pair filtering sub-task (a) and the final task of matching semantically KPIs (b). Our full model, KPI-Check, achieves the highest micro- and macro F_1 scores of 73.00% and 70.52%, which significantly improves upon the variation with no filtering module and outperforms the other baselines which all employ the filtering module. We also report upper-bound metrics for our approach, assuming a perfect filtering module with no mistakes.

in %		Micro			Macro		
Task	Architecture	Precision	Recall	F_1	Precision	Recall	F_1
(a) Filtering	$\text{BERT}_{\text{filter}} + \text{MLP}_{\text{filter}}$	70.72	80.01	75.08	72.75	81.73	70.31
(b) KPI Matching	Fuzzy String Matching	49.18	40.88	44.65	56.91	48.62	41.37
	Siamese Network	52.94	67.43	59.31	64.15	72.19	60.98
	MLP	71.83	73.71	72.76	76.09	76.76	70.21
	$\text{KPI-Check}_{\text{no filtering}}$	62.13	**77.55**	68.99	66.42	**80.34**	65.71
	KPI-Check	**73.16**	72.84	**73.00**	**77.21**	76.21	**70.52**
	$\text{KPI-Check}_{\text{perfect filtering}}$	90.81	84.09	87.32	91.79	83.65	85.57

defined as

$$\hat{y} = \text{MLP}\left(\left[\bar{\boldsymbol{s}}; \bar{\boldsymbol{t}}\right]\right). \tag{6.11}$$

Third, we compare KPI-Check's classification performance with a siamese network, first introduced by [146], that is also trained via contrastive learning but utilizes the siamese architecture of two identical MLPs with shared weights. In particular, the sentence- and table KPI embeddings $\bar{\boldsymbol{s}}$ and $\bar{\boldsymbol{t}}$ are passed individually through the same MLP, and the similarity of their resulting representations is calculated based on the cosine similarity. Similar to the CAE we employ a contrastive loss to minimize the cosine similarity for unrelated KPI pairs while maximizing it for semantically equivalent pairs. During inference, the classification decision is thus based on the cosine similarity score and a fixed threshold which is again tuned on the validation set.

Note both the MLP and the Siamese Network make use of the previous filtering module to ensure a fair comparison to KPI-Check.

Last, we train a fine-tuned ablation version of our complete system that discards the filtering module and directly encodes and classifies all KPI pairs. We call this variation $\text{KPI-Check}_{\text{no filtering}}$.

6.4.6 Results

We evaluate and compare KPI-Check, its variations, and all baselines on the previously specified hold-out test set. Table 6.3 reports micro- and macro precision, recall, and F_1 scores for (a) the filtering sub-task and (b) the final task of matching semantically equivalent KPIs.

First, it can be seen that classifying learned KPI representations significantly outperforms the purely string-based approach of fuzzy string matching by almost more than 25 percentage

Table 6.4: Impact of sentence/table pair filtering on the KPI pair imbalance. Our actual filtering model drastically reduces the number of negative KPI pairs while keeping the majority of positive once to reduce the overall test set imbalance from 395 : 1 to 46 : 1. "−" denotes no filtering (see Table 6.1) and "perfect" assumes a perfect filtering model that makes no mistakes to establish an upper bound.

KPI Statistics	Filtering	Training	Validation	Testing
Pairs +	−	48 736	5893	5431
	model	48 731	4659	4408
	perfect	48 736	5893	5431
Pairs −	−	17 871 922	2 192 954	2 145 354
	model	1 688 112	199 468	200 839
	perfect	1 589 333	189 152	180 401
Imbalance	−	367:1	372:1	395:1
	model	35:1	43:1	46:1
	perfect	32:1	32:1	33:1

points on all metrics.

Second, we find that including a carefully tuned filtering module improves KPI-Check's overall micro and macro F_1 score performance by 4 and 5 percentage points to 73.00% and 70.52%, respectively. The same filtering module is also utilized by the baselines, MLP and Siamese Network. However, KPI-Check with its contrastive autoencoder classification head outperforms both competing approaches.

Since the filtering of irrelevant sentence/table pairs is applied before the actual KPI pair classification, any errors are inevitably carried over to the KPI matching module. Hence, we analyze different filtering thresholds α_1 to find the right trade-off between reducing the dataset imbalance on the one hand and falsely removing correct KPI pairs on the other hand. The best-performing model is included in KPI-Check and the other baselines, and it is separately evaluated on the filtering sub-task (see Table 6.3 (a)). Further, we report KPI-Check's hypothetical performance, assuming a perfect filtering model that makes no mistakes. These numbers can be interpreted as an upper bound for the current approach.

Table 6.4 shows the impact of our filtering module on the dataset imbalance of positive and negative KPI pairs. While the number of negative test set pairs decreases considerably from around 2.1 million to 0.2 million (91%), the number of positive pairs remains relatively high with 4408 compared to originally 5431 (19% decrease). Of course, the difference of 1023 wrongly filtered out pairs is in the process automatically classified negatively, thereby reducing the maximum possible micro recall to 81, 16%.

In addition to the quantitative evaluation, Table 6.5 qualitatively highlights a few test set examples where KPI-Check correctly predicts the equivalence of two KPIs, whereas the automated annotations generated via number matching (see Section 6.4.2) are wrong. Examples 1 to 3 reveal rounding and unit inconsistencies in the current and prior year values which lead to wrong negative annotations. Nevertheless, KPI-Check correctly predicted these samples as semantically equivalent. On the contrary, examples 4 and 5 showcase accidental current and

Table 6.5: Translated test set samples of wrongly annotated but correctly predicted Key Performance Indicators (KPI) pairs. Examples 1 to 3 indicate rounding and unit errors in the financial report leading to wrong negative annotations via automated number matching. Examples 4 and 5 reveal accidentally matched but actually unrelated KPIs.

	KPI-Pair (Text/Table)	Current Year		Prior Year		Prediction	Label
1	Active difference	991	T€			+	−
	E. Active difference from asset offsetting	992	T€	863	T€		
2	Net income	293 475.51	€	1 078 302.11	T€	+	−
	15. Net income	293 475.51	€	1 078 302.11	€		
3	Passive surplus of deferred taxes	170	T€	226	T€	+	−
	G. Passive deferred taxes	170 500	€	226	T€		
4	Supervisory board remuneration	16	T€	16	T€	−	+
	Other provisions	15 710	€	15 710	€		
5	Reserve for own shares	1499	T€			−	+
	21. Withdrawals from other revenue reserves	1499	T€	390	T€		

T = thousand

prior year value matching that results in wrong positive annotations. However, the model again correctly classified both samples as completely unrelated.

In summary, these examples demonstrate the model's generalization capability despite automatically generated imperfect annotations.

6.5 Conclusion and Future Work

Numerical inconsistencies of KPIs within published financial reports may diminish investors' trust in a firm's compliance and governance process which potentially impacts the firm economically and in the worst case harms its reputation.

In this work, we approach this issue by introducing KPI-Check, a novel system that aids auditors in automatically identifying semantically equivalent KPIs and validating their numerical consistency. The architecture combines a tailored financial named entity and relation extraction module with a BERT-based filtering component and a contrastive autoencoder (CAE) based text pair classification head. It first extracts KPIs and their numerical facts from unstructured sentences before linking them to synonymous mentions in the balance sheet and profit & loss statement. It achieves a strong matching performance of 73.00% micro F_1-score on a hold-out test set, which shows the model's capability to detect semantically equivalent KPIs with high confidence.

KPI-Check is momentarily being integrated in the auditing process of a major auditing company and first user tests have already promised significant efficiency gains.

In future work, we plan to extend the system to arbitrary table types to guarantee automated KPI cross-checking across the entire financial document. While the balance sheet and profit & loss statement arguably represent the most important pieces of information, many less relevant KPIs are reported in smaller tables throughout the document. Semantically linking these to

other table- and sentence occurrences is high on our agenda.

Further, we set out together with our industry partner to manually annotate a small number of financial statements with respect to creating high-quality ground truth labels of semantically equivalent KPIs. Currently, our model training and quantitative evaluation build on a carefully designed automated annotation process leveraging informed number matching. Despite the high annotation quality and the model's success, we hope to improve the quantitative evaluation even further by utilizing manually crafted labels.

Part III

Large Language Models for Financial Document Compliance

CHAPTER 7

Semantic Text Classification for Sustainability Reports

Building upon the previous chapters where we introduced KPI-Check, a system for extracting and linking financial Key Performance Indicators (KPIs) to assess numerical consistency in financial reports, we now shift our focus towards assisting auditors in ensuring compliance with auditing guidelines and standards. While numerical consistency is essential, adherence to relevant standards such as IFRS (International Financial Reporting Standards) and HGB (Handelsgesetzbuch) for annual financial reporting, and CSRD (Corporate Sustainability Reporting Directive) and GRI (Global Reporting Initiative) for sustainability reports, is paramount.

In this and the following chapter, we develop solutions that enable auditors to quickly retrieve the most relevant text passages of a financial report corresponding to specific regulatory requirements. Subsequently, we advance towards assessing the compliance of these retrieved passages with respect to the disclosure requirements by leveraging LLMs, which will be discussed in Chapter 9.

Now, we start out by introducing sustain.AI, a context-aware recommender system designed to facilitate the retrieval of the most pertinent document sections related to specified disclosure items in sustainability reports, particularly under the GRI standard. Manually locating relevant information in extensive sustainability reports is a daunting and time-consuming task for auditors and stakeholders. Our system addresses this challenge by efficiently identifying and highlighting the most relevant text passages for a given regulatory requirement.

We utilize a BERT-based [34] encoding module paired with a non-linear multi-label classification head, trained together in an end-to-end manner. To address class imbalance in the data, we apply weighted random sampling. Evaluating on two new German sustainability reporting datasets, sustain.AI surpasses numerous strong baselines, achieving over a 10 percentage point improvement in mean average precision.

By providing auditors with an efficient tool to match concrete regulatory requirements to the corresponding text passages in sustainability reports, sustain.AI significantly streamlines the auditing process and enhances compliance verification.

This chapter is based on the following publication:

- L. Hillebrand, M. Pielka, D. Leonhard, T. Deußer, T. Dilmaghani, B. Kliem, R. Loitz, M. Morad, C. Temath, T. Bell, R. Stenzel, and R. Sifa, “sustain.AI: a Recommender System

to analyze Sustainability Reports," *Proc. ICAIL*, 2023, DOI: 10.1145/3594536.3595131 [29].

Lars Hillebrand, as first author, originated and executed the concept for this research, conducting the implementation, preprocessing, experimental design, and model training. He carried out the experiments, processed and analyzed the data, assessed the outcomes, and wrote the paper. The co-authors offered valuable feedback and participated in discussions throughout the writing process.

7.1 Introduction

In the face of climate change and environmental degradation, our society's expectations of sustainable and responsible entrepreneurial action have increased continuously over the past years. Legislators worldwide and particularly in the EU become increasingly aware of the situation and have taken concrete political measures to enforce Corporate Social Responsibility (CSR). In 2014, the EU approved the Non-Financial Reporting Directive (NFRD) which forces large companies to extend their reporting on policies, risks, and KPIs regarding sustainability and social matters. Beginning in 2024, the NFRD will be updated with the stricter CSR-Directive, which applies to around 50 000 European companies and includes a wider catalog of reporting requirements covering environmental, social, and governance aspects. The majority of these requirements are based on the popular regulatory framework from the GRI. Its universal reporting standards provide a detailed set of indicators that address a company's impact on the economy, environment, and people.

In light of these more comprehensive and rigorous sustainability regulations and the public's growing interest in corporate social responsibility, it is of vital importance to make the disclosed information easily accessible and comparable. However, manually retrieving and analyzing the published reports concerning specific GRI-Indicators is practically infeasible, especially considering that the documents often span around a hundred pages or more. This is particularly true for the auditing domain, where auditors spend hours to ensure a report's compliance related to said CSR standards.

Hence, we introduce sustain.AI, a sophisticated, context-aware recommender system that utilizes modern techniques of NLP and ML to process and analyze uploaded sustainability reports. Concretely, interested users like consumers or investors can query the recommender engine for specific GRI-indicators, e.g., the company's emissions (see Figure 7.1), and the engine returns and renders the most relevant document segments related to the query. Thus, stakeholders are able to quickly assess investment risks and opportunities arising from social and environmental issues and to evaluate the sustainability performance of companies. Similarly, auditors significantly benefit from the automated matching of concrete regulatory requirements to the relevant text passages. In fact, a large part of the sustainability report audit is about ensuring the completeness and correctness of the report according to the specified GRI standards.

Our recommender system builds on a BERT-based [34] encoding module followed by a non-linear multi-label classification head. Both components are trained jointly in an end-to-end fashion, leveraging weighted random sampling (WRS) to counter the significant class label imbalance. We evaluate the model on two novel German sustainability reporting data sets

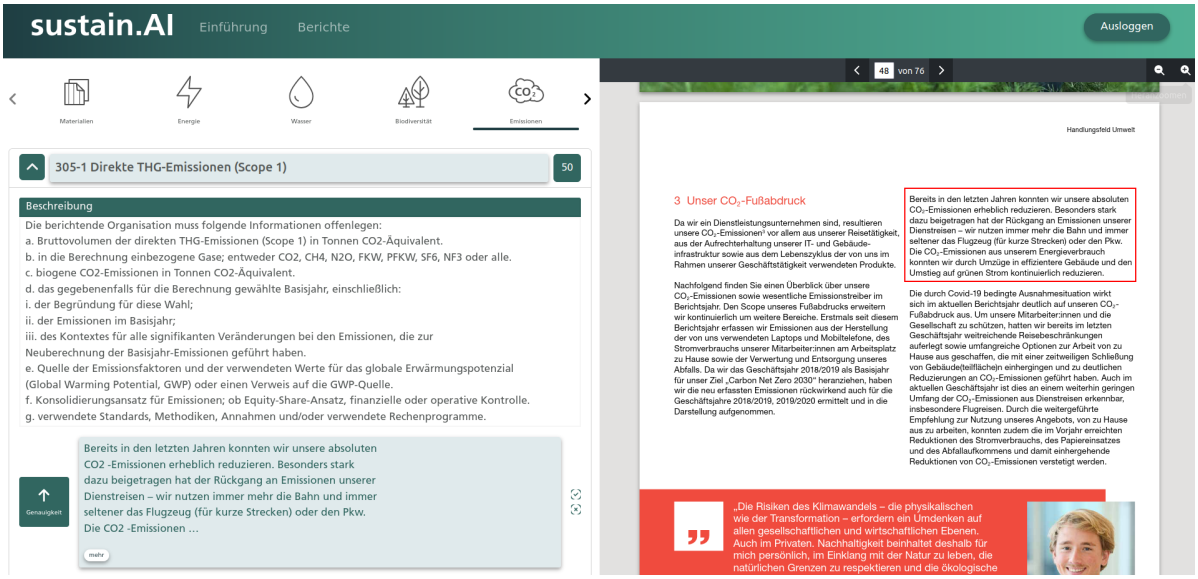


Figure 7.1: A screenshot of the sustain.AI recommender tool. After selecting a specific regulatory requirement from one of the categories, the system predicts the most relevant segments of a provided sustainability report. On the right side, the recommended segments are highlighted in the rendered report, fostering an efficient sustainability analysis.

while consistently outperforming a large set of strong baselines by more than 10 percentage points in mean average precision.

sustain.AI is released to the public as a KI-NRW demonstrator, which is available at `https://sustain.ki.nrw/`. First user tests have already promised significant efficiency gains for the analysis of sustainability reports in the context of auditing. Moreover, the continuous use in production will further improve the system's recommendation capabilities due to the integration of human feedback, e.g., in the form of correcting wrong predictions.

7.2 Related Work

Before continuing with the description of the inner workings of sustain.AI, we take a look at prior accomplishments of other researchers related to this work.

In terms of facilitating the audit of annual financial statements, [27] presented the ALI tool, a recommender system that ranks textual elements of financial documents to associated requirements of predefined regulatory frameworks like IFRS or HGB. For the ranking task, the authors used classical NLP techniques like TF-IDF [57], latent semantic indexing, neural networks, and logistic regression (LR) with the combination of the first and last methods giving the best performance. In follow-up work, [28] improved ALI by utilizing a pre-trained BERT [34] language model as the backbone to encode text segments. Our architecture extends this approach by including weighted random sampling in the training process which speeds up the model convergence time and improves the overall performance.

When it comes to the NLP-based analysis of sustainability or CSR reports, different aspects

have been researched. [147] and [148] addressed the problem of automatically evaluating the GRI- and ESG[1]-accordance of CSR-reports. Both applied unsupervised text similarity measures building on GloVe embeddings [53]. Similarly, [149] leveraged the language model RoBERTa [92] to predict the relevance of sustainability reports according to the sustainable development goals in the USA. Specifically targeted for the banking sector, [150] developed a rule-based NER approach to estimate an index that displays the level of compliance of the climate-related financial disclosures with the TCFD[2] recommendations.

7.3 Methodology

In this section, we formally define the problem of matching text segments within documents to relevant legal requirements before turning to the in-depth analysis of our proposed architecture, which is visualized in Figure 7.2.

7.3.1 Problem Formulation

Given a sustainability report consisting of N distinct text segments $\mathcal{S}$, e.g., paragraphs, titles, tables, or diagrams, and a set of M regulatory checklist requirements $\mathcal{R}$, our goal is to identify all semantically relevant text segments for each requirement. Since the number of requirements M is static, but each document has a different length (number of text segments) N, we initially model the described matching task from a segment-to-requirements perspective as a multi-label classification problem. Formally, for every $s_i \in \mathcal{S}$ our recommender model assigns relevance scores to all $r_j \in \mathcal{R}$.

However, from the users' point of view, the reverse direction of getting relevant segment recommendations for a specific requirement r_j (requirement-to-segments perspective) is far more beneficial. This is especially true because a significant amount of text segments within a sustainability report is unrelated to concrete requirements in $\mathcal{R}$. This is why, based on the assigned relevance scores, our model ranks the text segments per requirement in descending order and subsequently recommends the top K relevant text blocks to the user.

7.3.2 Document Parsing

Before we focus on the actual recommender module, the core component of sustain.AI, we briefly touch upon the non-negligible task of document parsing. The large majority of publicly available sustainability reports are published as PDF documents, an inherently difficult format to convert into a structured machine-readable form like XML or JSON. The latter is particularly true for scanned PDF reports that only contain image information.

To solve this issue our system utilizes a custom PDF parser (see Figure 7.2), that is capable of parsing machine-created as well as scanned PDFs with arbitrarily complex formattings. The parser leverages a refined image segmentation technique by combining the powerful object detection network Faster R-CNN [11] with the density-based clustering algorithm DBSCAN [151]. It is also trained to recognize specific elements of a document, such as footers, headers, or pagination. For further details about the parser's functionality, we refer to [152].

[1] Environmental, Social and Governance factors.
[2] Task Force on Climate-related Financial Disclosures.

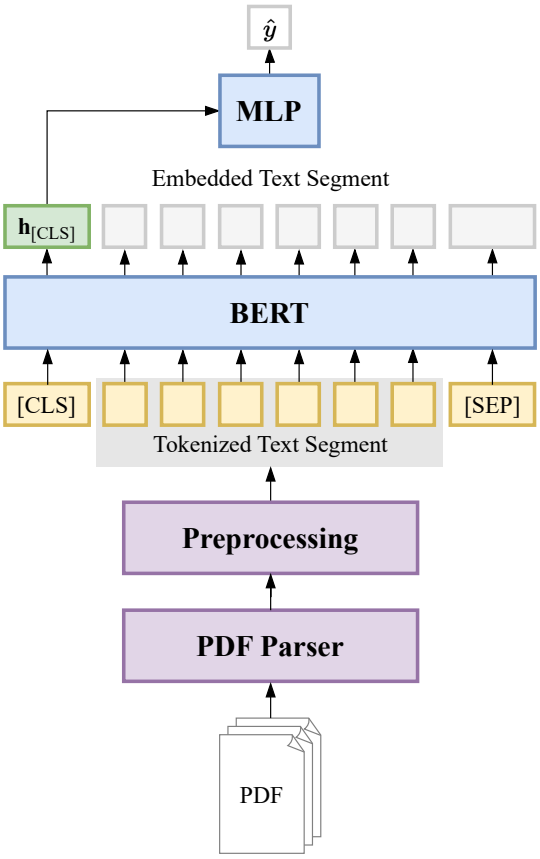


Figure 7.2: Schematic visualization of the recommender system and the data flow in sustain.AI. A custom PDF parser processes the raw sustainability reports. After some textual clean-ups, a fine-tuned BERT model encodes individual text segments that are subsequently matched to relevant regulatory requirements.

After the successful PDF parsing, we apply some basic textual preprocessing in the form of removing line break hyphens and filtering out irrelevant text segment types like footer, header, and table of contents. Our final set of considered segments $\mathcal{S}$ consists of titles, paragraphs, enumerations, tables, and diagrams.

7.3.3 Recommender System

Considering a parsed and processed sustainability report, we use a pre-trained BERT [34] model to individually encode each text segment $s_i \in \mathcal{S}$.

Formally, we first apply WordPiece [120] tokenization to transform an exemplary input segment s into a sequence of sub-word tokens $t = ([\text{CLS}], t_1, \ldots, t_n, [\text{SEP}])$. Note [CLS] denotes a BERT-specific special token that aggregates the content of the entire segment while [SEP] simply highlights the end of the sequence.

Passing t to the BERT model with pre-trained parameters $\boldsymbol{W}_{\text{bert}}$ yields a sequence of contextual token embeddings $\boldsymbol{h}_{[\text{CLS}]}, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_n, \boldsymbol{h}_{[\text{SEP}]}$, where $\boldsymbol{h}_{[\text{CLS}]}$ represents the aggregated

context hidden state for the whole segment s.

Subsequently, we employ an MLP with trainable parameters $\boldsymbol{W}_{\text{mlp}}$ to predict relevance probabilities $\hat{\boldsymbol{y}} = [\hat{y}_1, \ldots, \hat{y}_M] \in \mathbb{R}^M$ for all requirements in $\mathcal{R}$. The classifying MLP consists of a fully connected hidden layer followed by dropout [123] and ReLU (Rectified Linear Unit) [143] activation functions and a sigmoidal output layer.

During training, we jointly optimize and fine-tune the parameters of the BERT model $\boldsymbol{W}_{\text{bert}}$ and the classification layer $\boldsymbol{W}_{\text{mlp}}$ to minimize the Binary Cross-Entropy (BCE) loss between target labels $\boldsymbol{y}$ and predicted probabilities $\hat{\boldsymbol{y}}$.

Finally, after assigning relevance scores over requirements for all $s_i \in \mathcal{S}$, we sort the segments for each requirement r_j in descending order to recommend the top K relevant text blocks.

7.4 Experiments

In the following sections, we introduce our two custom data sets of German sustainability reports, define our evaluation metrics, discuss the overall training setup, describe the competing baseline methods, and finally, evaluate results.

7.4.1 Data

We train and evaluate our algorithms on two novel sustainability reporting data sets.

The first data set, named GRI, consists of 92 published sustainability reports from major German companies. The reports have been sourced in PDF format from the companies' websites. After the parsing step domain experts from the auditing industry annotated all text segments in accordance with the requirements of the GRI standards. Concretely, we consider the 89 indicators of the GRI topic standards, which cover the three main categories, economy, environment, and social that are further split into granular topics like anti-corruption, energy consumption, and human rights assessment. The annotation workload was equally split among three auditors which were supervised by a senior auditor. In multiple iterations, the created requirement labels have been validated and refined via double-checking randomly selected sample annotations and a qualitative inspection of the false positive and negative model predictions.

The second data set, named DNK, leverages the public sustainability reporting database from the German Sustainability Code[3] (DNK). The platform is used by the majority of German companies to annually disclose their sustainability activities with respect to 33 requirements from 20 DNK criteria, e.g., usage of natural resources and human rights. The categories and their requirements cover most of the GRI topics but are generally less granular. In contrast to the PDF documents of the GRI data set, the DNK reports in HTML format follow a predefined structure where each section of text segments answers a distinct requirement. Since the requirement descriptions precede their respective sections we can automatically retrieve the ground truth annotations from the HTML during the parsing process.

Table 7.1 displays descriptive statistics for both data sets. Due to the smaller amount of training documents, the greater document size, and the annotation sparsity, we consider the GRI data set the harder challenge for our models. We separately train, optimize and evaluate our

[3] `https://www.deutscher-nachhaltigkeitskodex.de/Home/Database`.

Table 7.1: Properties of our GRI and DNK data sets. We display the number of requirements and documents, the average number of segments per document, the average percentage of segments assigned to at least one requirement, and the average number of matched segments per requirement.

Data set	GRI	DNK
# requirements	89	33
# documents	92	1779
# segments s per document	972	242
% segments s matched	9	100
# matched segments s per requirement	2.7	7.3

algorithms on both data sets to verify this hypothesis, investigating how well sustain.AI handles different sizes of training data and number of labels. For our GRI and DNK experiments, we employ fixed training, validation and testing splits of 65-15-20 and 70-15-15, respectively.

As a contribution to the open-source community and for further research concerning German sustainability reports we make the DNK data set publicly available[4].

7.4.2 Evaluation Metrics

We quantitatively evaluate all models by calculating modified Mean Sensitivity (MS) and Mean Average Precision (MAP) scores for the top K recommendations. While MAP punishes the lower-ranked recommendations of relevant segments, MS only considers whether the relevant segments are contained in the set of recommendations. For a single document and a concrete requirement r_j the modified sensitivity $\mathrm{S}(K)$ from [27] and the average precision $\mathrm{AP}(K)$ are respectively defined as:

$$\mathrm{S}(K) = \frac{|\text{top } K \text{ recommendations} \cap L \text{ annotations}|}{\min(K, L)}, \tag{7.1}$$

$$\mathrm{AP}(K) = \frac{1}{\min(K, L)} \sum_{i=1}^{K} \left(\mathrm{P}(i) \cdot \mathrm{rel}(i)\right), \tag{7.2}$$

where L denotes the number of relevant segment annotations, $\mathrm{rel}(i)$ indicates whether the i^{th} recommendation is relevant ($\mathrm{rel}(i) = 1$) or not ($\mathrm{rel}(i) = 0$), and

$$\mathrm{P}(i) = \frac{|\text{top } i \text{ recommendations} \cap L \text{ annotations}|}{i} \tag{7.3}$$

represents the precision score considering the top i recommendations. Averaging $\mathrm{S}(K)$ and $\mathrm{AP}(K)$ over all checklist requirements $r_j \in \mathcal{R}$ and documents yields the subsequently reported mean sensitivity $\mathrm{MS}(K)$ and mean average precision $\mathrm{MAP}(K)$ metrics.

[4] `https://github.com/LarsHill/dnk-dataset`.

Table 7.2: Evaluated hyperparameter configurations of sustain.AI. The best configuration on the validation set is highlighted in boldface.

Hyperparameter	Configurations
MLP hidden dimensions	None, 512, **1024**, 2048
Dropout	0.0, 0.1, **0.3**, 0.5,
Batch size	2, 4, **8**, 16
Learning rate	1×10^{-6}, $\mathbf{1 \times 10^{-5}}$, 1×10^{-4}

7.4.3 Training Setup

In this section, we shed light on the training process and the hyperparameter optimization of sustain.AI.

For all evaluated models we conduct an exhaustive grid search comparing various parameter combinations based on their validation set MAP(3) performance to determine the best training setup. Table 7.2 highlights the explored ranges and respective best values of sustain.AI's tuned model parameters.

As encoding backbone we employ a $\text{BERT}_{\text{BASE}}$ model, published by the MDZ Digital Library team (dbmdz)[5]. It mirrors the architectural setup of the English $\text{BERT}_{\text{BASE}}$ counterpart[6] and is pre-trained on a large corpus of German books, news reports and Wikipedia articles. We train our model and all neural network-based baselines via gradient descent utilizing the AdamW [122] optimizer with a linear warmup of 10% and a linearly decaying learning rate schedule. Additionally, we apply weight decay of 0.01 and gradient clipping with a maximum value of 1. We also analyze different learning rates, batch sizes, levels of dropout regularization [123], and MLP hidden dimensions, as can be seen in Table 7.2. For all training runs we set a random seed of 42 and fix the maximum number of epochs to 15 while applying early stopping with a patience of 3 epochs.

Due to the small percentage of annotated segments s in the GRI data set (9%, see Table 7.1) we employ WRS with replacement to expose these relevant segments more frequently during training. Concretely, we alter the originally uniform sampling probability of each segment to the normalized inverse frequency of relevant $+$ or irrelevant $-$ occurrences in the training set.

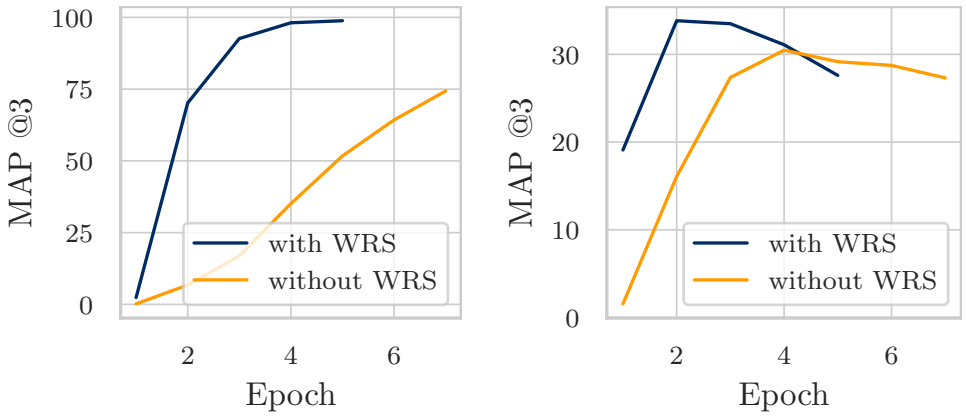
Figure 7.3 showcases the benefits of integrating WRS into the model training process for the GRI data set. We achieve a much faster training convergence and thus, save a considerable amount of training time and compute power benefitting from early stopping. At the same time, our model's MAP(3) score on the validation set increases by 3 percentage points.

7.4.4 Baselines

We compare sustain.AI's end-to-end recommender model from Section 7.3.3 with 4 competing baseline architectures. For a fair comparison, all baselines make use of weighted random sampling concerning the imbalanced GRI data set.

[5] `https://huggingface.co/dbmdz/bert-base-german-cased`.

[6] 12 multi-head attention layers with 12 attention heads per layer and 768-dimensional output embeddings.



(a) Training (b) Validation

Figure 7.3: Positive impact of weighted random sampling (WRS) on training convergence and validation performance. We report the mean average precision considering the top 3 recommendations (MAP(3)) with and without WRS.

First, we utilize word frequency-based TF-IDF [153] representations that have been fitted on our respective training corpora. Prior to training, all segments have been preprocessed in terms of lowercasing, punctuation- and digit removal as well as stemming. The resulting 8000-dimensional segment vectors are then used as input for an ensemble of one-vs-rest binary LR classifiers. Each classifier is trained for a specific requirement r and a maximum of 100 iterations using the "liblinear" solver from the `scikit-learn` `Python` library.

Second, we pass the same TF-IDF representations into an MLP with one hidden layer of dimensionality 1024. In contrast to the binary logistic regression heads, the MLP performs multi-label classification and predicts the relevant requirements simultaneously. We find an optimal batch size of 64 and a learning rate of $1e{-}3$.

Third, we exchange the TF-IDF input vectors with frozen contextual embeddings from sustain.AI's BERT model. As classifiers, we evaluate the previously defined MLP and a GRU. While the MLP takes BERT's CLS output embedding as input, the bidirectional GRU processes the resulting token representations of the frozen BERT model. Specifically, the last/first hidden state of the forward/backward GRU is concatenated and passed to a sigmoidal output layer. Optimal settings are obtained with a hidden size of 512 neurons, a batch size of 8, and a learning rate of $1e{-}5$.

7.4.5 Results

We evaluate and compare sustain.AI and all baseline methods on the previously specified hold-out test set for both the GRI and DNK data. Table 7.3 reports MS and MAP scores for the top 3 and top 5 recommendations.

First, it can be seen that the overall DNK performance across all methods is much better compared to the GRI data. This was expected, considering the reduced number of requirements and the larger amount of training documents and annotations.

Second, we find that the application of WRS during training significantly improves the test

Table 7.3: Test set results for the recommendation of relevant segments in GRI and DNK sustainability reports. sustain.AI outperforms all competing baselines in top 3/5 Mean Sensitivity (MS) and Mean Average Precision (MAP).

in %	GRI				DNK			
Model	MS		MAP		MS		MAP	
	3	5	3	5	3	5	3	5
Tf-Idf + LR	24.3	33.7	17.1	19.5	70.8	66.3	66.8	59.0
Tf-Idf + MLP	33.0	39.8	22.6	24.2	77.4	77.8	74.8	71.8
$\text{BERT}_{\text{frozen}}$ + MLP	28.4	36.1	21.0	22.4	75.2	70.6	73.5	66.4
$\text{BERT}_{\text{frozen}}$ + GRU	28.1	36.8	20.5	22.1	84.0	80.2	83.0	77.2
$\text{sustain.AI}_{\text{no WRS}}$	35.5	44.2	28.4	30.5	**90.3**	**87.8**	**89.7**	**86.1**
$\text{sustain.AI}_{\text{WRS}}$	**48.0**	**53.8**	**35.9**	**37.0**	–	–	–	–

– = not applicable, since weighted random sampling (WRS) is only applied on GRI data.

set performance of our model. Compared to the version without WRS all metrics have increased by more than 6 percentage points. To enable a fair comparison we apply WRS during the training process of all baseline methods. Also, WRS is solely employed for the GRI data, since the DNK reports do not exhibit any annotation scarcity.

Finally, the results in Table 7.3 show the overall superiority of sustain.AI's end-to-end architecture, outperforming all baselines by a large margin.

7.5 Conclusion and Future Work

We presented sustain.AI, an interactive, AI-powered tool for the semi-automated analysis of German sustainability reports. Our transformer-based model achieves promising results both on the well-structured DNK data set and on the real-world GRI data, compared to a number of strong baselines. Qualitative exploration of the results also suggests that it is indeed helpful in analyzing those long documents. The tool is planned to be deployed on an online platform soon and will then be openly accessible to the public.

Future work includes improving the current model with additional annotated data, which can easily be inferred from the user feedback we will collect through the tool. We also plan to extend the framework to English reports, as currently, only the processing of German documents is possible. Another idea for improvement is to extract specific numeric KPIs from the reports, such as different types of CO_2 emissions, water consumption, or indicators for social welfare.

CHAPTER 8

Enhancing Large Language Models with Paragraph-Level Awareness

In the previous chapter, we presented sustain.AI, a BERT-based [34] text classification system designed to recommend the most semantically relevant text passages from sustainability reports for given disclosure requirements from the Global Reporting Initiative (GRI) sustainability standard. This model employed a classification approach that generated embeddings for individual paragraphs using BERT and used these embeddings to match text segments to the appropriate requirements. While the model achieved excellent performance, it operated under a significant limitation: the absence of paragraph-level contextual awareness.

Specifically, the prior approach processed each paragraph in isolation. Each paragraph was independently transformed into an embedding using BERT, and the model considered only this single-paragraph representation for classification. This independent processing meant that the model did not take into account the surrounding textual context, such as preceding or succeeding paragraphs, which often contain crucial information influencing the meaning and relevance of a passage. As a result, the model might miss subtle contextual cues and relationships between paragraphs that are essential for accurate classification, particularly in documents where the narrative develops across multiple sections.

To address this limitation, we introduce a novel pre-training methodology called *Pointer-Guided Segment Ordering* (SO) in this chapter. Our approach enhances the language model's ability to generate contextually rich paragraph embeddings by training it to comprehend narrative flow and inter-paragraph relationships. By reconstructing the original order of shuffled text segments using a self-attention-based [19] pointer network, the model learns to understand how paragraphs relate within a document. This task effectively infuses the model with paragraph-level contextual awareness. When combined with standard pre-training techniques like Masked Language Modeling (MLM), our method enables the model to capture both local and global contextual information.

Furthermore, we implement a dynamic sampling strategy during fine-tuning to increase the diversity of training instances across epochs, thereby improving sample efficiency. This strategy is particularly beneficial for smaller fine-tuning datasets characterized by long documents, as it mitigates the risk of overfitting and promotes better generalization by exposing the model to a wider range of paragraph sequences.

Through extensive experiments, we demonstrate that models pre-trained with our pointer-guided SO task consistently outperform competing baselines. Our approach sets new state-of-the-art results on the task from the previous chapter and in the scientific literature domain, highlighting the advantages of incorporating paragraph-level context and advanced pre-training techniques in enhancing sequential text classification performance.

In the following sections, we delve deeper into our methodology, provide detailed experiments to validate our approach, and discuss the implications of our findings in the context of NLP advancements.

This chapter is based on the following publication:

- L. Hillebrand, P. Pradhan, C. Bauckhage, and R. Sifa, "Pointer-Guided Pre-Training: Infusing Large Language Models with Paragraph-Level Contextual Awareness," *Proc. ECML PKDD*, 2024, DOI: `10.1007/978-3-031-70359-1_23` [38].

Lars Hillebrand originated and refined the methodology, which encompassed the novel self-supervised SO pre-training and dynamic sampling to enhance downstream performance in sequential text classification. He was responsible for implementing the codebase in `Python` using `PyTorch` [94] for deep neural network modeling, training, and evaluation, while also executing the experiments, and conducting data processing. Lars Hillebrand also evaluated the results, drafted the manuscript, and finalized the work after including valuable feedback from all co-authors.

8.1 Introduction

The landscape of NLP has been profoundly transformed by the emergence of generative LLMs such as OpenAI's GPT series [18, 91], Mixtral [89], and Llama [154]. These models have set new benchmarks across a wide range of NLP tasks, showcasing remarkable capabilities in understanding and generating human language. Despite the significant advancements achieved by these large-scale models, there remains an equally important domain for smaller, specialized language models that excel in fast retrieval and semantic search, particularly those that generate precise paragraph and section representations. This domain is crucial, especially in the context of Retrieval-Augmented Generation (RAG) [35], where the integration of retrieval mechanisms with generative models enhances the reliability and informativeness of the output.

In this work, we address the important area of representation learning for improved paragraph-level contextual understanding, which is critical for enhancing the capabilities of NLP systems in sequential text classification and retrieval-based applications such as semantic text search.

At the heart of our approach is the introduction of a novel pre-training methodology, named "pointer-guided segment ordering" (SO). This technique is designed to infuse language models with a deep awareness of paragraph-level context. Utilizing a self-attention-driven pointer network, the pointer-guided SO task challenges the model to reconstruct the original sequence of shuffled text segments (see Figure 8.1). This complex task enables the model to develop a nuanced comprehension of narrative flow, coherence, and contextual relationships, significantly enhancing its ability to understand and represent paragraph-level context when combined with standard pre-training techniques like masked language modeling.

To complement our pre-training methodology, we further introduce dynamic sampling during the fine-tuning phase. This sampling technique increases the diversity of training instances across

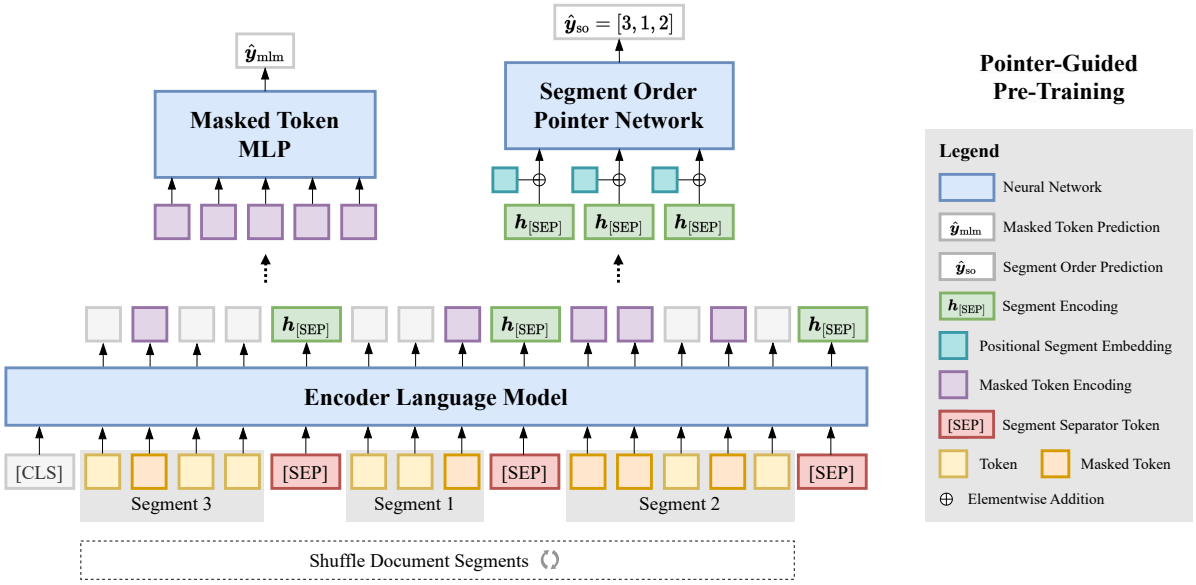


Figure 8.1: Schematic visualization of our "Pointer-Guided Pre-Training" methodology. During pre-training a self-attention-based pointer network classification head learns to reconstruct the original order of shuffled text segments based on their hidden state representations ($\boldsymbol{h}_{[\text{SEP}]}$). Employing this segment ordering (SO) pre-training mechanism alongside masked language modeling (MLM) increases the segment-level contextual awareness of the encoding language model and subsequently improves its downstream classification capabilities.

epochs, thereby improving sample efficiency. Dynamic sampling is particularly advantageous for smaller fine-tuning datasets characterized by long documents, where it effectively mitigates the risk of overfitting and promotes better generalization.

We demonstrate the effectiveness of these contributions through extensive experiments. We show that models pre-trained with our pointer-guided SO task consistently outperform competing baselines and raise the state-of-the-art across various datasets and tasks in the scientific literature and financial reporting domain. Furthermore, the model-agnostic nature of our methodology positions it to capitalize on future advancements in language model design, promising further improvements in paragraph-level text representation.

In summary, our work not only introduces a novel and effective methodology for enhancing paragraph-level embeddings but also establishes a new benchmark for sequential text classification. By opening new avenues for research in leveraging document structure for enhanced language modeling, our work marks a significant step forward in the ongoing evolution of NLP, with substantial potential impact on retrieval-based applications like semantic text search.

In the following, we first review related work, before describing our modeling approach in Section 8.3. In Section 8.4, we outline our experiments, describe our datasets, and discuss the results. Section 8.5 then draws a conclusion and provides an outlook into conceivable future work.

8.2 Related Work

Traditional language modeling tasks such as MLM and next token prediction (NTP) have been instrumental in learning token-level representations. Models like RoBERTa [92], ELECTRA [155], and GPT variants [18, 91, 154] have shown significant success in these areas. However, these models often lack mechanisms to enforce the learning of meaningful segment-level representations, crucial for understanding paragraph-level context. BERT [34] introduced the next sentence prediction (NSP) task to bridge this gap, but its simplicity limited its effectiveness. Subsequent models, such as RoBERTa, abandoned NSP due to its limited contribution to model performance. Unlike these approaches, our work introduces a pointer-guided segment ordering methodology that directly leverages the inherent structure of textual data, offering a novel way to enhance paragraph-level understanding without relying on external data sources like Wikipedia article links in LinkBERT [156] or knowledge-graph reasoning [157]. The underlying architecture of our method, the pointer network [158], has been successfully used for stand-alone sequence ordering tasks, as demonstrated by [159–161]. However, to the best of our knowledge, we are the first to employ a novel self-attention-driven pointer network for segment ordering in conjunction with LLM pre-training.

We evaluate our pre-training technique on several sequential text classification tasks. Previous studies have tackled these challenges with domain-adapted and fine-tuned BERT models [29, 162] and the incorporation of hierarchical LSTMs, attention mechanisms, and CRF layers for improved sequential label dependency handling [163–166].

8.3 Methodology

In this section, we provide a comprehensive description of our methodological contributions aimed at enhancing the contextual sensitivity of paragraph-level text representations, as well as their optimization for various downstream applications. Initially, we describe our novel "pointer-guided segment ordering" approach, a versatile pre-training strategy that employs a self-attention-driven pointer network to accurately restore the original sequence of shuffled text segments. Subsequently, we detail our fine-tuning methodology, which incorporates dynamic sampling to augment the diversity of training instances throughout successive training epochs, thereby improving sample efficiency.

8.3.1 Pointer-guided Segment Ordering

A text document is inherently composed of consecutive text segments, which can range from whole paragraphs and individual sentences to enumerations, tables, and headlines. These segments are typically contextually interdependent, forming a coherent narrative in various types of documents, such as news articles, fiction novels, annual reports, or legal contracts.

To capture the essence of this structural coherence, we propose a novel self-supervised pre-training technique denoted as pointer-guided segment ordering (SO) that is capable of leveraging large amounts of unlabeled text data to infuse language models with additional embeddings for individual text segments. Concretely, we employ a self-attention-based pointer network to reconstruct the original order of a randomly shuffled sequence of text segments. This non-trivial pre-training task becomes exponentially more complex as the number of segments

increases. Given a document of N consecutive text segments, the number of possible segment permutations grows factorially to $N!$. This inherent complexity requires the model to gain a deep contextual understanding, picking up on nuanced intricacies like coherence, chronological order, and causal relationships to ensure that the narrative flows logically and maintains continuity from beginning to end.

To address the fact that transformer-based language models [19] typically exhibit an upper limit on the maximum token context size[1], denoted as C, we start with dissecting long text documents into individual training samples. Concretely, a training sample consists of K text segments s, where K is the maximum number of segments that fit within the language model's context window. Each segment is appended with a special delimiting token, [SEP], that indicates the segment's end. Note the value of K varies for each sample, depending on the token length of the individual segments.

We enable the segment ordering task by randomly shuffling the segments within each training sample before encoding the entire sequence with a bidirectional language model denoted as `BiLM`. Specifically, we first apply WordPiece [120] tokenization to transform an exemplary input sample consisting of K segments into a sequence of sub-word tokens $t = ([\text{CLS}], s_1, [\text{SEP}]_1, s_2, [\text{SEP}]_2, \ldots, s_K, [\text{SEP}]_K)$. [CLS] denotes the special start of sequence token and an individual segment $s_i = (t_1, \ldots, t_m)$ consists of m sub-word tokens, where m can differ between segments.

We couple our segment ordering pre-training task with MLM to enhance the model's understanding of context and word relationships. In line with the insights from [167] and [168], we implement whole word masking and mask 15% of randomly selected whole words. For the remaining MLM pre-training methodology of predicting the correct sub-words from a given vocabulary for all masked tokens, we refer to [34].

The tokenized, masked, and permuted input sequence is then encoded by a `BiLM`, which yields a series of d-dimensional hidden state vectors $\boldsymbol{H} = (\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_T) \in \mathbb{R}^{T \times d}$ corresponding to each token t_i for a sequence of length T.

For the segment reordering task, we collect the hidden states corresponding to the [SEP] tokens, denoted as $\boldsymbol{H}_{[\text{SEP}]} = (\boldsymbol{h}^1_{[\text{SEP}]}, \boldsymbol{h}^2_{[\text{SEP}]}, \ldots, \boldsymbol{h}^K_{[\text{SEP}]}) \in \mathbb{R}^{K \times d}$. We add learnable absolute positional embeddings $\boldsymbol{E} = (\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_K)$ to each segment hidden state, yielding the enhanced segment representations $\boldsymbol{H}'_{[\text{SEP}]} = \boldsymbol{H}_{[\text{SEP}]} + \boldsymbol{E}$. Naturally, the added positional bias encodes the new segment position after shuffling, preventing the reordering task from being compromised.

Subsequently, we pass $\boldsymbol{H}'_{[\text{SEP}]}$ to a pointer network [158], which is particularly suited for our SO task due to the varying number of segments per sample K, which precludes the use of a static output layer with a fixed number of classes. The network calculates each segment's probability distribution over the original segment positions using a multiplicative self-attention mechanism, defined as follows:

$$\boldsymbol{A} = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{q}}\right), \quad \boldsymbol{Q} = \boldsymbol{H}'_{[\text{SEP}]}\boldsymbol{W}^\top_{\text{query}}, \quad \boldsymbol{K} = \boldsymbol{H}'_{[\text{SEP}]}\boldsymbol{W}^\top_{\text{key}}, \tag{8.1}$$

$$\boldsymbol{W}_{\text{query}} \in \mathbb{R}^{q \times d},\ \boldsymbol{W}_{\text{key}} \in \mathbb{R}^{q \times d},\ \boldsymbol{Q} \in \mathbb{R}^{K \times q},\ \boldsymbol{K} \in \mathbb{R}^{K \times q},\ \boldsymbol{A} \in \mathbb{R}^{K \times K}, \tag{8.2}$$

[1] The self-attention mechanism incurs a computational cost that scales quadratically in sequence length, imposing practical limits on the processable context size.

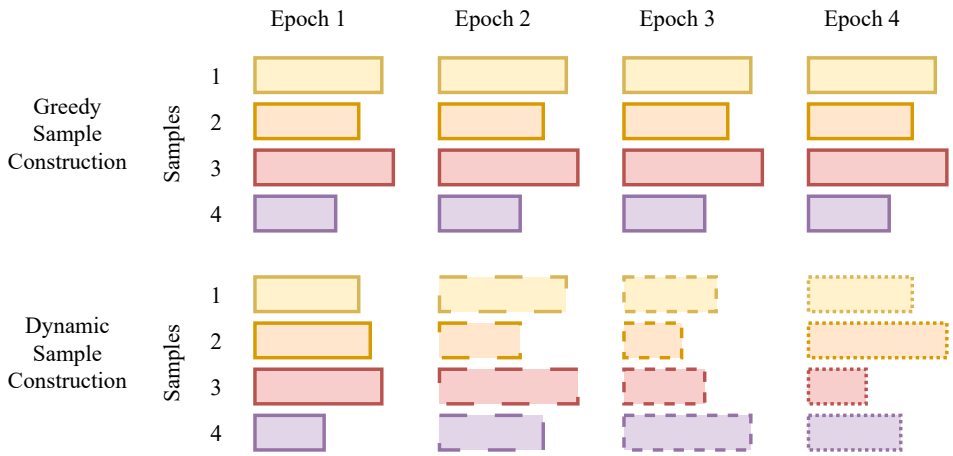


Figure 8.2: Schematic visualization of "Dynamic Sampling". Dynamic sample construction increases sample diversity across epochs which improves contextual understanding and model generalization in contrast to simply creating samples greedily.

where $\boldsymbol{W}_{\text{query}}$ and $\boldsymbol{W}_{\text{key}}$ are learnable query and key weight matrices, $q = d/4$ is their respective row dimension, and $\boldsymbol{A}$ is the row-stochastic[2] square attention matrix, with each element a_{ij} denoting the predicted probability that segment i originated from position j. It follows that the predicted position of segment i equals $\hat{y}_i = \arg\max_j(\boldsymbol{a}_i)$.

The loss for the segment ordering task is computed using negative log-likelihood, $\mathcal{L}_{\text{SO}}(\boldsymbol{A}, \boldsymbol{y}) = -\sum_{i=1}^{K} \log(a_{i,y_i})$, where y_i denotes the ground truth position of segment i.

8.3.2 Sample-efficient Fine-Tuning using Dynamic Sampling

Based on the previously detailed concept of combining multiple text segments to improve contextual understanding, this section focuses on the associated benefits of sample efficiency and introduces dynamic sampling to enhance data diversity.

Traditional text classification fine-tuning approaches for encoder-only language models like BERT often treat each text segment as an independent sample, which can lead to suboptimal context utilization and unnecessary computational overhead, especially for short segments. Our method dynamically combines multiple text segments into a single sample, thereby maximizing the use of the model's context capacity C and enhancing training efficiency.

For an average segment length of $\bar{T}$ tokens, the maximum number of segments per sample $K = \left\lfloor C/\bar{T} \right\rfloor$ represents the efficiency gain factor, which quantifies the improvement over processing segments individually. This gain is more evident when using large batch sizes B, as the longest sample in a batch dictates the memory and computational requirements.

A drawback of uniting multiple segments in one sample is the reduction in sample diversity, which is particularly problematic in small datasets. To mitigate this issue and promote sample diversity, we introduce dynamic sampling for fine-tuning in scenarios with scarce data (see

[2] $\sum_{j=1}^{K} a_{ij} = 1 \quad \forall i \in \{1, 2, \ldots, K\}$.

Figure 8.2). Instead of deterministically merging the maximum number of segments K, we randomly select the number of combined segments L, sampling from a uniform distribution $\mathcal{U}(L_{\min}, K)$, where $L_{\min}$ denotes the minimum number of merged segments. While this reduces the expected computational efficiency gain per sample, it introduces beneficial randomness into the training process. By exposing the model to varying segment combinations of different lengths during each epoch, we encourage better generalization and reduce the risk of overfitting.

Subsequently, our experiments validate that combining segment order pre-training with sample-efficient fine-tuning using dynamic sampling significantly enhances performance in downstream text classification tasks that require a comprehensive understanding of complex and extended document structures.

8.4 Experiments

We split our experiments into two parts. First, we focus on our pointer-guided pre-training setup before quantitatively evaluating its impact on five downstream tasks requiring sequential text classification.

The experiments were conducted on a GPU cluster equipped with eight 32 GB Nvidia Tesla V100 GPUs. The cumulative pre-training duration of all models amounted to 380 GPU hours. Our code is implemented in `PyTorch` [94], with the initial weights of pre-trained models being loaded from Huggingface. We open-source our code base on GitHub[3].

8.4.1 Pre-Training

In the following, we briefly introduce our pre-training datasets and discuss the overall training setup including baselines and results.

Data

Table 8.1 details the mixture of our pre-training datasets and reports various descriptive statistics like the number of tokens and segments per dataset.

The `Wikipedia` datasets comprise 5.9 (English) and 2.4 (German) million articles respectively that were directly retrieved from their rendered HTML pages. In contrast to the commonly used Wikimedia XML dumps, our corpus resolves Wikipedia's templating syntax embedded in the dumps and thus represents the articles in their original form leading to improved data quality.

`Bundesanzeiger` contains 1.9 million German corporate annual reports from the Bundesanzeiger, a platform where German companies are mandated to publish their legally required documents. Compared to Wikipedia the average document length is roughly three times higher resulting in around 2000 tokens per report.

Lastly, we include two proprietary datasets of English (2 million) and German (650 thousand) `news` articles that raise the total number of pre-training tokens to 12.24 billion.

For all datasets, we employ a 90-10 split for training and validation. We parse raw HTML articles using the lxml Python library, distinguishing headlines, paragraphs, tables, enumerations, and more.

[3] `https://github.com/LarsHill/pointer-guided-pre-training`.

Table 8.1: Descriptive statistics of pre-training datasets with document, segment, sample, and token counts in English and German, including total and average values. Token and sample statistics are calculated based on the multilingual word-piece vocabulary, custom-100K (see Table 8.2), created from all pre-training datasets.

	Wikipedia		Bundesanzeiger	News		Sum
	EN	DE	DE	EN	DE	
Documents (M)	5.85	2.44	1.91	2.02	0.65	12.88
Segments (M)	99.37	19.63	85.06	36.25	17.51	257.81
Samples (M)	13.83	4.80	9.53	5.23	1.23	34.62
Tokens (B)	4.48	1.62	3.74	1.97	0.44	12.24
Tokens (%)	36.56	13.26	30.52	16.10	3.56	100

M: million, B: billion.

Training Setup and Results

We evaluate the efficacy of our pointer-guided segment ordering (SO) task by pre-training and fine-tuning three variants of the BERT language model: BERT, RoBERTa, and our proposed PointerBERT. The BERT model adheres to the original design by [34], employing self-supervised MLM and NSP. RoBERTa [92] modifies the BERT pre-training scheme by omitting NSP and maximizing the use of context by concatenating multiple text segments. PointerBERT extends RoBERTa with the inclusion of our SO task, as detailed in Section 8.3.1. Note the application of SO is architecture agnostic and can be used to generally enhance the paragraph-level contextual comprehension of bidirectional encoder language models like RNNs [169], DeBERTa [170] and Electra [155].

Table 8.2 presents the pre-training configurations for each model variant. All variants are based on Google's $\text{BERT}_{\text{BASE}}$ architecture as encoding backbone, differing only in their tokenizer and vocabulary construction.

Concretely, we distinguish three scenarios. First, we train each model from scratch on the English Wikipedia corpus (wiki-en) using Google's bert-base-cased tokenizer. The validation results demonstrate that PointerBERT correctly reorders shuffled segments in 35% of cases. Although this may appear modest, it is significantly higher than the baseline random guess accuracy of approximately $1/5040 \approx 2 \times 10^{-4}$, given an average of 7 segments per sample (see Table 8.1) and 5040 possible permutations. Importantly, the inclusion of SO does not compromise MLM performance, with only a marginal decrease in validation accuracy. Lastly, we observe that combining MLM with NSP significantly diminishes MLM performance, likely due to the reduced sample efficiency and the underutilization of the model's context capacity.

Second, we train RoBERTa and PointerBERT on the combined multilingual datasets, denoted "all", using a newly developed 100K token word-piece vocabulary created from the pre-training data. This step aims to assess the impact of SO in a multilingual context.

Third, we build upon the pre-trained checkpoints of Allen AI's SciBERT [171] and the German BERT model released by the MDZ Digital Library team (dbmdz)[4] and continue pre-training,

[4] `https://huggingface.co/dbmdz/bert-base-german-cased`.

Table 8.2: Training configurations and validation accuracies for all language model variations and their pre-training tasks, masked language modeling (MLM), next sentence prediction (NSP), and segment ordering (SO). The scores represent averaged batch accuracies across the validation set.

					Accuracy ↑ (%)		
Architecture	Datasets	Pre-trained	Train steps	Tokenizer	MLM	NSP	SO
BERT	wiki-en	✗	1×10^5	bert-cased	30.14	73.85	–
RoBERTa	wiki-en	✗	1×10^5	bert-cased	44.77	–	–
PointerBERT	wiki-en	✗	1×10^5	bert-cased	43.10	–	34.90
RoBERTa	all	✗	2×10^5	custom-100K	57.66	–	–
PointerBERT	all	✗	2×10^5	custom-100K	55.11	–	39.49
PointerSciBERT	wiki-en	✓	1×10^5	scibert-uncased	58.27	–	52.45
PointerBERT	all-de	✓	1×10^5	$\text{bert-cased}_{\text{dbmdz}}$	73.62	–	57.35

incorporating both MLM and SO. Here we aim to demonstrate the applicability of SO for already pre-trained language models and show that continuous pre-training with MLM and SO not only further increases the MLM performance but also manages to induce improved paragraph-level text understanding thanks to pointer-guided SO.

All models are trained using gradient descent with the AdamW optimizer [122], featuring a 10% linear warmup and a decaying learning rate schedule. We apply a weight decay of 0.01 and clip gradients at a maximum value of 1. The peak learning rate is set to 1×10^{-4}, with a batch size of 16 and gradient accumulation over 4 steps, resulting in an effective batch size of 64. Model performance is evaluated on a hold-out validation set every 5000 steps. Figure 8.3 provides a detailed view of the training progress, depicting both MLM and SO validation losses and accuracies.

Notably continued pointer-guided pre-training not only further improves MLM accuracy but also yields robust SO performance, which translates into enhanced results on downstream classification tasks, as discussed in the following section.

8.4.2 Downstream Fine-Tuning for Sequential Text Classification

In this section, we delve into the application of our pre-trained models to a series of fine-tuning downstream tasks that necessitate sequential text classification. We explore a diverse array of datasets, spanning both scientific literature and financial reporting domains, to assess the models' capabilities in categorizing text segments in different scenarios.

Datasets

Our evaluation encompasses five datasets, three from the scientific literature corpus and two from the financial reporting sector, each presenting unique challenges for text segment classification.

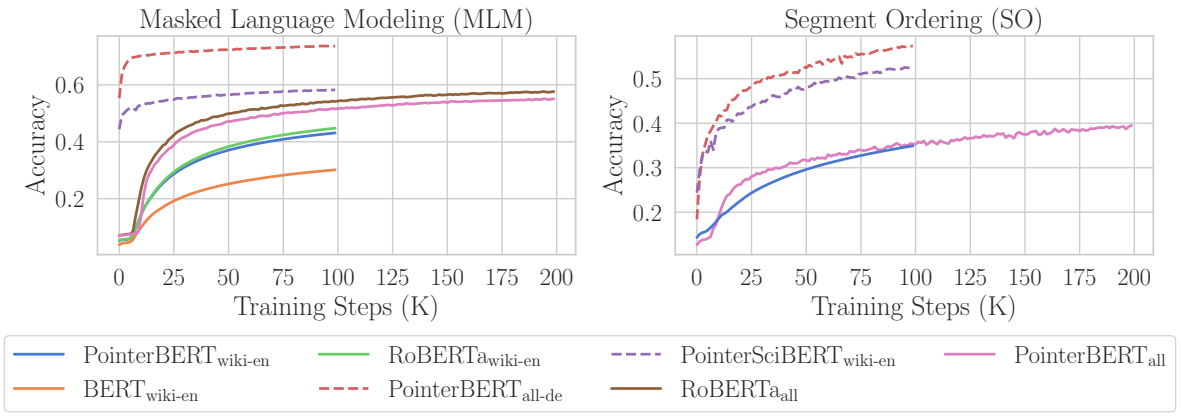


Figure 8.3: Pre-training progress for all model variants, showcasing validation accuracy curves for masked language modeling (MLM) and segment ordering (SO).

Within the scientific literature, we examine the `CSAbstruct` dataset [162], which comprises 2189 computer science abstracts with sentences annotated to discern their rhetorical roles. The `PubMed-RCT` dataset [172] extends our evaluation to 20 000 biomedical abstracts from PubMed, segmented into five rhetorical categories, following the preprocessing methodology outlined by [164]. The `Nicta` dataset [173] further contributes with 1000 biomedical abstracts, where sentences are classified according to the PICO framework (Population, Intervention, Comparison, Outcome) [174].

Transitioning to the financial sector, we incorporate the $\texttt{GRI}_{\texttt{DE}}$ dataset [29] (introduced in the previous Chapter 7), which consists of 92 sustainability reports from leading German companies. These reports were initially obtained as PDFs from corporate websites and subsequently annotated by experts to correspond with the GRI standards, covering 89 indicators across economic, environmental, and social dimensions. The $\texttt{IFRS}_{\texttt{EN}}$ dataset is composed of 45 English annual reports adhering to the International Financial Reporting Standards (IFRS). Provided by an auditing firm, these reports contain annotations that map paragraphs to 543 distinct legal requirements, with some paragraphs addressing multiple items.

A summary of the datasets' descriptive statistics is presented in Table 8.3. The scientific abstract datasets (CSAbstruct, PubMed-RCT, and Nicta) contrast with the financial report datasets (GRI_{DE} and IFRS_{EN}) in terms of structure. The former category includes a higher volume of documents, each with approximately 10 sentences, typically fitting within the model's context window of 512 tokens. This is reflected in the average number of samples per document being close to 1 for these datasets. They feature a smaller set of categories and require multi-class classification, where every text segment is annotated and assigned to a single category.

Conversely, the financial datasets contain fewer but significantly longer documents, averaging over 400 segments. The resulting classification complexity is further amplified by a larger number of checklist categories and annotation scarcity, with only 8.5% of paragraphs in the GRI_{DE} dataset linked to a GRI requirement. Moreover, both datasets contain segments that refer to multiple checklist items, making this task a multi-label classification challenge that resembles the difficulty of information retrieval due to the severe class imbalance and annotation sparsity.

Table 8.3: Descriptive statistics of scientific and financial fine-tuning datasets. Sample statistics are calculated based on the custom-100K vocabulary (see Table 8.2).

	Scientific Abstracts			Financial Reports	
	CSAbstruct	PubMed-RCT	Nicta	IFRS_{EN}	GRI_{DE}
Documents	2189	20 000	1000	45	92
Segments	14 708	240 386	9771	19 573	89 412
Samples	2191	23 289	1061	4773	21 306
Is multi-label	✗	✗	✗	✓	✓
Classes	5	5	6	543	89
Segments labeled (%)	100.00	100.00	100.00	78.37	8.57

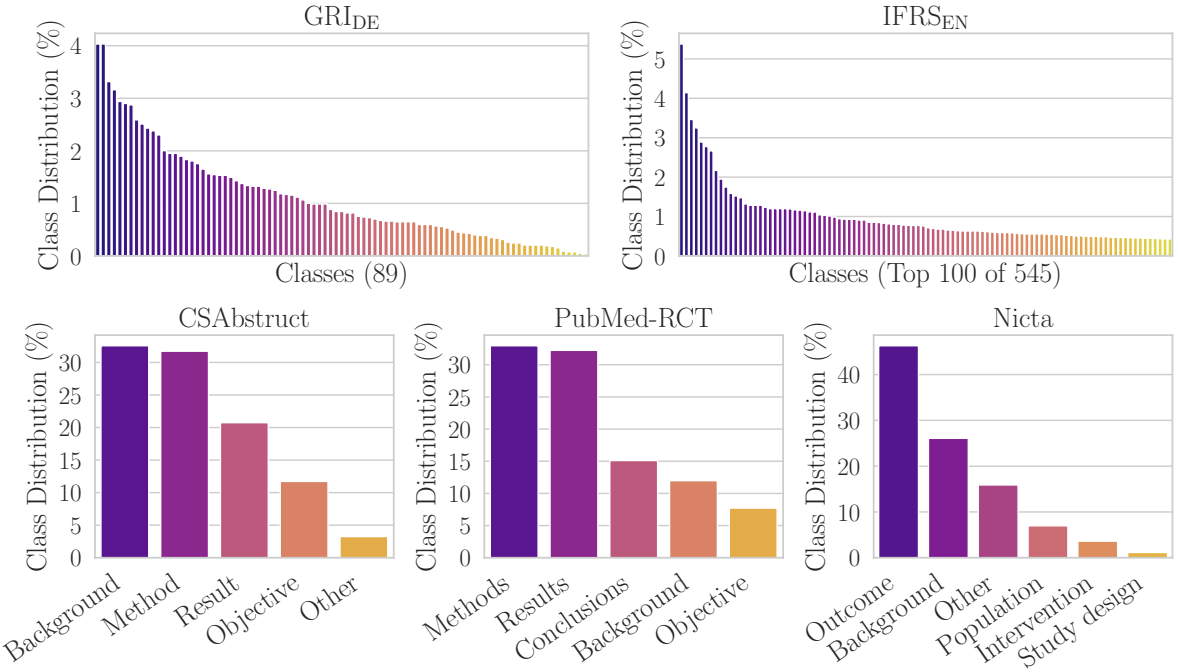


Figure 8.4: Class distributions across all datasets showcasing label imbalances.

The variation in class distribution across all datasets is graphically depicted in Figure 8.4. For all datasets, we adhere to the established training, validation, and test splits introduced in prior work. For the new IFRS_{EN} dataset, we employ a random split of 35 training, 5 validation, and 5 test documents.

Baselines and Classification Tasks

In the following comparative analysis, we benchmark our PointerBERT model variants against the various pre-trained baselines described in Section 8.4.1. In contrast to the other baselines,

Table 8.4: Selected hyperparameters per dataset for our PointerBERT models based on the best validation-set micro F_1 and MAP@3 performances.

Hyperparameter	CSAbstruct	PubMed-RCT	Nicta	$IFRS_{EN}$	GRI_{DE}
Dropout	0.1	0.2	0.2	0.2	0.2
Batch size	8	4	4	4	4
Learning rate	1×10^{-4}	5×10^{-5}	5×10^{-5}	5×10^{-5}	1×10^{-5}
Epochs	2	2	3	30	3
Loss weighting	✗	✗	✗	✓	✓
Random oversampling	✗	✗	✗	✗	✓
Dynamic sampling	✗	✗	✗	✓	✓
Classification head	RNN	Linear	Linear	Linear	Linear
Label embedding dim	32	–	–	–	–

$BERT_{wiki\text{-}en}$ processes each text segment individually, and we utilize the hidden state vector of its special classifier token [CLS] as input for the downstream classification head. The remaining RoBERTa-based models handle multiple segments at a time, which is why we leverage the hidden states of their segment separating special tokens [SEP] for subsequent category prediction. The final output layer of each model differs depending on the dataset and its respective classification task. The scientific abstract datasets require multi-class classification which implies a softmax output layer that uniquely maps each segment s_i to a single category $r_j \in \mathcal{R}$. Conversely, the financial report datasets necessitate a sigmoidal output layer for multi-label classification, where a segment s_i receives relevance scores for all checklist requirements in $\mathcal{R}$.

We also compare our methodology with external state-of-the-art models that have not been pre-trained by us. For the scientific abstract datasets, we draw comparisons with [162] who utilize a pre-trained SciBERT [171] model and fine-tune it in the fashion of our RoBERTa baselines using the model's [SEP] tokens as input for the classification layer. Additionally, we report results for the latest sequential sentence classification models [163–166] that are particularly optimized for incorporating sequential label dependencies while decoding. Lastly, we compare our novel methodology with the dedicated recommender model for sustainability reports, sustain.AI [29], that we introduced previously in Chapter 7. It employs weighted random sampling and leverages a pre-trained BERT architecture equipped with an MLP to identify the most relevant checklist requirements for each text segment in the GRI_{DE} dataset.

Training Setup

For all models and datasets, we conduct an exhaustive grid search across a wide range of hyperparameters to identify the best parameter combinations, evaluated on the hold-out validation set micro F_1 (scientific abstracts) and Mean Average Precision (MAP) @3 scores (financial documents). Initially, we perform a broad parameter sweep to establish a viable starting point for each architecture and dataset, followed by a more detailed fine-tuning within the proximity of these initial parameters. The outcome of this rigorous process is detailed in Table 8.4, presenting the optimal configurations for our PointerBERT model across datasets.

In the following, we highlight a few insights from Table 8.4. First, we compare different

classification heads for the single-label prediction tasks. Besides a standard linear output layer that classifies each segment simultaneously, we evaluate the performance of employing a GRU [83] that incorporates the previously predicted label information for the subsequent segment prediction (see [30] for more details). Surprisingly, this more elaborate decoding method only slightly improves results for CSAbstruct, which indicates that label dependencies are already sufficiently encoded in the separator token hidden states.

Second, we mitigate the challenges of annotation scarcity and data imbalance in the financial report datasets, by utilizing class-based loss weighting and adjusting the binary cross-entropy loss according to the segments' inverse class frequencies. Following [29], we also adopt random oversampling for the GRI_{DE} dataset, enhancing model exposure to annotated segments.

Third, we employ dynamic segment sampling (Section 8.3.2), which increases sample diversity across epochs for models that are capable of processing multiple segments together. This sampling technique proves especially useful for the financial report datasets characterized by a small number of long documents. For our experiments, we set the minimum number of randomly selected segments per sample to $L_{\text{min}} = 3$. We refrain from applying dynamic sampling on the scientific abstract datasets because of their substantially larger size and almost all documents comfortably fitting within our models' 512-token context window.

All additional training parameters not specified in Table 8.4 align with the pre-training configurations described in Section 8.4.1, except for gradient accumulation, which is not needed during fine-tuning due to smaller batch sizes. Also, to ensure a level playing field, all pre-trained baseline models undergo the same hyperparameter selection process and benefit from the described training enhancements, which enables fair test-set evaluations.

Results

Table 8.5 presents the test-set performance metrics for all evaluated models across each dataset. To ensure reliability of our results, each model-dataset pairing has been evaluated 10 times using different seeds, with the average and standard deviation of these runs reported. We categorize the comparisons into three distinct groups. In the first category, our continuously pre-trained PointerBERT models are compared with current state-of-the-art models, revealing that the English PointerSciBERT model surpasses previous benchmarks on two out of three scientific abstract datasets. Notably, we did not focus on incorporating improved decoding mechanisms like advanced attention and CRF output layers, as included in the external baselines. Joining these methods with our PointerBERT methodology would likely further improve results. Additionally, the German $\text{PointerBERT}_{\text{all-de}}$ model exhibits enhanced retrieval performance on the German GRI_{DE} sustainability report dataset. Due to the distinct vocabularies and monolingual pre-training approach, English models were not evaluated on German datasets and vice versa.

The second comparison group evaluates the PointerBERT architecture against RoBERTa and BERT models in a controlled setting. Both sets of models have been identically pre-trained from scratch, allowing any performance discrepancies to be attributed to architectural differences. This comparison underscores the superiority of our pointer-guided SO task, with PointerBERT consistently outperforming the other architectures across all English datasets.

In the final comparison group, the multilingual $\text{PointerBERT}_{\text{all}}$ model is pitted against the $\text{RoBERTa}_{\text{all}}$ baseline. Both models have been pre-trained from scratch as well using the

Table 8.5: Test set results for sequential text classification on scientific abstract and financial document datasets. PointerBERT outperforms all competing baselines in micro and macro F_1 score as well as top 3/5 Mean Average Precision (MAP). We report mean (best scores in bold) and standard deviation values from 10 independently seeded training runs for robust test set evaluation.

	Scientific Abstracts						Financial Reports			
	CSAbstruct		PubMed-RCT		Nicta		$IFRS_{EN}$		GRI_{DE}	
in %	F_1		F_1		F_1		MAP		MAP	
Architecture	Micro	Macro	Micro	Macro	Micro	Macro	@3	@5	@3	@5
Jin et al. [164]	81.30	–	92.60	–	84.70	–	–	–	–	–
Cohan et al. [162]	83.10	–	92.90	–	84.80	–	–	–	–	–
Yama. et al. [166]	–	–	93.10	–	84.40	–	–	–	–	–
Shang et al. [165]	–	–	92.80	–	**86.80**	–	–	–	–	–
Brack et al. [163]	–	–	93.00	–	86.00	–	–	–	–	–
Pointer-$SciBERT_{wiki\text{-}en}$	**83.21** ±0.31	**82.05** ±0.57	**93.56** ±0.10	**89.56** ±0.16	85.07 ±0.33	**76.22** ±1.13	**62.75** ±1.00	**63.99** ±0.94	–	–
Hillebrand et al. [29]	–	–	–	–	–	–	–	–	33.37 ±0.95	35.29 ±0.91
Pointer-$BERT_{all\text{-}de}$	–	–	–	–	–	–	–	–	**34.25** ±1.07	**36.19** ±1.06
$BERT_{wiki\text{-}en}$	73.25 ±0.72	73.83 ±0.60	85.98 ±0.07	80.72 ±0.09	72.74 ±0.48	65.80 ±0.82	56.77 ±0.71	57.61 ±0.67	–	–
$RoBERTa_{wiki\text{-}en}$	81.21 ±0.70	79.37 ±1.02	92.48 ±0.05	88.17 ±0.10	80.98 ±0.56	69.92 ±1.28	54.68 ±0.94	56.05 ±0.90	–	–
Pointer-$BERT_{wiki\text{-}en}$	**81.91** ±0.41	**80.42** ±0.66	**92.63** ±0.06	**88.29** ±0.08	**81.25** ±0.27	**70.82** ±0.73	**57.17** ±1.17	**58.43** ±1.11	–	–
$RoBERTa_{all}$	81.60 ±0.46	79.92 ±0.72	92.77 ±0.10	88.53 ±0.12	81.67 ±0.38	70.42 ±0.90	59.07 ±0.72	60.47 ±0.89	27.83 ±1.79	29.88 ±1.91
$PointerBERT_{all}$	**81.82** ±0.71	**80.36** ±0.83	**92.87** ±0.10	**88.65** ±0.18	**81.92** ±0.33	**71.52** ±1.06	**59.50** ±0.94	**60.52** ±0.87	**28.84** ±1.80	**30.99** ±1.75

same datasets, vocabulary, and training configurations (see Table 8.2). Evaluated across all five datasets, the $PointerBERT_{all}$ model consistently outperforms the $RoBERTa_{all}$ baseline, showcasing the efficacy of our pointer-guided architecture.

Overall, our findings demonstrate that the pointer-guided SO methodology, combined with dynamic sampling for efficient fine-tuning, surpasses all competing baselines across a diverse array of datasets and classification tasks.

8.4.3 Limitations

Despite the promising results, our current experiments have a few practical limitations that we plan to improve upon in future work. Firstly, the models are constrained by a context window size of only 512 tokens and employ absolute positional embeddings. Incorporating advanced attention mechanisms [87], along with relative positional embeddings [175], enables our pre-training approach to accommodate longer input sequences during inference. This enhancement will not only increase the complexity and effectiveness of the SO pre-training task but also enable the model to capture more distant paragraph-level context.

Besides scaling up our pre-training methodology in terms of larger model sizes, increased number of training steps, and larger datasets, we also aim to extend our evaluation and fine-tuning efforts to information retrieval and semantic search tasks [44, 176]. Specifically, we seek to assess our methodology's effectiveness in identifying semantically relevant passages from long documents in response to natural language queries. We assume that our method's enhanced understanding of paragraph-level context and its ability to jointly embed subsequent text segments has the potential to improve semantic search and thereby RAG [35].

8.5 Conclusion

We introduce a novel approach to enhance the contextual sensitivity of paragraph-level text representations through a pointer-guided segment ordering (SO) pre-training strategy and dynamic sampling for fine-tuning. Our methodology aims at improving the understanding of document structure and coherence, which is crucial for a wide range of downstream NLP applications, including text classification and information retrieval.

Our pre-training methodology leverages a self-attention-driven pointer network to restore the original sequence of shuffled text segments, thereby requiring the model to develop a deep understanding of narrative flow, coherence, and contextual relationships. This task, combined with masked language modeling, significantly enhances the model's ability to comprehend and represent paragraph-level context. We further establish dynamic sampling during the fine-tuning phase to increase the diversity of training instances across epochs and improve sample efficiency. This sampling technique proves particularly beneficial for small datasets with long documents, as it helps to mitigate overfitting and foster better generalization.

Our experiments demonstrate that models pre-trained with our pointer-guided SO task outperform existing baselines across a variety of datasets and tasks. Notably, our PointerBERT models achieve superior performance on both scientific literature and financial reporting datasets, showcasing the versatility and effectiveness of our approach. Looking ahead, we aim to overcome current limitations by incorporating more sophisticated language model backbones and broadening our evaluation framework to include information retrieval and semantic search tasks.

In conclusion, our work contributes an advancement in representation learning for paragraph-level text, setting a new benchmark for sequential text classification and paving the way for future research in document structure-based language modeling.

CHAPTER 9

Large Language Models for Compliance Verification

In the previous chapter, we introduced a new self-supervised pre-training task, *Pointer-Guided Segment Ordering (SO)*, to enable encoder-only Transformer-based [19] language models like BERT [34] to learn improved paragraph-level representations by considering the surrounding context of preceding and succeeding paragraphs. We demonstrated the effectiveness of this approach in sequential text classification tasks, where the model performs multi-label classification per paragraph given a predefined set of possible classes. While this method showed significant improvements, it relies on statically trained classifiers mapping paragraphs to a fixed set of requirement IDs. This rigidity poses a limitation: if regulatory standards are updated or new requirements are introduced, the model would require retraining with annotated data, which is time-consuming and not scalable.

To address this issue, in this chapter, we shift towards a more flexible and dynamic approach leveraging advanced text matching techniques and LLMs. We introduce *ZeroShotALI*, a methodology that combines BERT-based text matching with an LLM-based re-ranking module utilizing GPT-4 [18]. This approach enables zero-shot text matching between new financial reports and unseen legal requirements without the need for retraining, offering a scalable solution adaptable to evolving regulatory landscapes.

Furthermore, we extend this retrieval capability by integrating a compliance verification module that utilizes zero- and few-shot learning in conjunction with prompting techniques like chain-of-thought [177] and tree-of-thought [178]. This allows us to assess the actual compliance of the retrieved financial document paragraphs with respect to the requirements from accounting standards such as IFRS (International Financial Reporting Standards) and HGB (Handelsgesetzbuch, the German Commercial Code).

We evaluate our approach on two datasets containing regulatory requirements, IFRS and HGB, demonstrating that our method effectively retrieves relevant text segments and accurately verifies compliance. Our experiments highlight the advantages of using advanced LLMs like GPT-4 for re-ranking and compliance assessment, as well as the potential of open-source models for these tasks. Importantly, we also explore the impact of prompt design and framing on model performance, emphasizing the need for tailored prompts to maximize effectiveness.

This chapter is based on the following publications:

- L. Hillebrand, A. Berger, T. Deußer, T. Dilmaghani, M. Khaled, B. Kliem, R. Loitz, M. Pielka, D. Leonhard, C. Bauckhage, et al., "Improving Zero-Shot Text Matching for Financial Auditing with Large Language Models," *Proc. DocEng*, 2023, DOI: 10.1145/3573128.3609344 [39],
- A. Berger, L. Hillebrand, D. Leonhard, T. Deußer, T. B. F. De Oliveira, T. Dilmaghani, M. Khaled, B. Kliem, R. Loitz, C. Bauckhage, et al., "Towards automated regulatory compliance verification in financial auditing with large language models," *Proc. BigData*, 2023, DOI: 10.1109/BigData59044.2023.10386518 [33].

Lars Hillebrand played a pivotal role in the development of both research papers. He was responsible for conceptualizing the core ideas and designing the experimental framework, including the evaluation setup. The implementation, carried out in `Python`, was a collaborative effort between the shared first authors, Lars Hillebrand and Armin Berger. Armin Berger took the lead in conducting and evaluating the experiments, while Lars Hillebrand focused on prompt design, data collection, processing, and the evaluation routine.

The writing process was a joint endeavor. Armin Berger drafted the initial manuscripts, while Lars Hillebrand made significant contributions, particularly to the methodology sections and the architectural pipeline, including its figures, and thoroughly revised the entire text. Other co-authors supported the writing by contributing to the Related Work section. Both first authors, along with other contributors, engaged in productive discussions and provided valuable feedback throughout the research and writing phases.

9.1 Introduction

Corporate financial disclosures in the form of financial statements play a vital role in informing the public about a company's financial situation and future prospects. These documents provide detailed information on the financial stability, productivity, and profitability of a company, thus having a major influence on investment decisions made by external investors. Due to their economic significance, these documents are highly regulated and examined annually to ensure conformity with relevant financial reporting frameworks, such as the IFRS and Germany's HGB. This examination process requires high-grade expert knowledge and manual analysis of lengthy financial texts, making it inherently time-consuming and prone to human error.

The regulatory requirements of IFRS and other accounting standards are generally presented as a large collection of individual checklist items. For each item, the assigned auditor has to identify the relevant text segments in the financial report before answering questions regarding completeness, accuracy, valuation, consistency, classification, and readability. While the initial retrieval task is particularly tedious, considering the report size and the number of items in the standard, the subsequent assessment of compliance is even more challenging since it requires a deep understanding of the legal requirements and the financial report's content.

In this chapter, we tackle both of these challenges. We introduce a novel method called ZeroShotALI, which enables zero-shot text matching between new financial reports and unseen legal requirements based on a pre-trained SentenceBERT [44] model from [179] and GPT-4 [18]. We evaluate multiple strong baselines, such as combining a vector store-based architecture utilizing OpenAI's Ada embeddings with GPT-4.

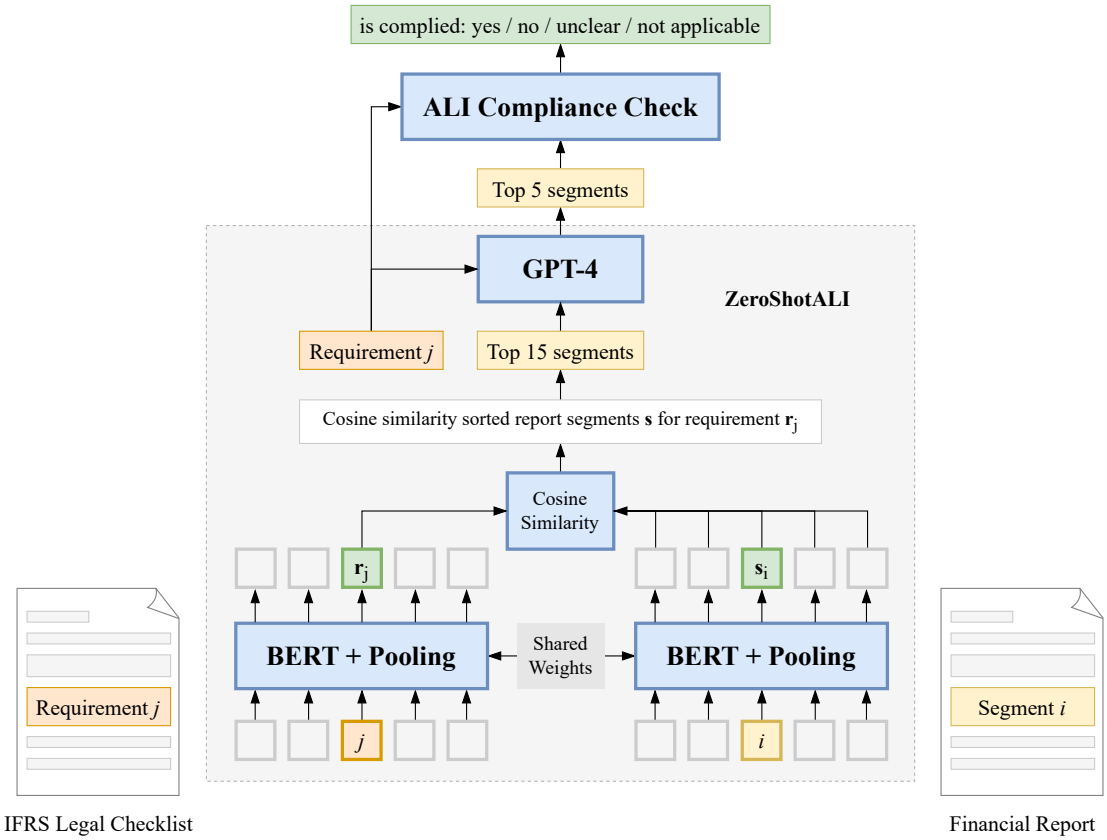


Figure 9.1: Schematic visualization of the complete auditing pipeline combining ZeroShotALI, an auditing-specific textual recommender system, and the ALI compliance check system. While ZeroShotALI focuses on retrieving the top 5 relevant text passages per legal requirement, the compliance check system evaluates whether the retrieved passages comply with the provided requirement.

We leverage ZeroShotALI to find the most relevant passages for each checklist item and subsequently couple it with an LLM like GPT-4 to assess the completeness and compliance of the respective item. Our primary objectives encompass two key aspects: firstly, to evaluate the performance of open-source models in comparison to prominent proprietary models like GPT-4; and secondly, to analyze the impact of framing the problem through the utilization of prompts. Our motivation for exploring open-source models primarily stems from considerations related to cost-effectiveness and data privacy, both of which are pivotal concerns in the contemporary landscape of accounting and ML research.

In the following, we first review related work, before describing our modeling approach in Section 9.3. In Section 9.4, we outline our datasets, present our experiments, and discuss the results. Section 9.5 then draws a conclusion and provides an outlook into conceivable future work.

9.2 Related Work

This section reviews the relevant literature in three key areas: the use of NLP in the financial domain, the application of OpenAI's GPT models and other LLMs for financial tasks, and research on compliance and completeness checks using NLP.

The field of financial NLP, particularly automated auditing, has garnered significant interest over the years. In 2019, we introduced the ALI tool [27], a supervised recommender system that ranks textual components of financial documents according to established regulatory frameworks like IFRS. Traditional NLP techniques such as TF-IDF, latent semantic indexing, neural networks, and logistic regression were employed, with the combination of the first and last methods yielding the best performance.

Subsequent enhancements to ALI included the integration of a pre-trained BERT language model [34] to encode text segments [28]. A more general framework for this task was later introduced by [179]. We build upon this work and present a novel recommender system for financial documents by employing a custom BERT-based model in conjunction with an LLM, improving recommendation results and incorporating completeness checks for the recommended text segments.

To address the need for detailed information extraction for automatic consistency checks of financial disclosures, we developed KPI-Check [31] (see Chapter 6), a BERT-based system that utilizes a customized model for named entity and relation extraction [30] (see Chapter 5). This tool automatically identifies and validates semantically equivalent key performance indicators in financial reports. The KPI extraction task was also studied on an English dataset in [180], which was released along with the results.

The GPT line of OpenAI, although relatively novel, has shown promising results in various financial tasks. [181] qualitatively demonstrated GPT-4's effectiveness in sentiment analysis, ESG analysis, corporate culture analysis, and Federal Reserve opinion analysis. Quantitative analysis by [182] revealed that ChatGPT struggled with complex financial advice, necessitating additional professional guidance. [183] proposed the Automated Financial Information Extraction framework using GPT-3.5 and GPT-4, achieving significant accuracy improvements in KPI extraction from financial reports. [184] investigated the general capabilities of GPT-3.5 and GPT-4 in financial analysis using mock exam questions from the Chartered Financial Analyst (CFA) program.

Other notable works include [185], which introduced BloombergGPT, a model trained on Bloomberg's extensive data sources and evaluated on financial tasks and general LLM benchmarks. FinBERT, introduced by [186] and further researched by [187–190], is another LLM specialized in financial language.

Ensuring compliance and completeness in financial reports, including the detection of contradictory statements, is a critical area of research. We studied the task of automated contradiction detection using a transformer-based model in [32]. Other related works include [191], who implemented a deep MLP for fraudulent statement detection, and [192], who proposed a novel capsule network for detecting fraudulent activities in accounting reports. [98] employed a combined entity and relation extraction method to verify formulas in Chinese financial documents. Additionally, [193] demonstrated the vulnerability of GPT-3 to adversarial attacks in financial sentiment analysis, highlighting the importance of robust, context-aware LLMs.

Our research uniquely integrates a proven recommender system with an LLM to improve

retrieval results and comprehensively verify the completeness of financial reports across varied reporting standards. This approach aims to enhance the reliability and accuracy of automated financial auditing systems.

9.3 Methodology

In this section, we formally define the problem of matching text segments within documents to relevant legal requirements before turning to the in-depth analysis of our proposed architecture, ZeroShotALI, which we couple with an LLM-based compliance check module (see Figure 9.1).

Formally, the task of assigning relevant text segments from a financial report to concrete requirements from an accounting standard can be defined as a text matching problem: For each legal requirement $r_j \in \mathcal{R}$, a list of relevant text segments from the report $s_i \in \mathcal{S}$ (where $\mathcal{S}$ is the set of all paragraphs and $\mathcal{R}$ is the set of all requirements) has to be predicted. The recommendation system has to assign relevance scores $n_{i,j} \in (0, 1)$ to every segment-requirement pair (s_i, r_j). To obtain a classification from those scores, the top K text segments for each requirement are being selected as relevant.

Previous work has mostly approached this task as a multi-label classification problem, where a sigmoidal output layer predicts individual relevance scores for each pre-defined legal requirement. This problem formulation is inflexible with respect to unseen requirements, i.e. a full model re-training is required. Also, the semantic information contained in the actual requirement texts is neglected, since the requirements have been translated to numeric class IDs.

To overcome these architectural shortcomings, we introduce ZeroShotALI. Following [179], we employ a text similarity-based matching model that individually encodes text segments from the financial report as well as legal requirements from the accounting standard before predicting matches based on semantic text similarity. Specifically, we leverage a domain-adapted SentenceBERT model [44] as our initial text retrieval solution. It is a modification of the BERT language model [34] that encodes both the requirement text and the financial report text segment, using two BERT models with shared weights, and applies mean-pooling to obtain paragraph level embeddings $\boldsymbol{s}_i$ for text segment i and $\boldsymbol{r}_j$ for requirement j, respectively. Subsequently, the cosine similarity between these embeddings is computed to measure how well the report segment and legal requirement semantically match, resulting in a normalized similarity score between 0 and 1. For full architectural details and training procedures, we refer to [179]. To further improve the final matching performance, we augment the described SentenceBERT model with the state-of-the-art generative language model, GPT-4. In the first pre-filtering step, SentenceBERT retrieves the top 15 most relevant financial report segments for each requirement query j. Then, we prompt GPT-4 with these 15 segments and the requirement text as input to further narrow down the recommendations to the five best-matching segments.

Subsequently, auditors must ascertain that these suggested passages not only pertain but also conform to the associated legal criterion. To streamline this process and enhance the efficacy of the auditing protocol, we integrate an LLM-based compliance check module to automatically validate the relevancy and compliance of these recommended text sections with their respective legal prerequisites.

The complete architecture is illustrated in Figure 9.1. We evaluate and compare the two-stage approach of ZeroShotALI and its compliance check module with multiple baselines in the next

section.

9.4 Experiments

Our experiments are structured into two main parts. Initially, we focus on the experimental setup and performance evaluation of ZeroShotALI, which is designed to retrieve the top 5 relevant text segments per legal requirement. Subsequently, we introduce the compliance check module to assess the architecture's capability in evaluating the completeness and correctness of the retrieved text segments.

9.4.1 ZeroShotALI

This subsection details our custom financial dataset, describes competing baseline methods for requirement matching, explores various LLM prompt designs, and concludes with a discussion of the results obtained from our approaches.

Data

We obtain 50 IFRS-compliant financial reports from PricewaterhouseCoopers GmbH (PwC), encompassing a total of 7097 text segments. Auditors carefully annotate these reports, mapping each text segment to one of the 1214 distinct IFRS requirements. This rigorous annotation process unfolds through multiple iterations involving various domain experts, ensuring the highest quality of data. The SentenceBERT model, employed in our experiments, undergoes fine-tuning on domain-specific data, with further details available in [179]. Apart from SentenceBERT, the other four architectures we test do not require additional training. Consequently, we exclusively utilize a test set for evaluation, carefully ensuring that the SentenceBERT model is neither trained nor evaluated on this test set to prevent data leakage.

Baselines

We evaluate ZeroShotALI against several baseline models: (1) a basic SentenceBERT model as outlined in Section 9.3, which operates without the GPT-4 filtering enhancement, (2) a vector database retrieval model utilizing OpenAI's Ada V1 or V2 embeddings, and (3) a hybrid approach that combines the same vector database model with GPT-3.5 Turbo or GPT-4 as filtering models.

The vector database employed in our study is Chroma DB, an open-source platform that integrates a k-means clustering algorithm with the ClickHouse database management system to facilitate the retrieval of semantically similar text passages based on a given query. For a comprehensive understanding of the concept and a review of analogous applications, please refer to [194]. We embed each of the 10 IFRS reports using either the lower-dimensional Ada V1 or Ada V2 embeddings and store them in Chroma DB. This setup allows each report to be accessed individually by its name, enabling queries to be made against specific reports. For each query, we retrieve the top five text segments that exhibit the highest cosine similarity scores.

The third set of systems we compare involves Chroma DB using V2 Embeddings, augmented with frozen LLMs, specifically GPT-3.5 Turbo or GPT-4. These systems embrace the concept of retrieval-augmented generation, which aims to equip LLMs with pertinent information for a specific task rather than relying solely on the language model's intrinsic knowledge. As with the previous setup, we retrieve the top 15 most relevant text segments for each query using Chroma DB. These segments then serve as prompt inputs for the LLM, which is tasked with returning the five most pertinent text segments for each requirement.

Effects of prompt design

For all architectures leveraging GPT-4 we evaluate the impact of prompt design on model performance. Prompt design refers to the format and phrasing of the task presented to the GPT-style LLMs. Building upon the findings of [195] regarding the effects of prompt phrasing on LLM performance, we conduct our evaluations by primarily investigating two factors: (1) the phrasing of the task and (2) the structure of the output allowed for the model's responses.

Our evaluation is as follows: We formulate a specific task for GPT-4, which involves the retrieval of the five most relevant text segments from a provided set of 15 segments, based on a given requirement. We then randomly sample 20 requirements from one of the 10 IFRS-compliant reports.

Regarding the phrasing of the task, we find no significant impact on the quality of the model's responses, implying that variations in the way the task was presented did not substantially influence the model's performance.

However, when examining the structure of the output allowed for the model's responses, we observe a notable effect. We categorize the types of outputs into two broad formats: "open-ended" and "closed." In the "open-ended" format, the model is permitted to provide explanations for its answers, while in the "closed" format, the model is restricted to returning the IDs of the five most relevant text segments. Interestingly, we find that the "closed" format yields improved performance compared to the "open-ended" format. Table 9.1 shows the four best-performing prompts and their respective performance.[1] For all GPT-based experiments we leverage the best performing prompt design A.

Evaluation and Results

As depicted in Table 9.2, the performance analysis reveals that the two architectures based on SentenceBERT surpass the vector store-based architectures for all metrics[2]. Notably, ZeroShotALI, combining SentenceBERT with GPT-4, demonstrates the highest performance which can be attributed to several factors.

First, the vector store-based architectures rely on embeddings from generically pre-trained language models that exhibit no domain-specific fine-tuning and leverage approximate nearest-neighbor calculations to return text matches. In contrast, the SentenceBERT model was fine-tuned specifically for the task of retrieving semantically similar text passages within an auditing context. This custom training process enables SentenceBERT to capture the

[1] Due to the stochastic nature of GPT-4 it is impossible to exactly reproduce results. We set the temperature parameter to 0, which reduces but does not remove stochasticity in the generation process.

[2] See Section 7.4.2 for definitions of mean sensitivity and mean average precision.

Table 9.1: Quantitative comparison of 4 different prompt setups for the text segment to requirement matching task. Mean Sensitivity and Mean Average Precision (MAP) are defined in Section 7.4.2. The (bracketed blue) text shows the respective differences between prompts B and A as well as D and C.

Prompt \ in %	Sensitivity	MAP	F_1
A	**36.92**	**26.38**	**23.75**
B	35.54	26.00	23.75
C & D	23.57	22.62	17.05

A & **(B)**: "(System: You are an expert auditor with perfect knowledge of the IFRS accounting standard.) Out of all document segments provided below which ones are the 5 most relevant for fulfilling the IFRS requirement? (Think step by step.) IFRS requirement: {requirement} document segments: {document} Your answer should only contain the IDs of the relevant document segments. Example: ['1129', '1139','1159', '1161', '829']. Your answer needs to be machine-readable. Do not add any additional text."

C & **(D)**: "System: You are an expert auditor with perfect knowledge of the IFRS accounting standard. Out of all document segments provided below which ones are the 5 most relevant for fulfilling the IFRS requirement? Explain for each requirement why you selected the 5 most relevant requirements. (Each should only be a sentence long.) Think step by step: IFRS requirement: {requirement} document segments: {document} Format your output complying to the following json schema: {'explanation': 'The most relevant document segments ...', 'answer': ['1129', '1139','1159', '1161', '829']}. Ensure that 'answer' is its own key in the json schema. Your answer needs to be machine-readable."

Table 9.2: Test set results for the top 5 recommendations of relevant financial report segments for legal requirements of the IFRS accounting standard. ZeroShotALI outperforms all competing methods in Mean Sensitivity, Mean Average Precision (MAP), and F_1 score.

Model \ in %	Sensitivity	MAP	F_1
Chroma (Ada V1)	14.00	7.12	9.12
Chroma (Ada V2)	25.73	17.33	13.15
Chroma (Ada V2) + GPT-3.5 Turbo	29.95	21.32	15.74
Chroma (Ada V2) + GPT-4	35.30	24.72	18.53
SentenceBERT (from [179])	52.12	39.00	27.69
ZeroShotALI (this work)	**57.62**	**44.65**	**30.57**

intricacies and nuances of the auditing domain, leading to more accurate and contextually relevant retrievals. The observed subpar performance of vector store-based architectures in our use cases raises an important question regarding the suitability of such systems in various applications. Currently, many applications leverage retrieval-augmented generation using vector databases, assuming that these systems can provide reliable results. However, our findings suggest that incorporating domain-specific, fine-tuned retrieval systems like SentenceBERT could significantly enhance the performance of such applications.

9.4.2 Compliance Check

Leveraging the strong retrieval capabilities of ZeroShotALI, we couple it with the LLM-based compliance check module introduced in Section 9.3. In the following, we detail its experimental setup and evaluate the results with respect to correctly assessing the completeness and correctness of the retrieved text segments.

Data

In addition to the previously introduced IFRS dataset, our study incorporates 50 HGB-compliant financial reports. The ALI tool, introduced in 2019, serves as a supervised recommender system that ranks text passages in financial documents according to their relevance to auditing standards such as IFRS or HGB [27]. PwC auditors subsequently verify these machine-processed reports to ensure accurate mapping of text segments to the respective accounting requirements under both frameworks. The annotation task, distributed among three auditors and overseen by a senior auditor, involves several rounds of review. This meticulous process includes re-examining randomly chosen annotated samples to fine-tune the generated labels and assess the model's accuracy in identifying false positives and negatives.

Considering the extensive size of the original datasets, we select a subset for detailed analysis. From the IFRS dataset, we sample 10 financial reports, from which we choose 100 specific requirements for evaluation. From the HGB dataset, we select 3 reports, focusing on requirements that have at least two annotations. This selection strategy aligns with our study's aim to assess the capabilities of publicly available LLMs without the necessity for training on domain-specific data, thereby eliminating the need for data splitting.

The annotation criteria for IFRS involve marking the compliance of text passages with a requirement as 'yes', 'no', or 'unclear'. For HGB, annotations include 'yes', 'no', 'unclear', or 'not applicable'. The distribution of these ground truth annotations for both datasets is detailed in Table 9.3.

Baselines

We evaluate six state-of-the-art LLMs, comprising both open-source and proprietary architectures. Specifically, we include three variants of the open-source Llama-2 model[3] [154], with sizes denoted as 7b, 13b, and 70b parameters, and three versions of the closed-source GPT architecture: GPT-3.5-Turbo, GPT-3.5-Turbo-16K, and GPT-4. Our rationale for juxtaposing

[3] The concrete model IDs from Huggingface are: meta-llama/Llama-2-7b-chat-hf, meta-llama/Llama-2-13b-chat-hf, meta-llama/Llama-2-70b-chat-hf.

Table 9.3: Class distribution of ground truth values for IFRS and HGB. HGB compliance data was annotated by auditors at PwC Germany with 'yes', 'no', 'unclear', or 'not applicable', while IFRS data was annotated with 'yes', 'no', and 'unclear'.

Label	IFRS	HGB
Yes	17	43
No	82	28
Unclear	1	23
Not Applicable	-	26
Total	100	120

open-source and proprietary models is twofold: economic considerations and data privacy issues. Additionally, the open-source nature of certain models offers the potential for fine-tuning, facilitating adaptability for specialized applications.

To ensure a controlled environment for model inference, we deploy a dedicated server equipped with an NVIDIA A100 80 GB GPU. We implement an inference API analogous to the OpenAI API to facilitate on-demand access to the Llama-2 models. For experimental consistency, all models are subjected to identical prompts and parameters during the inference phase.

Prompt Design

Since all systems involve the querying of an LLM, we evaluate the impact of prompt design on model performance. The term prompt design encompasses how a task is presented to LLMs. Building on insights by [196] into the impact of prompt phrasing on LLM performance, our evaluation centers on two key factors: (1) task phrasing and (2) the structure of permitted model responses.

In our evaluation methodology, we have devised a specific task for all tested LLMs, involving the assessment of text passages from financial reports against regulatory accounting standards like IFRS or the German HGB. Through a process of trial and error and qualitative assessment, we have selected a total of eight prompts aimed at solving the above-stated task. Prompt performance was then quantified using metrics including Precision, Recall, and F_1-Score per predicted class, as well as Micro and Macro F_1-Score averages across all classes (detailed in the Evaluation Metric section). Below we have added one exemplary prompt.

Exemplary prompt

System: You are an expert auditor with perfect knowledge of the IFRS accounting standard. You always answer truthfully whether a given regulatory requirement is fully complied with in the following line IDs.
Answer with "yes", if all sub-requirements are fully complied. Answer with "no", if at least one of the sub-requirements is not fully complied.
Format your output complying to the following json schema:

```
{{"answer": <"yes"|"no">}}
```

requirement: "`{requirement}`"
document: "`{document}`"

The main two factors we discuss, are (1) task phrasing and (2) the structure of permitted model responses.

In terms of task phrasing, we used a variety of techniques such as chain of thought prompting, providing the model with an example prompt answer combination (also referred to as a one-shot prompt), asking the model for an explanation for their answer, and Tree-of-Thought prompting. Tree of Thought prompting as introduced by [178] and [197] (`https://www.promptingguide.ai/techniques/tot`) refers to the idea of enhancing LLM's ability to solve more elaborate problems through tree search via a multi-round conversation. Since this technique traditionally requires multiple LLM calls, the technique is costly and compute-intensive, thus not scaling well for commercial applications like ours. To overcome this issue, we have employed the prompting method from [198] that adapts key elements from Tree-of-Thought frameworks to create a singular prompt that enables LLMs to assess intermediate ideas.

When examining the response structure, a noteworthy effect was observed. Outputs were categorized into two main formats: "open-ended", allowing the model to provide explanations, and "closed", constraining the model to return only 'yes', 'no', 'unclear', or 'not applicable'. Intriguingly, the "closed" format yielded superior performance compared to the "open-ended" format.

Due to the lengthy nature of each prompt, we summarize the differences between each evaluated prompt. The complete prompt definitions can be found in Section B.2 in Appendix B.

I. In-Out-Sub-Template:

- Simple Yes/No/Unclear/Not applicable answer.
- Short, point-by-point answers without explanation.
- Formatting in JSON schema.

II. Cot-Sub-Template:

- Chain of thought response leveraging intermediate explanations before the final answer.
- Specification of relevant line numbers from the document.
- No formatting in the JSON schema.

III. In-Out-Template:

- Simplified version of In-Out-Sub.
- Direct Yes/No/Unclear/Not applicable response.
- Formatting in JSON Schema.

IV. Cot-Template:

- Step-by-step response and explanation for each sub-request.
- Detailed explanation for each sub-request.
- No formatting in JSON schema.

V. In-Out-Tot-Template:

- Same as In-Out, with the addition of tree-of-thought Prompting.
- Three experts give their opinion on the issue and come to a conclusion through majority voting.

VI. In-Out-Tot-One-Shot-Template:

- Same as In-Out-Tot, with the addition of a one-shot example.
- In the one-shot example chosen, the text passage complies with the requirement.

VII. In-Out-One-Shot-Template:

- Same as In-Out with the addition of a one-shot example.
- In the one-shot example chosen, the text passage complies with the requirement.

VIII. In-Out-One-Shot-No-Template:

- Same as In-Out with the addition of a one-shot example.
- In the one-shot example chosen, the text passage does not comply with the requirement.

Evaluation and Results

This section aims to assess the applicability of current state-of-the-art LLMs in an auditing context, specifically to validate the compliance of financial report passages with regulatory standards. For each model, we execute the same eight selected prompts, with the sole exception being an answer adaptation for the open-source Llama-2 models. If an LLM provides an answer outside the predefined choices, we categorize the response as 'invalid'. We explore three main questions through various configurations:

1. Performance Across Configurations: Among all selected systems, which LLM and prompt configuration performed the best per class and across all classes?
2. Prompt Consistency Across Models: Is prompt performance consistent across models? If yes, which prompts perform the best across all models and datasets?
3. Deploying LLM for Compliance: How can we deploy an LLM-based compliance check system in a manner that saves auditors time while minimizing false negatives?

Performance Across Configurations: Evaluating the overall micro F_1-Score performance across both datasets, HGB and IFRS, the generally best-performing model is GPT-4. As can be seen in Table 9.4, the model achieves a micro F_1 score of 59.31% on HGB and 71.65% on IFRS data averaged across all prompts. A similar picture can be seen in the detailed analysis of comparing

Table 9.4: Micro F_1-Scores per model and dataset (HGB and IFRS) for the best performing prompt and the average over all prompts.

in %	HGB		IFRS	
Model	Average	Best Prompt	Average	Best Prompt
GPT-3.5-Turbo	41.83	46.15	66.68	77.56
GPT-3.5-Turbo-16k	40.48	46.15	49.92	**77.58**
GPT-4	**59.31**	**75.60**	**71.65**	77.05
Llama2-7b	28.49	49.23	47.56	68.02
Llama2-13b	24.81	47.34	48.01	65.58
Llama2-70b	18.04	49.23	53.02	70.04

all prompt configurations for all models and datasets in Table B.1 in Appendix B.1. Overall, we observe significantly worse performance on HGB data than on IFRS data across all models. This is likely attributed to the fact that most state-of-the-art LLMs are almost exclusively trained on an English text corpus, thus neglecting text understanding in other languages. The Llama-2 models for reference were trained on a 98% English text corpus as stated in Meta's technical report [154].

A further notable finding is that the increasing parameter size in the open-source Llama-2 models did not translate into superior performance across all prompt types. Llama-2-70b for reference performs worse overall, considering micro F_1-Score performance across both the HGB and the IFRS dataset, than its significantly smaller counterpart Llama-2-7b.

Prompt Consistency Across Models: Prompt performance does not appear to be consistent across models for our task. Despite the varying performance across models, we notice that prompt I. 'In-Out-Sub-Template' achieves the best score in 4 out of 12 cases, when considering micro F_1-scores (see Table 9.5). These results indicate that a combination of keeping the prompt instructions brief and providing the model with examples to follow can improve the response quality. It is interesting to note that the more advanced prompting techniques such as chain of thought (II. and IV.)or trees of thought (VI. and VII.) did not induce a significant performance increase.

Deploying LLMs for Compliance: In collaboration with our partners at PwC Germany, we have determined that a compliance check system should avoid false positive predictions at all costs. Assuming that a text passage complying with a given auditing context is considered positive, a false negative prediction, in this case, would imply a financial report text passage falsely being classified as complying with a regulatory requirement. To avoid false positives, we prefer a model that has very high precision and recall for the class 'No'. In our evaluations, we found that the best model for the 'No' class in terms of F_1-Score is the open-source Llama-2-70b with a Precision of 80.21%, Recall of 96.25% and an F_1-Score of 87.50% on IFRS data (see Table 9.6). This performance did not translate into a similar performance on HGB data, which is likely due to the German language.

Even though the 'No' class is most important in practice, we also report the models' micro

Table 9.5: Best-performing prompt per model and dataset based on the Micro F_1-Score.

	Prompt	
Model	HGB	IFRS
GPT-3.5-Turbo	I	VI
GPT-3.5-Turbo-16k	I	I
GPT-4	VII	II
Llama-2-7b	II	I
Llama-2-13b	VI	III
Llama-2-70b	VI	III

Table 9.6: Results for Llama-2-70b in % IFRS Data - Class 'No'.

	No		
Prompt	Precision	Recall	F_1
I	75.00	41.25	53.23
II	80.22	91.25	85.38
III	82.22	46.25	59.20
IV	82.86	72.50	77.33
V	81.58	77.50	79.49
VI	80.21	**96.25**	**87.50**
VII	82.35	17.50	28.87
VIII	**85.00**	21.25	34.00

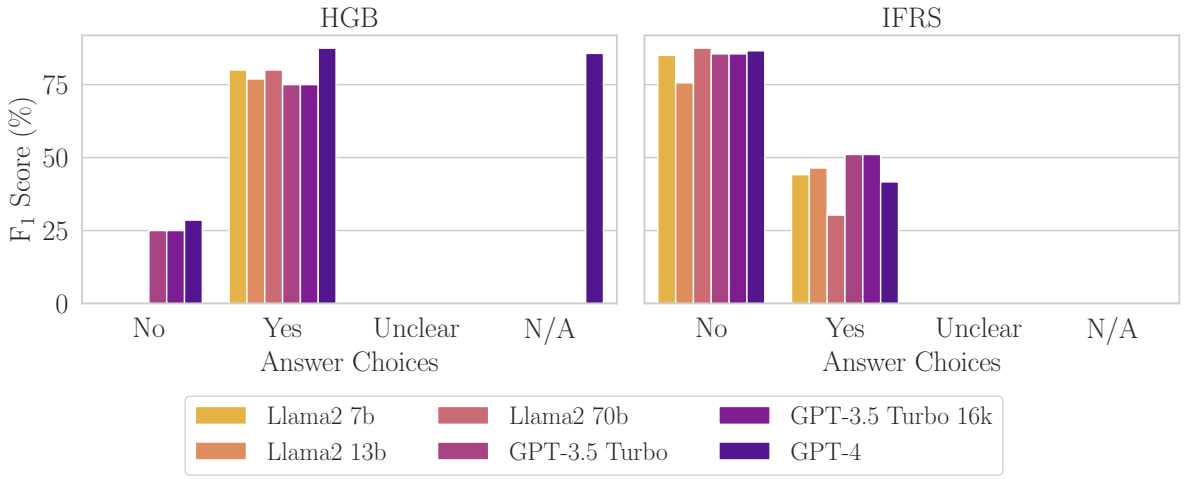


Figure 9.2: Grouped bar plot of F_1-Scores by model and answer choices on HGB and IFRS data. Due to poor model performance in the German language, some LLMs were incapable of generating any machine-readable consistent outputs that are interpretable with a heuristic for some prompt formats, leading to some F_1-Scores being 0.

F_1 scores for the other classes in Figure 9.2 for IFRS and HGB. It can be seen that for both datasets no model picked up on the 'Unclear' class which might be explained by the models being overly confident in their 'Yes' (is complied) and 'No' (is not complied) answers. Also, the figures reveal that surprisingly for HGB the models perform best on the 'Yes' class while performing significantly better on the 'No' class for IFRS.

9.5 Conclusion and Future Work

This study has conducted a thorough evaluation of the application of LLMs in the auditing of financial documents, focusing on two main aspects: retrieval of relevant text segments and

compliance verification against regulatory standards such as IFRS and HGB.

The first part of our experiments centered on ZeroShotALI, a system designed to retrieve the top five relevant text segments per legal requirement using a domain-tuned SentenceBERT model combined with OpenAI's GPT-4. This system significantly outperformed other baseline models, highlighting the effectiveness of domain-specific adaptations over generic systems employing general-purpose embeddings.

Building on the strong retrieval capabilities of ZeroShotALI, we introduced a compliance check module to evaluate the completeness and correctness of the retrieved text segments. This extension leverages the same LLM architecture to assess whether the text segments fully comply with the specified requirements. Our findings indicate that while GPT-4 showed robust performance, especially with the IFRS dataset, challenges remain when dealing with non-English datasets like the German HGB, due to the models' training predominantly on English text corpora.

The study also revealed that there is no universally optimal prompt for LLMs; different models responded best to tailored prompts, emphasizing the need for customizing prompts to maximize the effectiveness of LLMs. Moreover, simpler prompts combined with exemplary instructions often yielded better results than more complex prompting techniques.

In deploying LLMs for compliance checks in financial reporting, precision is crucial to avoid significant repercussions associated with false positives. Our collaboration with PwC Germany highlighted the importance of achieving high precision in detecting non-compliance, with the open-source Llama-2-70b model showing promising performance in this regard on the IFRS dataset.

Future research should explore several avenues:

1. The deployment of domain-specific fine-tuned LLMs for auditing, utilizing open-source models like LLaMA, could potentially overcome current limitations related to language and domain specificity.

2. Advanced prompt tuning methodologies, such as Chain-of-Thought [177] or Tree-of-Thoughts [178], warrant further exploration to optimize LLM performance across different configurations and tasks.

3. The expansion of the ZeroShotALI system to assess requirement completeness based on relevant text passages could further enhance its utility in auditing applications.

Additionally, given the commendable performance of the open-source Llama-2-70b model in accurately predicting true negatives, further investigation into the potential improvements achievable through additional model training on comprehensive accounting compliance data may enhance its effectiveness and enable a more reliable assessment of regulatory compliance. Opting for dedicated open-source models also presents advantages in terms of data privacy and economic feasibility, particularly when compared to the costs associated with commercial LLM APIs.

In conclusion, while LLMs hold immense promise for transforming the auditing landscape, their deployment must be approached judiciously, with careful model selection, prompt customization, and an acknowledgment of their inherent limitations to ensure their effective integration into auditing workflows.

CHAPTER 10

Retrieval-Augmented Generation for Risk and Quality Assurance

In the previous chapter, we explored the application of LLMs for compliance verification in financial reporting, introducing ZeroShotALI, a flexible approach combining BERT-based text matching with GPT-4 for re-ranking and compliance assessment. While this method effectively aided in verifying compliance with accounting standards, organizations operating in highly regulated industries such as auditing, finance, and legal services face additional challenges. Compliance with Risk Management and Quality (R&Q) standards is not only crucial but also complex, as non-compliance can lead to significant legal penalties and financial losses. Employees must navigate intricate regulations and policies, often dealing with numerous internal queries that require nuanced interpretation of trusted sources.

To address these challenges, we introduce a specialized chatbot powered by GPT with an optimized Retrieval-Augmented Generation (RAG) pipeline. By integrating hybrid search techniques and relevance boosting, our system enhances retrieval accuracy and improves response quality, providing precise and contextually appropriate answers to users' queries related to R&Q standards. This chatbot assists employees in efficiently accessing and interpreting complex regulatory information, thereby supporting compliance efforts within the organization.

Our work is evaluated using a handcrafted dataset with expert-annotated answers, and we develop a custom evaluation framework to assess the chatbot's performance. The chatbot has been successfully deployed within the Risk and Quality department of PricewaterhouseCoopers GmbH, enhancing the efficiency and accuracy of handling internal queries.

In summary, the key contributions of this chapter include the development of a RAG-based chatbot tailored for R&Q compliance, the creation of a robust evaluation framework aligned with expert assessments, and insights into optimizing system performance through careful hyperparameter tuning, valuable for practitioners.

This chapter is based on the following publication:

- L. Hillebrand, A. Berger, D. Uedelhoven, D. Berghaus, U. Warning, T. Dilmaghani, B. Kliem, T. Schmid, R. Loitz, and R. Sifa, "Advancing Risk and Quality Assurance: A RAG Chatbot for Improved Regulatory Compliance," *Proc. BigData*, 2024, DOI: 10.1109/BigData62323.2024.10825431 [40].

Lars Hillebrand played a key part in the development of this research. He was responsible for

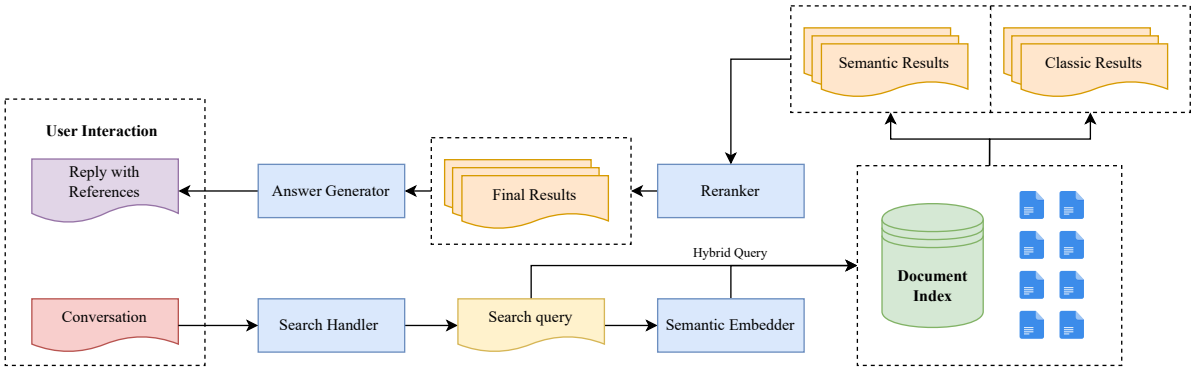


Figure 10.1: Architecture of the Retrieval-Augmented Generation chatbot system, demonstrating the workflow for query resolution.

formulating the foundational concepts and crafting the experimental design, including the setup for evaluation. The implementation, executed in `Python`, was a collaborative effort between Lars Hillebrand and Armin Berger, who share first authorship of the publication. Lars Hillebrand took the lead in conducting and analyzing the experiments, while Armin Berger concentrated on gathering and processing the data, as well as managing parts of the evaluation process.

The writing process was a collaborative effort. Lars Hillebrand made substantial contributions, particularly in detailing the methodology and designing the architectural framework, including the development of illustrative figures. He also meticulously revised the entire document to enhance clarity and coherence. Other co-authors contributed by focusing on the Related Work section. Throughout the research and writing stages, both first authors, along with the team, engaged in constructive discussions and provided valuable feedback and insights.

10.1 Introduction

Compliance with Risk Management & Quality Standards is fundamental in regulated industries like auditing, finance, and legal services, where non-compliance can lead to significant legal penalties and financial losses. Employees face the challenge of navigating complex regulations and policies, handling numerous internal queries daily that require nuanced interpretation of trusted sources.

Recent advancements in LLMs such as GPT [18, 91] have shown significant potential in assisting with complex language understanding tasks. In the legal domain, LLMs are increasingly being utilized to support various aspects of legal work [199]. However, directly applying LLMs in regulated contexts poses challenges, particularly regarding the accuracy and reliability of generated responses.

In this work, we propose a specialized chatbot that leverages an optimized RAG pipeline to address the specific needs of R&Q compliance assistance. By integrating hybrid search methods and relevance boosting, our system (see Figure 10.1) enhances the retrieval of relevant information from a trusted corpus of regulatory documents. The RAG approach allows the LLM to generate responses grounded in these documents, ensuring that the answers are accurate, contextually appropriate, and compliant with organizational policies and regulations. Here is

an example of how the system processes a user query and generates a response.

Example Query and R&Q chatbot Response

Employee query:
"How far does the prohibition on the provision of tax advisory services extend in the case of EU PIEs?"

Workflow:
```
Preparing and embedding query
Running retrieval using hybrid search
Re-ranking text- and vector search results
Generating final answer with references
```

Final answer:
"To answer the question of how far the prohibition on providing tax advisory services to EU Public Interest Entities (PIEs) extends, we need to analyze the relevant information from the provided documents. [...] In summary, the prohibition on providing tax advisory services to EU-PIEs in Germany is very extensive and encompasses almost all tax advisory services listed on the blacklist. However, there are specific exceptions and national regulations that may allow certain tax advisory services under certain conditions."

We evaluate our solution using a handcrafted dataset with expert-annotated answers and develop a custom evaluation framework using `DeepEval`[1]. The framework achieves a 0.70 correlation coefficient with expert assessments. The chatbot has been successfully deployed within the R&Q department of PricewaterhouseCoopers GmbH. The key contributions of this work are:

- **Development of a RAG chatbot for R&Q standards:** We introduce a specialized chatbot combining advanced AI capabilities with RAG.
- **Establishment of a Robust Evaluation Framework:** We devise an automated chatbot evaluation method corroborated by expert assessments.
- **Insights into Hyperparameter Optimization:** We identify how core hyperparameters affect system performance.

In the following sections, we first review related work. We then describe our modeling approach in Section 10.3. In Section 10.4, we outline our datasets, present our experiments, and discuss the results. Finally, Section 10.5 draws a conclusion and provides an outlook into future work.

10.2 Related Work

The advancement of LLMs, such as GPT-3 and GPT-4 [18, 91], has significantly transformed NLP, enabling applications across diverse domains, including legal and regulatory compliance. These models exhibit remarkable capabilities in understanding and generating human-like text, which is crucial for automating complex tasks in compliance and risk management. The evaluation of these models' capabilities has also advanced, with [200] demonstrating that LLMs

[1] `https://github.com/confident-ai/deepeval`.

themselves can serve as effective judges for assessing model performance, achieving over 80% agreement with human evaluators and providing a scalable framework for evaluating both model capabilities and human alignment.

RAG techniques [35] enhance LLMs by integrating external knowledge sources, allowing for more accurate and contextually relevant responses. Efficient retrieval methods, including semantic search [201] and dense passage retrieval [202], are essential for fetching pertinent information in knowledge-intensive domains.

In the legal industry, LLMs have raised attention after GPT-4 [18] was able to achieve impressive legal question answering results, such as passing first-year law school exams [203] and passing the bar exam [204]. Moreover, ML techniques have been applied to tasks like contract analysis [205] and legal assistance in French [206]. The integration of AI in compliance processes has involved combining text analysis with structured data, as seen in consistency checks for verifying financial documents [31, 98], and contradiction detection in financial texts using transformer-based models [32].

10.3 Methodology

In this section, we present the methodology underlying our proposed system designed to enhance compliance with Risk Management & Quality standards through AI-driven solutions. The system architecture comprises three primary components: (1) an ingestion pipeline that processes and indexes documents into a structured knowledge base; (2) a RAG chatbot leveraging LLMs to provide accurate and contextually appropriate responses; and (3) an automated evaluation framework for quantitatively assessing the system's performance.

10.3.1 Ingestion Pipeline and Knowledge Base Construction

Effective knowledge retrieval necessitates a robust and well-structured knowledge base. To this end, we developed an ingestion pipeline that systematically transforms raw documents into a format suitable for AI-driven search and retrieval. The pipeline consists of several stages: parsing, processing, chunking, embedding, and indexing, as depicted in Figure 10.2.

Parsing and Processing

We employ the `Unstructured`[2] library to parse various document formats, including PDFs, Word documents, and HTML files. The library extracts fundamental text elements such as titles, paragraphs, tables, and lists while preserving the semantic and structural integrity of the original documents. Hierarchical relationships, such as headings and subheadings, are maintained to reflect document organization accurately. The extracted elements are converted into a standardized Markdown format, facilitating uniform handling in subsequent processing stages and ensuring compatibility with downstream NLP tasks.

Specific elements with limited relevance or redundant information, such as headers and footers, are removed to enhance the quality of the extracted text.

[2] `https://github.com/Unstructured-IO/unstructured`

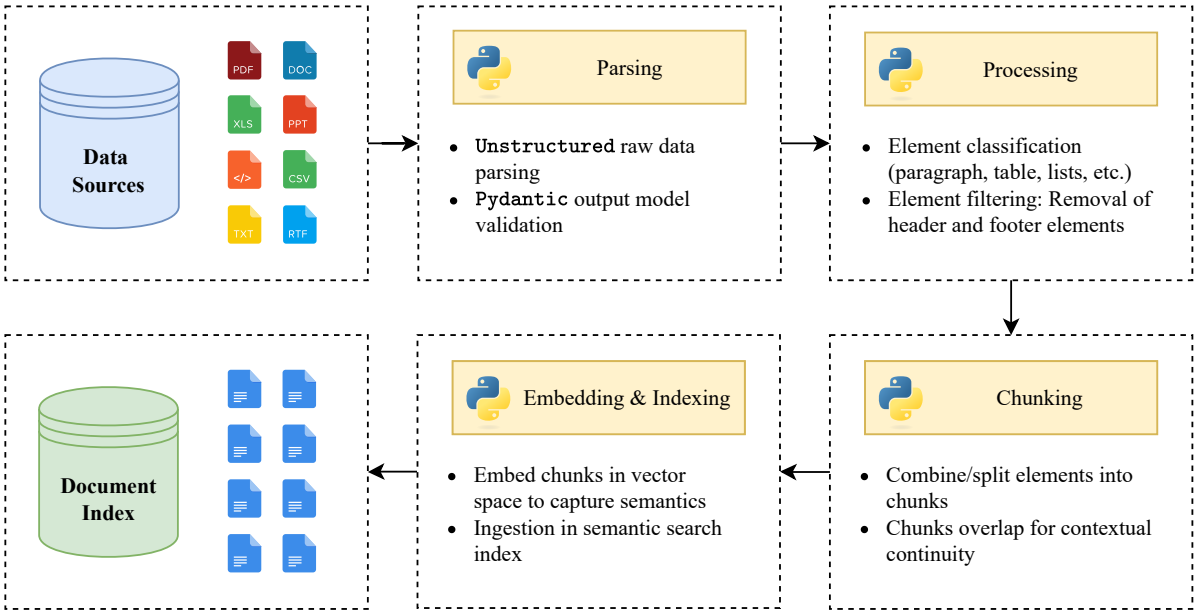


Figure 10.2: Schematic visualization of the document ingestion pipeline architecture, illustrating the individual steps from raw document parsing from various formats to indexed knowledge base creation.

Chunking

To optimize documents for embedding and retrieval, we segment the processed text into coherent chunks. Our chunking strategy involves combining smaller text elements sequentially until reaching a predefined maximum token length (e.g., 500 tokens), introducing overlaps of 50 tokens between chunks to maintain context continuity. For larger elements exceeding the maximum size, such as extensive tables or long paragraphs, we implement sentence-level splitting using natural language sentence tokenizers. This approach balances the need for context preservation with computational efficiency in embedding and retrieval processes.

Embedding and Indexing

We generate embeddings for each chunk using OpenAI's `ada-002` [207] and `3-large` [208] embedding models. These embeddings capture semantic representations of the chunks, enabling effective similarity searches. The embedded chunks are indexed using Azure AI Search [3], supporting hybrid search capabilities that combine vector similarity with keyword matching. We employ boosting strategies within the indexing process to prioritize internal documents ($2\times$ boosting factor) over external sources, ensuring that authoritative organizational knowledge is prominently featured in search results.

[3] `https://learn.microsoft.com/en-us/azure/search/search-what-is-azure-search`

10.3.2 Retrieval-Augmented Generation Chatbot

Our RAG chatbot system is designed to effectively interpret user queries, retrieve relevant information from a knowledge base, and generate contextually appropriate responses. The system employs a sophisticated hybrid search strategy that enhances retrieval accuracy and relevance.

The hybrid search strategy combines vector similarity search with fulltext search using TF-IDF-based BM25 [209] algorithms. Vector similarity search leverages dense vector representations of text, capturing semantic meanings beyond mere keyword matching. Analogously to the ingestion stage, we utilize OpenAI's `ada-002` [207] and `3-large` [208] embedding models to convert the user queries into high-dimensional vectors. The similarity between the embedded query and the ingested document chunk vectors is calculated using cosine similarity, allowing the system to identify chunks that are semantically aligned with the user's intent.

In parallel, the full-text search component employs the BM25 algorithm, which ranks chunks based on TF-IDF. BM25 is particularly effective for capturing the importance of exact term matches, providing a complementary perspective to the semantic insights from vector similarity search.

To integrate the results from both search methods, we apply Reciprocal Rank Fusion (RRF)-[210], which combines the rankings by assigning higher importance to documents that are highly ranked by either method. The RRF score for a retrieved document chunk is calculated as

$$\mathrm{RRF}_d = \sum_{k=1}^{K} \frac{1}{\gamma + r_d^k}, \tag{10.1}$$

where r_d^k is the rank of chunk d in the k-th retrieval method (in our case $K = 2$), and $\gamma = 60$ is a constant to relatively increasing the impact of lower-ranked chunks compared to highly ranked chunks.

By employing this hybrid search strategy and coupling it with the previously described relevance boosting, our RAG chatbot system ensures that responses are generated based on the most trusted and contextually relevant information, providing users with accurate and helpful answers.

10.3.3 Automated Evaluation Framework

We establish an automated evaluation framework using `DeepEval`[4] and the G-Eval scoring method [211]. The evaluation focuses on correctness, completeness, relevance, and adherence to R&Q standards. We leverage GPT-4o as the LLM backbone for the evaluation and define the metric range between 0 (worst) and 5 (best). The following evaluation steps are performed to create the final score per sample.

[4] `https://github.com/confident-ai/deepeval`.

Evaluation Steps

Answer Evaluation Steps

- Check if the facts in 'Actual Output' contradict any facts in 'Expected Output'.
- DO NOT punish long and detailed answers if the 'Actual Output' is perfectly correct. Generally, more details in the 'Actual Output' are encouraged.
- If the 'Actual Output' misses details compared to the 'Expected Output' you should slightly penalize omission of detail.

Context Evaluation Steps

- Summarize the expected 'Context' and note the most important points.
- Compare the summary with the 'Retrieval Context' and check if the most important points are present.
- If the 'Retrieval Context' is missing important points compared to the 'Context' you should penalize the response.
- If the 'Retrieval Context' contains irrelevant information, you should very slightly penalize the response.
- If the 'Retrieval Context' contains contradictory information, you should heavily penalize the response.

To validate reliability, we compare LLM-based scores with manual evaluations from domain experts across 124 responses, achieving a Pearson correlation coefficient of $r = 0.70$. While acknowledging potential LLM biases [200], this correlation supports the use of automated evaluations as proxies for expert judgment.

10.4 Experiments

We conduct experiments on an expert-curated dataset to provide insights for implementing LLM-based chatbots in production environments. We present our dataset, experimental setup, and discuss our findings.

10.4.1 Data

Our dataset [5] consists of 124 Risk and Quality question-answer pairs created by PwC Germany domain experts. An illustrative example is highlighted in Figure **??**. For each question, experts identified relevant sources from both internal company data and external resources, utilizing these in conjunction with their professional expertise to formulate answers. Of these questions, 110 were based on internal sources, while the remainder drew from external data. The task of creating and answering these questions was distributed evenly among 13 experts, with oversight provided by three senior Risk and Quality assurance specialists. The question-answer pairs underwent multiple rounds of review for verification and refinement.

10.4.2 Model Configurations

Our ablation studies examine three key areas: (1) ingestion parameters, (2) retrieval parameters, and (3) model parameters, measuring their impact on system performance. Table 10.1 presents the complete configuration space, with bold values indicating our baseline setup. All configurations use an LLM temperature value of 0 to increase answer robustness. Through a systematic

[5] Dataset and `Python` code are currently unpublishable due to ongoing industrial project constraints

Table 10.1: Hyperparameter configurations. Bold values indicate the Baseline setup used for ablation studies.

Module	Hyperparameter	Configurations
Ingestion	Max Chunk Size	256, **512**, 1024, 2048
	Min Chunk Overlap	32, **64**, 128, 256
	Markdown Conversion	Yes, **No**
Search	Top-k	5, **10**, 20
	Search Type	Text, **Hybrid**, Vector
	Relevance boosting	Yes, **No**
	Embedding model	ada-002, 3-large
Chatbot	LLM-Backbone (GPT)	4o-mini, **4o**, 3.5-Turbo, 4-Turbo

evaluation of ingestion and retrieval parameters, we identify the optimal configuration achieving the highest correctness scores. While initially using `ada-002` for embeddings, we discovered that `3-large` yields superior performance during our retrieval optimization process, leading to its adoption in subsequent experiments. The final optimized configuration is then used as the foundation to assess different LLM backbones (see Table 10.3).

10.4.3 Prompt Design

Our prompt design includes a template that ensures consistent and accurate responses from the LLM. We utilize dynamic language detection using the `langdetect`[6] library to automatically adjust the language of the response to match the user's query. The prompt instructs the model to cite sources appropriately and avoid hallucinations by stating when information is not present in the provided context.

[6] `https://github.com/Mimino666/langdetect`.

Prompt Template

{user_question}

<instruction>
Write your answer in {language}. If you cannot answer the question based on the provided context, state that the information is not present, don't invent or hallucinate an answer, and don't reference any sources. After each fact you state, provide the corresponding document name and chunk ID from the appended sources in brackets and separated by "/". For example: "Apple was founded in 1976." ∗(apple.docx/1)∗
Don't combine sources but list each individual source separately if a fact contains multiple sources. E.g. ∗(apple.docx/1)∗, ∗(apple.docx/2)∗, etc.
You must comply with the following sources format: ∗(<document_name_as_str>/<chunk_id_as_int>)∗
Before answering the question, lay out your full thought process and dissect the user question and its implications.
</instruction>

<document_context>
{retrieved_chunks}
</document_context>

10.4.4 Results

Our experiments demonstrate that the optimal configuration, $\text{Baseline}_{\text{3-large}}$ with relevance boosting enabled, achieves the highest correctness scores for both answers and context, as shown in Table 10.2. We see that our hybrid search strategy outperforms both, full-text and semantic vector search. Additionally, we find that a chunk size of 512 tokens with an overlap of 64 tokens strikes an optimal balance between providing sufficient context and maintaining processing efficiency. Configurations with notably smaller or larger chunk sizes and overlaps lead to performance degradation, emphasizing the importance of these parameters in preserving search and subsequently answer quality.

In the second part of our evaluation, we assess the impact of different LLM backbones using the optimal retrieval configuration identified earlier. As presented in Table 10.3, although the previously evaluated ingestion and retrieval parameters contribute to the overall answer quality, the choice of LLM has the most significant effect. GPT-4o outperforms the other models, achieving the highest scores in both answer and context correctness. In stark contrast, GPT-3.5-Turbo scores considerably lower, despite all other parameters remaining unchanged. This pronounced difference illustrates that the selection of the LLM backbone is critical to maximizing the effectiveness of the RAG chatbot system. For robust analysis, each model configuration was evaluated using 5 independent runs, with results reported as mean and standard deviation for our G-Eval correctness metric.

10.5 Conclusion and Future Work

In this work, we introduced a novel RAG chatbot system tailored for R&Q assurance in highly regulated industries. Our system effectively leverages LLMs with optimized retrieval strategies, including hybrid search and relevance boosting, to improve query processing and compliance adherence. The evaluation demonstrates significant performance gains over baseline approaches,

Table 10.2: Detailed ablation study to evaluate multiple model configurations. We report mean and standard deviation values of 5 independent runs (best scores in bold) for both, answer and context correctness (Scale: 0-5).

Model Configuration	**G-Eval Correctness Score**	
	Answer ↑	Context ↑
$\text{Baseline}_{\text{ada-002}}$	**3.72** (±.026)	2.80 (±.028)
Chunking: 256/64	3.61 (±.048)	2.75 (±.041)
Chunking: 512/32	3.69 (±.042)	2.84 (±.043)
Chunking: 512/128	3.69 (±.053)	**2.88** (±.046)
Chunking: 1024/128	3.67 (±.045)	2.74 (±.049)
Chunking: 1024/256	3.66 (±.039)	2.83 (±.045)
Chunking: 2048/256	3.56 (±.050)	2.29 (±.015)
+Markdown	3.65 (±.050)	2.77 (±.030)
$\text{Baseline}_{\text{3-large}}$	3.76 (±.030)	**2.91** (±.031)
Vector Search	3.72 (±.030)	**2.91** (±.032)
Text Search	3.60 (±.030)	2.62 (±.026)
Top-k: 5	3.72 (±.033)	2.77 (±.027)
Top-k: 20	3.72 (±.016)	2.90 (±.048)
+Relevance Boosting	**3.79** (±.037)	2.90 (±.018)

Chunking: 512/64 (Max Chunk Size = 512, Min Chunk Overlap = 64)

Table 10.3: Results of the best architectural setup for different LLM backbones (Scale: 0-5 and best scores in bold).

Model Configuration	**G-Eval Correctness Score**	
	Answer ↑	Context ↑
GPT-4o (R&Q-Chatbot)	**3.79** (±.037)	**2.90** (±.018)
GPT-4-Turbo	3.69 (±.047)	2.84 (±.048)
GPT-4o-mini	3.63 (±.053)	2.79 (±.037)
GPT-3.5-Turbo	3.27 (±.012)	2.53 (±.077)

validating the efficacy of our system. The integration of hybrid search allows the system to balance semantic understanding with precise keyword matching, while relevance boosting ensures that the most credible and pertinent information is prioritized.

Future research will focus on extending the chatbot to a dynamic multi-agent system capable of intelligent query dissection, clarifying questions, and multi-hop reasoning to further enhance its conversational capabilities. This will involve developing mechanisms for the chatbot to break down complex queries into manageable components and engage in dialogue to clarify ambiguous questions. These advancements aim to further improve the chatbot's ability to handle complex interactions and provide users with even more reliable and insightful information.

CHAPTER 11

Conclusion

In this thesis, we have introduced novel representation learning techniques to improve the efficiency and reliability of financial document analytics with a particular emphasis on auditing. Our methodologies address the inherent challenges in processing complex corporate disclosure documents, e.g., technical terminology, large volumes, rigorous accounting standards to comply with, many numerical figures, and more, and significantly contribute to automating and refining their analysis. In the following sections, we summarize our main findings and provide an outlook on potential directions for future work.

11.1 Summary

Our research is organized into three main parts, with Part I tracing the evolution of representation learning techniques in NLP and Parts II and III building upon these foundations to develop novel methods for improved document consistency and compliance in the financial domain.

We started out in Chapter 2 by introducing the foundational building blocks of modern NLP, with a particular focus on (sequential) text classification, text matching, and the underlying concept of word embeddings. We indicated how the majority of problems in NLP can be formulated as supervised classification tasks with text matching being no exception. Leveraging a binary classification signal (related or unrelated) text matching aims to learn a latent similarity metric that enables the general matching of semantically similar text pairs without the need to previously define a fixed set of text categories. We further established the importance of word embeddings since the discrete nature of text, consisting of individual characters, is not directly compatible with the vectorized input of ML models. Thus, the overall goal of embedding methods is to properly encode the text's syntactic, semantic, and contextual nature into numerical vector representations. We traced the evolution of these methods from early frequency-based approaches to semantic embeddings, showcasing their respective advantages and limitations.

Acknowledging the lack of interpretability as one of these limitations, Chapter 3 presented a novel matrix factorization-based approach that enhances the interpretability of word embeddings by implicitly utilizing topic modeling. Leveraging the DEDICOM algorithm, we developed a method for creating interpretable word embeddings where each dimension corresponds to

distinct topics derived from the corpus. This approach not only retains the semantic richness of traditional embeddings but also provides a transparent framework for understanding the relationships between words and topics.

Chapter 4 addressed another core limitation of static word embeddings by introducing contextual awareness in the embedding learning process. Tightly linked to language modeling, we explored how deep neural network architectures like BERT and GPT, capable of modeling sequential dependencies, capture the dynamic meanings of words based on context. We discussed the differences of autoregressive and bidirectional models while also covering self-supervised learning, and transfer learning paradigms like fine-tuning and zero-shot learning. Additionally, we examined Retrieval-Augmented Generation (RAG) as a means to enhance the reliability of generative models by integrating external knowledge sources.

Transitioning to the application of these concepts, **Part II** focused on improving the consistency of financial documents. In Chapter 5, we laid the foundation for subsequent consistency checks by presenting a novel method for the joint extraction and linking of Key Performance Indicators (KPIs) from financial reports. KPIs like revenue or operating cash flows embody key financial information about the company's health and future prospects which is why their validity across reports is paramount. Recognizing that KPIs are often presented in varied linguistic forms and contexts, we developed *KPI-BERT*, a BERT-based language model that combines an RNN with conditional label masking to autoregressively extract KPIs before it classifies their relations. By jointly training the system's components end-to-end, our model effectively captures the contextual dependencies necessary for accurate KPI extraction and linking. This approach facilitates the automation of subsequent auditing tasks, such as consistency checks and trend analysis.

Building on *KPI-BERT*, Chapter 6 addressed the critical task of verifying the numerical consistency of KPIs across financial documents. We introduced *KPI-Check*, a system that employs BERT-based cross-encoders in conjunction with contrastive learning to generate robust embeddings for KPIs, enabling the automatic identification and cross-verification of semantically equivalent KPIs within financial reports. By utilizing dedicated modules like joint sentence and table encoding, and a contrastive autoencoder for classification, we effectively handled the data imbalance inherent in the KPI matching tasks. Our system demonstrated the ability to detect numerical inconsistencies, thereby enhancing the reliability of financial analyses.

In **Part III**, we focused on ensuring compliance with regulatory standards in financial reporting. Chapter 7 introduced *sustain.AI*, a context-aware recommender system designed to assist auditors in efficiently locating relevant text passages in sustainability reports corresponding to specific disclosure requirements. Utilizing a BERT-based encoding module and addressing class imbalance through weighted sampling, our system outperformed strong baselines, significantly improving retrieval performance measured in mean average precision. However, we recognized that processing paragraphs in isolation limited the model's ability to capture the full contextual meaning.

To overcome this limitation, Chapter 8 presented a novel pre-training method called *Pointer-Guided Segment Ordering (SO)*. By training a bidirectional language model to reconstruct the original order of shuffled text segments using a self-attention-based pointer network, we enhanced the model's understanding of narrative flow and inter-paragraph relationships. This resulted in contextually rich paragraph embeddings that improved semantic text classification performance. Our experiments demonstrated that models pre-trained with the SO task consis-

tently outperformed baselines, setting new state-of-the-art results in sequential text classification tasks.

Taking into account the dynamic nature of regulatory standards and the need for flexibility, Chapter 9 explored the use of Large Language Models (LLMs) for compliance verification. We introduced *ZeroShotALI*, a methodology that combines BERT-based text matching with an LLM-based re-ranking and compliance assessment module utilizing GPT-4. This approach enabled zero-shot matching between financial reports and unseen legal requirements without retraining, offering a scalable solution adaptable to evolving standards. We demonstrated the effectiveness of LLMs in assessing compliance, highlighting the potential of advanced models in auditing applications.

Finally, Chapter 10 tackled the challenges faced by organizations in navigating complex regulatory landscapes, particularly in Risk and Quality (R&Q) assurance. We developed a specialized chatbot powered by GPT with an optimized RAG pipeline. By integrating hybrid search techniques and relevance boosting, our system provided precise and contextually appropriate answers to users' queries related to R&Q standards. The chatbot was successfully deployed within PricewaterhouseCoopers GmbH, a globally operating accounting and consulting firm, enhancing the efficiency in handling internal compliance queries.

Collectively, our research demonstrates a comprehensive approach to addressing the complexities of financial document analytics using deep representation learning. By starting from foundational concepts and progressively tackling more complex challenges such as information extraction, relationship linking and consistency and compliance verification, we have developed a suite of methods that utilize advanced representation learning techniques to enhance auditing processes for financial disclosure documents.

11.2 Discussion and Outlook

While our work represents a substantial step forward in leveraging deep representation learning to improve the analysis of financial documents, several opportunities remain that offer avenues for future research.

In Chapter 10, our retrieval-based chatbot for R&Q assurance uses a single LLM to generate the final answer based on a fixed retrieval step. While being fast, a more flexible system would be capable of dynamically analyzing the request, potentially asking clarifying questions, planning and optimizing multiple retrieval steps to find the most relevant information, and generating the answer following specific formatting and source referencing instructions. In such complex settings, a single LLM may not always suffice. In future research, we plan to explore developing *LLM-Powered Multi-Agent Systems* [212, 213]. By employing a divide-and-conquer approach, such systems can mimic human teams, delegating subtasks to specialized agents that collaborate to solve complex tasks. Implementing self-correcting mechanisms and feedback loops could enhance the robustness and quality of the outputs. For instance, in our R&Q consulting chatbot, individual agents could handle tasks like query analysis, planning, search optimization, final answer formulation, and feedback integration, working together to produce comprehensive and accurate results.

Another area for future work is addressing the computational and privacy challenges associated with large-scale LLMs. Leveraging *Model Distillation* [214] and *Domain-Specific Fine-Tuning*

[215] could enable the creation of smaller, efficient models that retain the capabilities of larger LLMs while being more suitable for deployment in resource-constrained environments. By transferring knowledge from large models to smaller ones and fine-tuning them with domain-specific data, we can make advanced AI tools more accessible for smaller corporations and address data privacy concerns by avoiding reliance on external providers.

Additionally, our focus predominantly on text-based analysis may overlook the richness of financial documents that include charts, images, and other modalities. Exploring *Multi-Modal Foundation Modeling and Reasoning* [216] could allow for more comprehensive analysis by integrating information from various sources. Developing models capable of processing and reasoning over multi-modal data would enhance the ability to extract insights from complex documents, containing more than just textual data.

Building on this, recent advancements in *reasoning-focused LLMs*, such as OpenAI's o-series [217] and DeepSeek's R1 models [218], present promising opportunities for refining our LLM-based systems. These models have been fine-tuned via reinforcement learning to utilize internal chain-of-thoughts for improved reasoning at inference time before generating the final answer, which could significantly boost the performance of very complicated analytical tasks. However, their employment requires a careful evaluation within our specific use cases, as the increased computational demands from generating additional tokens during internal reasoning could negatively impact system efficiency.

Lastly, enhancing *Search and Retrieval* capabilities for highly diverse and unstructured document corpora remains an important area for improvement. Implementing multi-modal and structure-agnostic retrieval methods, such as automatically constructing knowledge graphs from unstructured data and utilizing graph relations [219] at retrieval time, could improve the relevancy of retrieved information.

APPENDIX A

DEDICOM for Interpretable Word Embeddings and Topic Modeling

A.1 Matrix derivatives for Non Negative DEDICOM

In this section, we derive the derivatives in 3.21 and 3.25 analytically.

The optimization problem can be formulated as

$$\min_{\boldsymbol{A},\boldsymbol{R}} \mathcal{L}(\boldsymbol{A},\boldsymbol{R}) \quad \text{s.t.} \quad \boldsymbol{A} \geq 0,\ \boldsymbol{R} \geq 0, \tag{A.1}$$

and we write the loss in trace form:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R}) &= \left\| \boldsymbol{S} - \boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^\top \right\|_F^2 && \text{(A.2)}\\
&= \operatorname{tr}\left[\left(\boldsymbol{S} - \boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^\top \right)^\top \left(\boldsymbol{S} - \boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^\top \right) \right] && \text{(A.3)}\\
&= \operatorname{tr}\left[\boldsymbol{Q}^\top \boldsymbol{Q} \right]. && \text{(A.4)}
\end{aligned}$$

Next, we differentiate the loss function

$$\begin{aligned}
\mathrm{d}\,\mathcal{L} &= \mathrm{d}\,\mathrm{tr}\left[\boldsymbol{Q}^\top \boldsymbol{Q}\right] && \text{(A.5)}\\
&= \mathrm{tr}\left[\mathrm{d}\left(\boldsymbol{Q}^\top \boldsymbol{Q}\right)\right] && \text{(A.6)}\\
&= \mathrm{tr}\left[(\mathrm{d}\,\boldsymbol{Q})^\top \boldsymbol{Q} + \boldsymbol{Q}^\top \mathrm{d}\,\boldsymbol{Q}\right] && \text{(A.7)}\\
&= \mathrm{tr}\left[\boldsymbol{Q}^\top \mathrm{d}\,\boldsymbol{Q} + \boldsymbol{Q}^\top \mathrm{d}\,\boldsymbol{Q}\right] && \text{(A.8)}\\
&= \mathrm{tr}\left[\left(\boldsymbol{Q}^\top + \boldsymbol{Q}^\top\right)\mathrm{d}\,\boldsymbol{Q}\right] && \text{(A.9)}\\
&= 2\,\mathrm{tr}\left[\boldsymbol{Q}^\top \mathrm{d}\,\boldsymbol{Q}\right] && \text{(A.10)}\\
&= 2\,\mathrm{tr}\left[\boldsymbol{Q}^\top \mathrm{d}\left(\boldsymbol{S} - \boldsymbol{A}'\boldsymbol{R}\boldsymbol{A}^\top\right)\right] && \text{(A.11)}\\
&= 2\,\mathrm{tr}\left[\boldsymbol{Q}^\top \mathrm{d}\,\boldsymbol{S}\right] - 2\,\mathrm{tr}\left[\boldsymbol{Q}^\top \mathrm{d}\left(\boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^\top\right)\right] && \text{(A.12)}\\
&= -2\,\mathrm{tr}\left[\boldsymbol{Q}^\top \mathrm{d}\left(\boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^\top\right)\right], && \text{(A.13)}
\end{aligned}$$

and express the resulting differential both in terms of $\mathrm{d}\,\boldsymbol{R}$

$$\begin{aligned}
\mathrm{d}\,\mathcal{L} &= -2\,\mathrm{tr}\left[\boldsymbol{Q}^\top \mathrm{d}\left(\boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^\top\right)\right] && \text{(A.14)}\\
&= -2\,\mathrm{tr}\left[\boldsymbol{Q}^\top \boldsymbol{A}\,\mathrm{d}\,\boldsymbol{R}\boldsymbol{A}^\top\right] && \text{(A.15)}\\
&= -2\,\mathrm{tr}\left[\boldsymbol{A}^\top \boldsymbol{Q}^\top \boldsymbol{A}\,\mathrm{d}\,\boldsymbol{R}\right], && \text{(A.16)}
\end{aligned}$$

and in terms of $\mathrm{d}\,\boldsymbol{A}$

$$\begin{aligned}
\mathrm{d}\,\mathcal{L} &= -2\,\mathrm{tr}\left[\boldsymbol{Q}^\top \mathrm{d}\left(\boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^\top\right)\right] && \text{(A.17)}\\
&= -2\,\mathrm{tr}\left[\boldsymbol{Q}^\top \mathrm{d}\,\boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^\top + \boldsymbol{Q}^\top \boldsymbol{A}\boldsymbol{R}(\mathrm{d}\,\boldsymbol{A})^\top\right] && \text{(A.18)}\\
&= -2\,\mathrm{tr}\left[\boldsymbol{R}\boldsymbol{A}^\top \boldsymbol{Q}^\top \mathrm{d}\,\boldsymbol{A} + \boldsymbol{R}^\top \boldsymbol{A}^\top \boldsymbol{Q}\,\mathrm{d}\,\boldsymbol{A}\right] && \text{(A.19)}\\
&= -2\,\mathrm{tr}\left[\left(\boldsymbol{R}\boldsymbol{A}^\top \boldsymbol{Q}^\top + \boldsymbol{R}^\top \boldsymbol{A}^\top \boldsymbol{Q}\right)\mathrm{d}\,\boldsymbol{A}\right]. && \text{(A.20)}
\end{aligned}$$

Hence, the final partial derivatives are given by

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \boldsymbol{R}} &= -2\left(\boldsymbol{A}^\top \boldsymbol{Q}\boldsymbol{A}\right) && \text{(A.21)}\\
&= -2\left(\boldsymbol{A}^\top \left(\boldsymbol{S} - \boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^\top\right)\boldsymbol{A}\right) && \text{(A.22)}\\
&= -2\left(\boldsymbol{A}^\top \boldsymbol{S}\boldsymbol{A} - \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^\top \boldsymbol{A}\right), && \text{(A.23)}
\end{aligned}$$

and

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{A}} = -2\left(\boldsymbol{Q}\boldsymbol{A}\boldsymbol{R}^\top + \boldsymbol{Q}^\top\boldsymbol{A}\boldsymbol{R}\right) \tag{A.24}$$

$$= -2\left(\left(\boldsymbol{S} - \boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^\top\right)\boldsymbol{A}\boldsymbol{R}^\top + \left(\boldsymbol{S} - \boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^\top\right)^\top\boldsymbol{A}\boldsymbol{R}\right) \tag{A.25}$$

$$= -2\left(\boldsymbol{S}\boldsymbol{A}\boldsymbol{R}^\top - \boldsymbol{A}\boldsymbol{R}\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{R}^\top + \boldsymbol{S}^\top\boldsymbol{A}\boldsymbol{R} - \boldsymbol{A}\boldsymbol{R}^\top\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{R}\right) \tag{A.26}$$

$$= -2\left(\boldsymbol{S}\boldsymbol{A}\boldsymbol{R}^\top + \boldsymbol{S}^\top\boldsymbol{A}\boldsymbol{R} - \boldsymbol{A}\left(\boldsymbol{R}\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{R}^\top + \boldsymbol{R}^\top\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{R}\right)\right). \tag{A.27}$$

A.2 Additional Results

This section presents additional results for our DEDICOM matrix and tensor factorization method to jointly identify topics and learn interpretable and semantic word embeddings. In addition, we showcase results for the baseline methods, NMF, LDA, and SVD. We present results on all our datasets: Wikipedia, Amazon Reviews, and New York Times.

A.2.1 Wikipedia – Matrix Input

Table A.1: Articles: “Soccer”, “Bee”, “Johnny Depp” – For each evaluated matrix factorization method we display the top 10 words for each topic and the 5 most similar words based on cosine similarity for the 2 top words from each topic.

		Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6
NMF		#619	#1238	#628	#595	#612	#389
	1	ball	bees	film	football	heard	album
	2	may	species	starred	cup	depp	band
	3	penalty	bee	role	world	court	guitar
	4	referee	pollen	series	fifa	alcohol	vampires
	5	players	honey	burton	national	relationship	rock
	6	team	insects	character	association	stated	hollywood
	7	goal	food	films	international	divorce	song
	8	game	nests	box	women	abuse	released
	9	player	solitary	office	teams	paradis	perry
	10	play	eusocial	jack	uefa	stating	debut
	0	ball	bees	film	football	heard	album
	1	invoke	odors	burtondirected	athenaeus	crew	jones
	2	replaced	tufts	tone	paralympic	alleging	marilyn
	3	scores	colour	landau	governing	oped	roots
	4	subdivided	affected	brother	varieties	asserted	drums
	0	may	species	starred	cup	depp	band
	1	yd	niko	shared	inaugurated	refer	heroes
	2	ineffectiveness	commercially	whitaker	confederation	york	bowie
	3	tactical	microbiota	eccentric	gold	leaders	debut
	4	slower	strategies	befriends	headquarters	nonindian	solo
LDA		#577	#728	#692	#607	#663	#814
	1	film	football	depp	penalty	bees	species
	2	series	women	children	heard	flowers	workers
	3	man	association	life	ball	bee	solitary
	4	played	fifa	role	direct	honey	players
	5	pirates	teams	starred	referee	pollen	colonies
	6	character	games	alongside	red	food	eusocial
	7	along	world	actor	time	increased	nest
	8	cast	cup	stated	goal	pollination	may
	9	also	game	burton	scored	times	size
	10	hollow	international	playing	player	larvae	egg
	0	film	football	depp	penalty	bees	species
	1	charlie	cup	critical	extra	bee	social
	2	near	canada	february	kicks	insects	chosen
	3	thinking	zealand	script	inner	authors	females
	4	shadows	activities	song	moving	hives	subspecies
	0	series	women	children	heard	flowers	workers
	1	crybaby	fifa	detective	allison	always	carcases
	2	waters	opera	crime	serious	eusociality	lived
	3	sang	exceeding	magazine	allergic	varroa	provisioned
	4	cast	cuju	barber	cost	wing	cuckoo
SVD		#1228	#797	#628	#369	#622	#437
	1	bees	depp	game	cup	heard	beekeeping
	2	also	film	ball	football	court	increased
	3	bee	starred	team	fifa	divorce	honey
	4	species	role	players	world	stating	described
	5	played	series	penalty	european	alcohol	use
	6	time	burton	play	uefa	paradis	wild
	7	one	character	may	national	documents	varroa
	8	first	actor	referee	europe	abuse	mites
	9	two	released	competitions	continental	settlement	colony
	10	pollen	release	laws	confederation	sued	flowers
	0	bees	depp	game	cup	heard	beekeeping
	1	bee	iii	correct	continental	alleging	varroa
	2	develops	racism	abandoned	contested	attempting	animals
	3	studied	appropriation	maximum	confederations	finalized	mites
	4	crops	march	clear	conmebol	submitted	plato
	0	also	film	ball	football	court	increased
	1	although	waters	finely	er	declaration	usage
	2	told	robinson	poised	suffix	issued	farmers
	3	chosen	scott	worn	word	restraining	mentioned
	4	stars	costars	manner	appended	verbally	aeneid

Table A.2: Articles: "Dolphin", "Shark", "Whale" – Top half lists the top 10 representative words per dimension of the basis matrix $\boldsymbol{A}$, bottom half lists the 5 most similar words based on cosine similarity for the 2 top words from each topic.

	Topic 1 #460	Topic 2 #665	Topic 3 #801	Topic 4 #753	Topic 5 #854	Topic 6 #721
1	shark (0.665)	calf (0.428)	ship (0.459)	conservation (0.334)	water (0.416)	dolphin (0.691)
2	sharks (0.645)	months (0.407)	became (0.448)	countries (0.312)	similar (0.374)	dolphins (0.655)
3	fins (0.487)	calves (0.407)	poseidon (0.44)	government (0.309)	tissue (0.373)	captivity (0.549)
4	killed (0.454)	females (0.399)	riding (0.426)	wales (0.304)	body (0.365)	wild (0.467)
5	million (0.451)	blubber (0.374)	dionysus (0.422)	bycatch (0.29)	swimming (0.357)	behavior (0.461)
6	fish (0.448)	young (0.37)	ancient (0.42)	cancelled (0.288)	blood (0.346)	bottlenose (0.453)
7	international (0.442)	sperm (0.356)	deity (0.412)	eastern (0.287)	surface (0.344)	sometimes (0.449)
8	fin (0.421)	born (0.355)	ago (0.398)	policy (0.286)	oxygen (0.34)	human (0.421)
9	fishing (0.405)	feed (0.349)	melicertes (0.395)	control (0.285)	system (0.336)	less (0.42)
10	teeth (0.398)	mysticetes (0.341)	greeks (0.394)	imminent (0.282)	swim (0.336)	various (0.418)
0	shark (1.0)	calf (1.0)	ship (1.0)	conservation (1.0)	water (1.0)	dolphin (1.0)
2	sharks (0.981)	calves (0.978)	dionysus (0.995)	south (0.981)	prey (0.964)	dolphins (0.925)
3	fins (0.958)	females (0.976)	riding (0.992)	states (0.981)	swimming (0.959)	sometimes (0.909)
4	killed (0.929)	months (0.955)	deity (0.992)	united (0.978)	allows (0.957)	another (0.904)
5	fishing (0.916)	young (0.948)	poseidon (0.987)	endangered (0.976)	swim (0.947)	bottlenose (0.903)
0	sharks (1.0)	months (1.0)	became (1.0)	countries (1.0)	similar (1.0)	dolphins (1.0)
2	shark (0.981)	born (0.992)	old (0.953)	eastern (0.991)	surface (0.992)	behavior (0.956)
3	fins (0.936)	young (0.992)	later (0.946)	united (0.989)	brain (0.97)	sometimes (0.945)
4	tiger (0.894)	sperm (0.985)	ago (0.939)	caught (0.987)	sound (0.968)	various (0.943)
5	killed (0.887)	calves (0.984)	modern (0.937)	south (0.979)	object (0.965)	less (0.937)

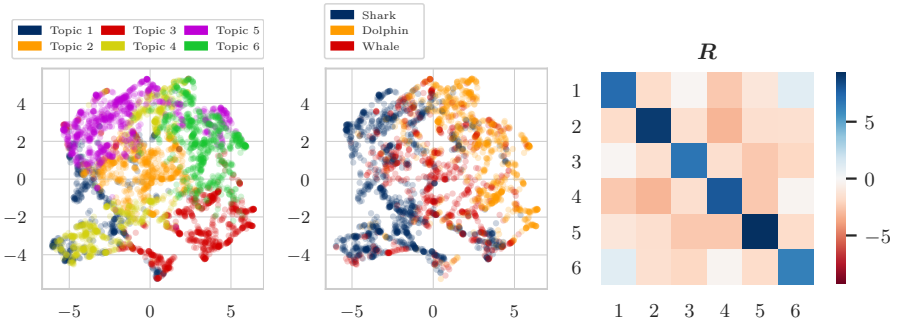


Figure A.1: Articles: "Dolphin", "Shark", "Whale" – (a) 2-dimensional representation of word embeddings $\boldsymbol{A}'$ colored by topic assignment. (b) 2-dimensional representation of word embeddings $\boldsymbol{A}'$ colored by original Wikipedia article assignment (words that occur in more than one article are excluded). (c) Colored heatmap of affinity matrix $\boldsymbol{R}$.

Table A.3: Articles: “Dolphin”, “Shark”, “Whale” – For each evaluated matrix factorization method we display the top 10 words for each topic and the 5 most similar words based on cosine similarity for the 2 top words from each topic.

		Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6
NMF		#492	#907	#452	#854	#911	#638
	1	blood	international	evidence	sonar	ago	calf
	2	body	killed	selfawareness	may	teeth	young
	3	heart	states	ship	surface	million	females
	4	gills	conservation	dionysus	clicks	mysticetes	captivity
	5	bony	new	came	prey	whales	calves
	6	oxygen	united	another	use	years	months
	7	organs	shark	important	underwater	baleen	born
	8	tissue	world	poseidon	sounds	cetaceans	species
	9	water	endangered	mark	known	modern	male
	10	via	islands	riding	similar	extinct	female
	0	blood	international	evidence	sonar	ago	calf
	1	travels	proposal	flaws	poisoned	consist	uninformed
	2	enters	lipotidae	methodological	signals	specialize	primary
	3	vibration	banned	nictating	≈–	legs	born
	4	tolerant	iniidae	wake	emitted	closest	leaner
	0	body	killed	selfawareness	may	teeth	young
	1	crystal	law	legendary	individuals	fuel	brood
	2	blocks	consumers	humankind	helping	lamp	lacking
	3	modified	pontoporiidae	helpers	waste	filterfeeding	accurate
	4	slits	org	performing	depression	krill	consistency
LDA		#650	#785	#695	#815	#635	#674
	1	killed	teeth	head	species	meat	air
	2	system	baleen	fish	male	whale	using
	3	endangered	mysticetes	dolphin	females	ft	causing
	4	often	ago	fin	whales	fisheries	currents
	5	close	jaw	eyes	sometimes	also	sounds
	6	sharks	family	fat	captivity	ocean	groups
	7	countries	water	navy	young	threats	sound
	8	since	includes	popular	shark	children	research
	9	called	allow	tissue	female	population	clicks
	10	vessels	greater	tail	wild	bottom	burst
	0	killed	teeth	head	species	meat	air
	1	postures	dense	underside	along	porbeagle	australis
	2	dolphinariums	cetacea	grooves	another	source	submerged
	3	town	tourism	eyesight	long	activities	melbourne
	4	onethird	planktonfeeders	osmoregulation	sleep	comparable	spear
	0	system	baleen	fish	male	whale	using
	1	dominate	mysticetes	mostly	females	live	communication
	2	close	distinguishing	swim	aorta	human	become
	3	controversy	unique	due	female	cold	associated
	4	agree	remove	whole	position	parts	mirror
SVD		#1486	#544	#605	#469	#539	#611
	1	dolphins	water	shark	million	poseidon	dolphin
	2	species	body	sharks	years	became	meat
	3	whales	tail	fins	ago	ship	family
	4	fish	teeth	international	whale	riding	river
	5	also	flippers	killed	two	evidence	similar
	6	large	tissue	fishing	calf	melicertes	extinct
	7	may	allows	fin	mya	deity	called
	8	one	air	law	later	ino	used
	9	animals	feed	new	months	came	islands
	10	use	bony	conservation	mysticetes	made	genus
	0	dolphins	water	shark	million	poseidon	dolphin
	1	various	vertical	corpse	approximately	games	depicted
	2	finding	unlike	stocks	assigned	phalanthus	makara
	3	military	chew	galea	hybodonts	statue	capensis
	4	selfmade	lack	galeomorphii	appeared	isthmian	goddess
	0	species	body	sharks	years	became	meat
	1	herd	heart	mostly	acanthodians	pirates	contaminated
	2	reproduction	resisting	fda	spent	elder	harpoon
	3	afford	fit	lists	stretching	mistook	practitioner
	4	maturity	posterior	carcharias	informal	wealthy	pcbs

Table A.4: Articles: "Soccer", "Tennis", "Rugby" – Top half lists the top 10 representative words per dimension of the basis matrix $\boldsymbol{A}$, bottom half lists the 5 most similar words based on cosine similarity for the 2 top words from each topic.

	Topic 1 #539	Topic 2 #302	Topic 3 #563	Topic 4 #635	Topic 5 #650	Topic 6 #530
1	may (0.599)	leads (0.212)	tournaments (0.588)	greatest (0.572)	football (0.553)	net (0.644)
2	penalty (0.576)	sole (0.205)	tournament (0.517)	tennis (0.497)	rugby (0.542)	shot (0.629)
3	referee (0.564)	competes (0.205)	events (0.509)	female (0.44)	south (0.484)	stance (0.553)
4	team (0.517)	extending (0.204)	prize (0.501)	ever (0.433)	union (0.47)	stroke (0.543)
5	goal (0.502)	fixing (0.203)	tour (0.497)	navratilova (0.405)	wales (0.459)	serve (0.537)
6	kick (0.459)	triggered (0.203)	money (0.488)	modern (0.401)	national (0.446)	rotation (0.513)
7	play (0.455)	bleeding (0.202)	cup (0.486)	best (0.4)	england (0.438)	backhand (0.508)
8	ball (0.452)	fraud (0.202)	world (0.467)	wingfield (0.394)	new (0.416)	hit (0.507)
9	offence (0.444)	inflammation (0.202)	atp (0.464)	sports (0.39)	europe (0.406)	forehand (0.499)
10	foul (0.443)	conditions (0.201)	men (0.463)	williams (0.389)	states (0.404)	torso (0.487)
0	may (1.0)	leads (1.0)	tournaments (1.0)	greatest (1.0)	football (1.0)	net (1.0)
2	goal (0.98)	tiredness (1.0)	events (0.992)	female (0.98)	union (0.98)	shot (0.994)
3	play (0.959)	ineffectiveness (1.0)	tour (0.989)	ever (0.971)	rugby (0.979)	serve (0.987)
4	penalty (0.954)	recommences (1.0)	money (0.986)	navratilova (0.967)	association (0.96)	hit (0.984)
5	team (0.953)	mandated (1.0)	prize (0.985)	tennis (0.962)	england (0.958)	stance (0.955)
0	penalty (1.0)	sole (1.0)	tournament (1.0)	tennis (1.0)	rugby (1.0)	shot (1.0)
2	referee (0.985)	discretion (1.0)	events (0.98)	greatest (0.962)	football (0.979)	net (0.994)
3	kick (0.985)	synonym (1.0)	event (0.978)	female (0.953)	union (0.975)	serve (0.987)
4	offence (0.982)	violated (1.0)	atp (0.974)	year (0.951)	england (0.961)	hit (0.983)
5	foul (0.982)	layout (1.0)	money (0.966)	navratilova (0.949)	wales (0.949)	stance (0.98)

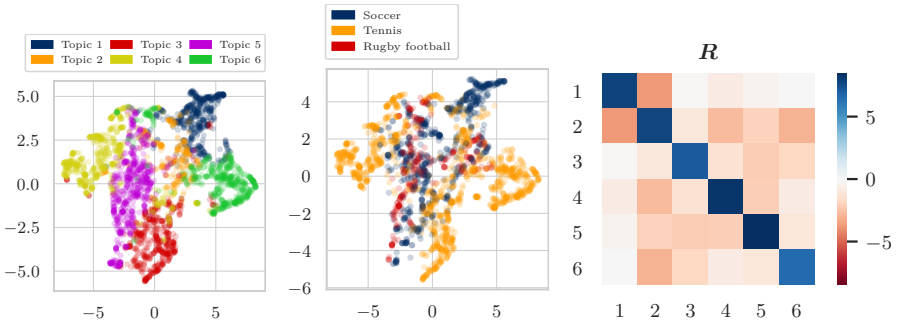


Figure A.2: Articles: "Soccer", "Tennis", "Rugby" – (a) 2-dimensional representation of word embeddings $\boldsymbol{A}'$ colored by topic assignment. (b) 2-dimensional representation of word embeddings $\boldsymbol{A}'$ colored by original Wikipedia article assignment (words that occur in more than one article are excluded). (c) Colored heatmap of affinity matrix $\boldsymbol{R}$.

Table A.5: Articles: “Soccer”, “Tennis”, “Rugby” – For each evaluated matrix factorization method we display the top 10 words for each topic and the 5 most similar words based on cosine similarity for the 2 top words from each topic.

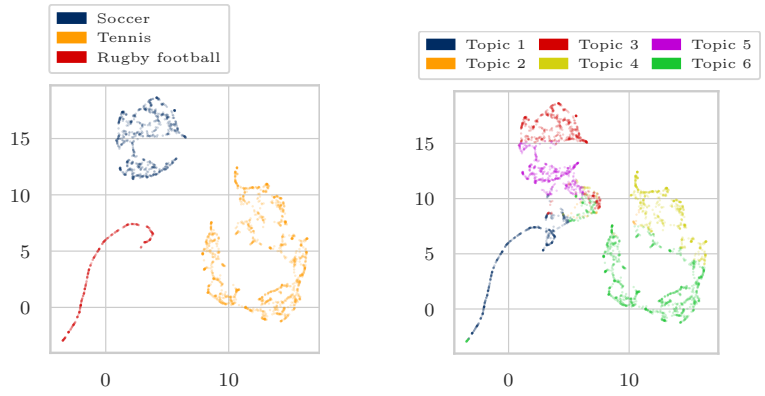
		Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6
NMF		#511	#453	#575	#657	#402	#621
	1	net	referee	national	tournaments	rackets	rules
	2	shot	penalty	south	doubles	balls	wingfield
	3	serve	may	football	singles	made	december
	4	hit	kick	cup	events	size	game
	5	stance	card	europe	tour	must	sports
	6	stroke	listed	fifa	prize	strings	lawn
	7	backhand	foul	union	money	standard	modern
	8	ball	misconduct	wales	atp	synthetic	greek
	9	server	red	africa	men	leather	fa
	10	service	offence	new	grand	width	first
	0	net	referee	national	tournaments	rackets	rules
	1	defensive	retaken	serbia	bruno	pressurisation	collection
	2	closer	interference	gold	woodies	become	hourglass
	3	somewhere	dismissed	north	eliminated	equivalents	unhappy
	4	center	fully	headquarters	soares	size	originated
	0	shot	penalty	south	doubles	balls	wingfield
	1	rotated	prior	asian	combining	express	experimenting
	2	execute	yellow	argentina	becker	oz	llanelidan
	3	strive	duration	la	exclusively	bladder	attended
	4	curve	primary	kong	woodbridge	length	antiphanes
LDA		#413	#518	#395	#776	#616	#501
	1	used	net	wimbledon	world	penalty	clubs
	2	forehand	ball	episkyros	cup	score	rugby
	3	use	serve	occurs	tournaments	goal	schools
	4	large	shot	grass	football	team	navratilova
	5	notable	opponent	roman	fifa	end	forms
	6	also	hit	bc	national	players	playing
	7	western	lines	occur	international	match	sport
	8	twohanded	server	ad	europe	goals	greatest
	9	doubles	service	island	tournament	time	union
	10	injury	may	believed	states	scored	war
	0	used	net	wimbledon	world	penalty	clubs
	1	seconds	mistaken	result	british	measure	sees
	2	restrictions	diagonal	determined	cancelled	crossed	papua
	3	although	hollow	exists	combined	requiring	admittance
	4	use	perpendicular	win	wii	teammate	forces
	0	forehand	ball	episkyros	cup	score	rugby
	1	twohanded	long	roman	multiple	penalty	union
	2	grips	deuce	bc	inline	bar	public
	3	facetiously	position	island	fifa	fouled	took
	4	woodbridge	allows	believed	manufactured	hour	published
SVD		#1310	#371	#423	#293	#451	#371
	1	players	net	tournaments	stroke	greatest	balls
	2	player	ball	singles	forehand	ever	rackets
	3	tennis	shot	doubles	stance	female	size
	4	also	serve	tour	power	wingfield	square
	5	play	opponent	slam	backhand	williams	made
	6	football	may	prize	torso	navratilova	leather
	7	team	hit	money	grip	game	weight
	8	first	service	grand	rotation	said	standard
	9	one	hitting	events	twohanded	serena	width
	10	rugby	line	ranking	used	sports	past
	0	players	net	tournaments	stroke	greatest	balls
	1	breaking	pace	masters	rotates	lived	panels
	2	one	reach	lowest	achieve	female	sewn
	3	running	underhand	events	face	biggest	entire
	4	often	air	tour	adds	potential	leather
	0	player	ball	singles	forehand	ever	rackets
	1	utilize	keep	indian	twohanded	autobiography	meanwhile
	2	give	hands	doubles	begins	jack	laminated
	3	converted	pass	pro	backhand	consistent	wood
	4	touch	either	rankings	achieve	gonzales	strings

A.2.2 Wikipedia – Tensor Input

Articles "Soccer", "Bee", "Johnny Depp" – DEDICOM Automatic gradient method

Table A.6: For the most significant two words per topic, the four nearest neighbors based on cosine similarity are listed. Matrix $\boldsymbol{A}'$, trained on the wikipedia data as input tensor using automatic gradient methods.

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6
0	hind (1.0)	game (1.0)	film (1.0)	heard (1.0)	bees (1.0)	disorder (1.0)
1	segments (0.995)	football (1.0)	starred (0.968)	court (0.954)	bee (0.999)	collapse (1.0)
2	legs (0.994)	players (0.999)	role (0.954)	divorce (0.925)	honey (0.99)	losses (1.0)
3	antennae (0.993)	ball (0.999)	series (0.951)	sued (0.907)	insects (0.976)	attrition (0.999)
4	wings (0.992)	team (0.998)	burton (0.945)	alleged (0.897)	food (0.97)	businesses (0.999)
0	segments (1.0)	football (1.0)	starred (1.0)	court (1.0)	bee (1.0)	collapse (1.0)
1	antennae (1.0)	game (1.0)	role (0.993)	divorce (0.996)	bees (0.999)	disorder (1.0)
2	wings (0.999)	players (0.999)	series (0.978)	sued (0.991)	honey (0.995)	losses (0.999)
3	bacteria (0.999)	ball (0.999)	burton (0.975)	alleged (0.981)	insects (0.984)	pesticide (0.998)
4	legs (0.998)	team (0.999)	film (0.968)	alcohol (0.981)	food (0.976)	businesses (0.998)



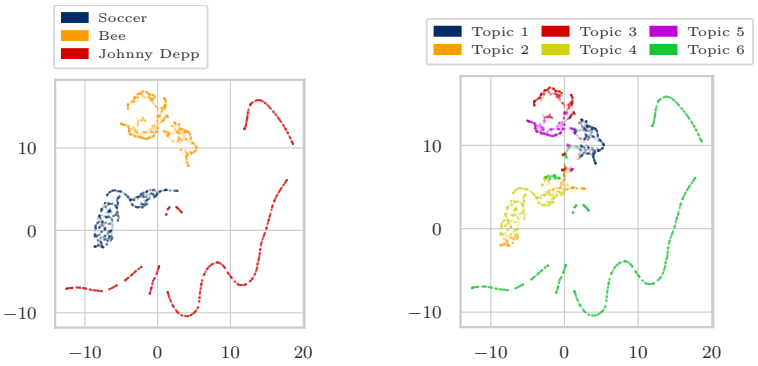
(a) (b)

Figure A.3: (a) 2-dimensional representation of word embeddings $\boldsymbol{A}'$ colored by topic assignment. (b) 2-dimensional representation of word embeddings $\boldsymbol{A}'$ colored by original Wikipedia article assignment (words that occur in more than one article are excluded).

Articles "Soccer", "Bee", "Johnny Depp" – DEDICOM Multiplicative Update Rules

Table A.7: For the most significant two words per topic, the four nearest neighbors based on cosine similarity are listed. Matrix $\boldsymbol{A}$, trained on the wikipedia data as input tensor using multiplicative update rules.

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6
0	species (1.0)	game (1.0)	honey (1.0)	allow (1.0)	insects (1.0)	depp (1.0)
1	easier (1.0)	football (1.0)	boatwrights (1.0)	emancipation (1.0)	ultraviolet (1.0)	charlie (1.0)
2	tiny (1.0)	players (1.0)	glade (1.0)	broadly (1.0)	mechanics (1.0)	infiltrate (1.0)
3	halictidae (0.999)	association (1.0)	tutankhamun (1.0)	disabilities (1.0)	exploit (1.0)	thenwife (1.0)
4	provision (0.999)	team (0.997)	oracle (1.0)	total (1.0)	swallows (1.0)	tourist (1.0)
0	eusocial (1.0)	football (1.0)	bee (1.0)	organised (1.0)	pollen (1.0)	film (1.0)
1	oligocene (1.0)	players (1.0)	subfamilies (0.995)	comes (1.0)	honeybees (1.0)	starred (1.0)
2	architecture (1.0)	game (1.0)	internal (0.994)	shows (1.0)	enlarged (0.998)	smoking (0.999)
3	uncommon (1.0)	association (1.0)	studied (0.994)	attention (1.0)	simple (0.998)	dislocated (0.999)
4	termed (1.0)	team (0.997)	cladogram (0.99)	deductions (1.0)	drove (0.998)	injuries (0.999)



(a) (b)

Figure A.4: (a) 2-dimensional representation of word embeddings $\boldsymbol{A}'$ colored by topic assignment. (b) 2-dimensional representation of word embeddings $\boldsymbol{A}'$ colored by original Wikipedia article assignment (words that occur in more than one article are excluded).

Articles "Dolphin", "Shark", "Whale" – DEDICOM Multiplicative Update Rules

Table A.8: Each column lists the top 10 representative words per dimension of the basis matrix $\boldsymbol{A}$.

	Topic 1 #226	Topic 2 #628	Topic 3 #1048	Topic 4 #571	Topic 5 #1267	Topic 6 #554
1	cells (1.785)	mysticetes (1.808)	shark (3.019)	bony (1.621)	dolphin (3.114)	whaling (3.801)
2	brain (1.624)	whales (1.791)	sharks (2.737)	blood (1.452)	dolphins (2.908)	iwc (2.159)
3	light (1.561)	feed (1.427)	fins (1.442)	fish (1.438)	bottlenose (1.629)	aboriginal (2.098)
4	cone (1.448)	baleen (1.33)	killed (1.407)	gills (1.206)	meat (1.403)	canada (1.912)
5	allow (1.32)	odontocetes (1.278)	endangered (1.377)	teeth (1.088)	behavior (1.399)	moratorium (1.867)
6	greater (1.292)	consist (1.162)	hammerhead (1.269)	body (1.043)	captivity (1.298)	industry (1.855)
7	slightly (1.269)	water (1.096)	conservation (1.227)	system (1.027)	river (1.281)	us (1.838)
8	ear (1.219)	krill (1.05)	trade (1.226)	skeleton (1.008)	common (1.275)	belugas (1.585)
9	cornea (1.158)	toothed (1.003)	whitetip (1.203)	called (0.99)	selfawareness (1.248)	whale (1.542)
10	rod (1.128)	sperm (0.991)	finning (1.184)	tissue (0.875)	often (1.218)	gb£ (1.528)

Table A.9: For the most significant two words per topic, the four nearest neighbors based on cosine similarity are listed.

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6
0	cells (1.0)	mysticetes (1.0)	shark (1.0)	bony (1.0)	dolphin (1.0)	whaling (1.0)
1	sensitive (1.0)	unborn (1.0)	native (1.0)	edges (1.0)	hybrid (1.0)	māori (1.0)
2	cone (1.0)	grind (1.0)	tl (1.0)	mirabile (1.0)	hybridization (1.0)	trips (1.0)
3	rod (0.998)	counterparts (1.0)	predators—organisms (1.0)	matches (1.0)	yangtze (1.0)	predominantly (1.0)
4	corneas (0.998)	threechambered (1.0)	cretaceous (1.0)	turbulence (1.0)	grampus (1.0)	revenue (1.0)
0	brain (1.0)	whales (1.0)	sharks (1.0)	blood (1.0)	dolphins (1.0)	iwc (1.0)
1	receive (0.998)	extended (0.996)	reminiscent (1.0)	hydrodynamic (0.998)	superpod (1.0)	distinction (1.0)
2	equalizer (0.998)	bryde (0.996)	electrical (1.0)	scattering (0.998)	masturbation (1.0)	billion (1.0)
3	lobes (0.997)	closes (0.996)	induced (1.0)	reminder (0.998)	interaction (1.0)	spain (1.0)
4	clear (0.997)	effects (0.996)	coarsely (1.0)	flows (0.998)	stressful (1.0)	competition (1.0)

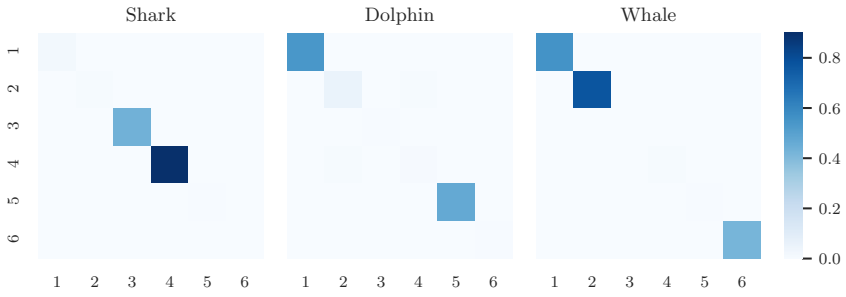


Figure A.5: Colored heatmap of affinity tensor $\boldsymbol{R}$.

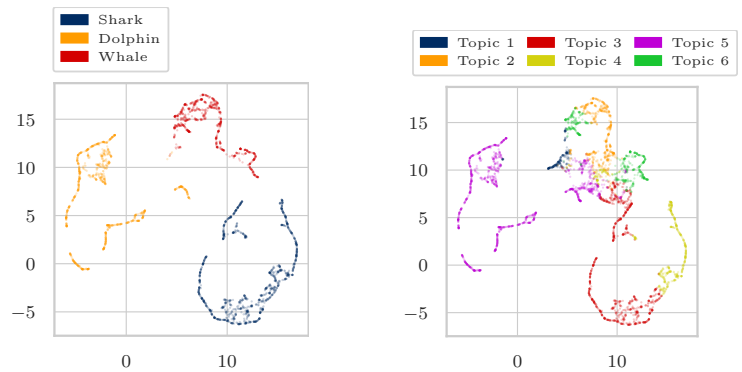


Figure A.6: (a) 2-dimensional representation of word embeddings $\boldsymbol{A}$ colored by topic assignment. (b) 2-dimensional representation of word embeddings $\boldsymbol{A}$ colored by original Wikipedia article assignment (words that occur in more than one article are excluded).

Articles "Soccer", "Tennis", "Rugby" – DEDICOM Multiplicative Update Rules

Table A.10: Each column lists the top 10 representative words per dimension of the basis matrix $\boldsymbol{A}$.

	Topic 1 #441	Topic 2 #861	Topic 3 #412	Topic 4 #482	Topic 5 #968	Topic 6 #57
1	rugby (2.55)	titles (1.236)	rackets (2.176)	net (2.767)	penalty (1.721)	doubles (2.335)
2	union (2.227)	wta (1.196)	wingfield (1.536)	shot (2.586)	football (1.701)	singles (2.321)
3	wales (1.822)	circuit (1.123)	modern (1.513)	serve (2.393)	team (1.507)	tournaments (2.245)
4	georgia (1.682)	futures (1.122)	racket (1.43)	hit (1.978)	laws (1.462)	tennis (1.752)
5	fiji (1.557)	earn (1.104)	th (1.355)	stance (1.945)	referee (1.449)	grand (1.662)
6	samoa (1.474)	offer (1.096)	lawn (1.316)	service (1.83)	fifa (1.439)	events (1.648)
7	zealand (1.458)	mixed (1.089)	century (1.236)	stroke (1.797)	may (1.435)	slam (1.623)
8	new (1.414)	draws (1.085)	strings (1.179)	server (1.761)	goal (1.353)	player (1.344)
9	tonga (1.374)	atp (1.072)	yielded (1.121)	backhand (1.692)	competitions (1.345)	professional (1.328)
10	south (1.369)	challenger (1.07)	balls (1.101)	forehand (1.554)	associations (1.288)	players (1.316)

Table A.11: For the most significant two words per topic, the four nearest neighbors based on cosine similarity are listed.

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6
0	rugby (1.0)	titles (1.0)	rackets (1.0)	net (1.0)	penalty (1.0)	doubles (1.0)
1	ireland (1.0)	hopman (1.0)	proximal (1.0)	hit (1.0)	organisers (1.0)	singles (1.0)
2	union (1.0)	dress (1.0)	interlaced (1.0)	formally (1.0)	elapsed (1.0)	tournaments (0.985)
3	backfired (1.0)	tennischannel (0.998)	harry (1.0)	offensive (1.0)	polite (1.0)	grand (0.975)
4	kilopascals (1.0)	seoul (0.998)	deserves (1.0)	deeply (1.0)	modest (1.0)	slam (0.971)
0	union (1.0)	wta (1.0)	wingfield (1.0)	shot (1.0)	football (1.0)	singles (1.0)
1	rugby (1.0)	helps (1.0)	proximal (1.0)	requires (1.0)	circumference (1.0)	doubles (1.0)
2	ireland (1.0)	hamilton (1.0)	interlaced (1.0)	backwards (1.0)	touchline (1.0)	tournaments (0.985)
3	backfired (1.0)	weeks (1.0)	harry (1.0)	entail (1.0)	sanctions (1.0)	grand (0.975)
4	zealand (1.0)	couple (1.0)	deserves (1.0)	torso (1.0)	home (0.999)	slam (0.971)

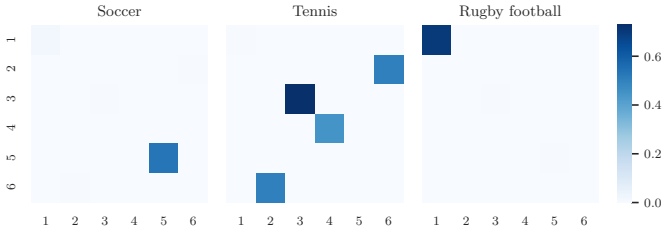
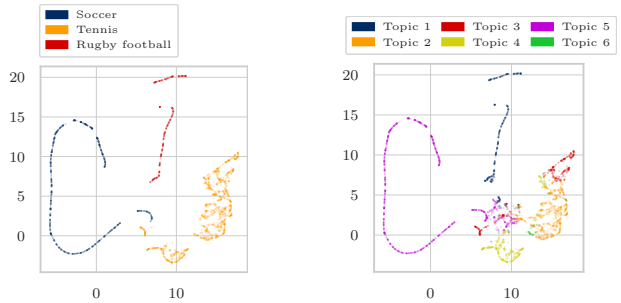


Figure A.7: Colored heatmap of affinity tensor $\boldsymbol{R}$.



(a) (b)

Figure A.8: (a) 2-dimensional representation of word embeddings $\boldsymbol{A}$ colored by topic assignment. (b) 2-dimensional representation of word embeddings $\boldsymbol{A}$ colored by original Wikipedia article assignment (words that occur in more than one article are excluded).

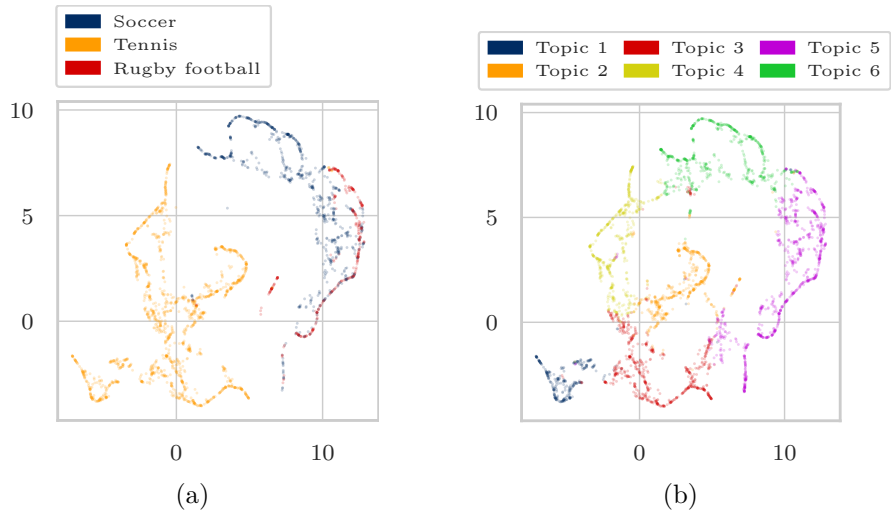
Articles "Soccer", "Tennis", "Rugby" – Tensor NMF

Table A.12: Each column lists the top 10 representative words per dimension of the basis matrix $\boldsymbol{A}'$.

	Topic 1 #275	Topic 2 #505	Topic 3 #607	Topic 4 #459	Topic 5 #816	Topic 6 #559
1	greatest (39.36)	rackets (33.707)	tournaments (29.126)	net (29.789)	football (27.534)	penalty (27.793)
2	ever (26.587)	modern (24.281)	events (25.327)	shot (27.947)	rugby (24.037)	referee (23.632)
3	female (25.52)	balls (22.016)	tour (23.488)	serve (25.722)	union (21.397)	goal (23.072)
4	navratilova (24.348)	wingfield (20.923)	prize (21.823)	hit (21.344)	south (20.761)	may (22.978)
5	best (24.114)	tennis (19.863)	atp (21.124)	stance (20.75)	national (19.586)	team (21.258)
6	williams (22.207)	strings (18.602)	money (20.667)	service (19.7)	fifa (19.331)	kick (21.052)
7	serena (21.256)	racket (18.369)	doubles (19.919)	server (19.051)	wales (18.627)	foul (19.018)
8	said (20.666)	made (17.622)	ranking (19.736)	stroke (18.781)	league (18.31)	listed (17.736)
9	martina (20.153)	yielded (17.284)	us (19.431)	backhand (17.809)	cup (17.015)	free (17.702)
10	budge (20.111)	th (16.992)	masters (18.596)	ball (17.2)	association (16.721)	goals (17.209)

Table A.13: For the most significant two words per topic, the four nearest neighbors based on cosine similarity are listed.

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6
0	greatest (1.0)	rackets (1.0)	tournaments (1.0)	net (1.0)	football (1.0)	penalty (1.0)
1	illustrated (1.0)	garden (1.0)	us (1.0)	lob (1.0)	midlothian (1.0)	whole (1.0)
2	johansson (1.0)	construction (1.0)	earned (1.0)	receiving (1.0)	alcock (1.0)	corner (1.0)
3	wilton (1.0)	yielded (1.0)	participating (1.0)	rotates (1.0)	capital (1.0)	offender (1.0)
4	jonathan (1.0)	energy (1.0)	receives (1.0)	adds (1.0)	representatives (1.0)	stoke (1.0)
0	ever (1.0)	modern (1.0)	events (1.0)	shot (1.0)	rugby (1.0)	referee (1.0)
1	deserved (1.0)	design (0.999)	juniors (1.0)	lobber (1.0)	slang (1.0)	dismissed (1.0)
2	stated (1.0)	version (0.999)	bowl (1.0)	unable (1.0)	colonists (1.0)	showing (1.0)
3	female (1.0)	shape (0.998)	comprised (1.0)	alter (1.0)	sevenaside (1.0)	stoppage (1.0)
4	contemporaries (1.0)	stitched (0.998)	carlo (1.0)	applying (1.0)	seldom (1.0)	layout (1.0)



(a) (b)

Figure A.9: (a) 2-dimensional representation of word embeddings $\boldsymbol{H}$ colored by topic assignment. (b) 2-dimensional representation of word embeddings $\boldsymbol{H}$ colored by original Wikipedia article assignment (words that occur in more than one article are excluded).

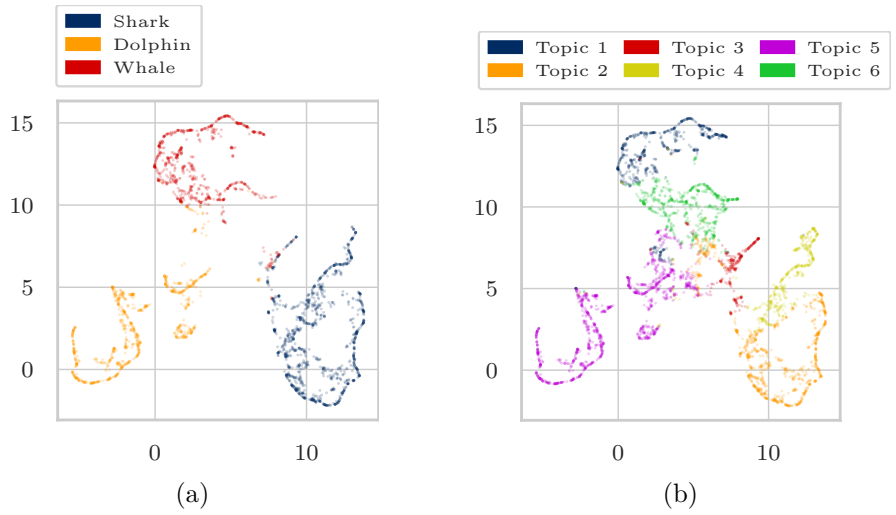
Articles "Dolphin", "Shark", "Whale" – Tensor NMF

Table A.14: Each column lists the top 10 representative words per dimension of the basis matrix $\boldsymbol{H}$.

	Topic 1 #675	Topic 2 #996	Topic 3 #279	Topic 4 #491	Topic 5 #1190	Topic 6 #663
1	whaling (34.584)	sharks (30.418)	young (33.823)	killed (23.214)	dolphin (35.52)	mysticetes (24.404)
2	whale (25.891)	fish (23.648)	born (27.62)	shark (22.6)	dolphins (31.881)	flippers (22.059)
3	whales (21.653)	bony (19.689)	oviduct (23.706)	states (21.24)	bottlenose (18.198)	odontocetes (21.621)
4	belugas (20.933)	prey (18.785)	viviparity (23.694)	endangered (20.976)	behavior (18.003)	water (21.087)
5	aboriginal (19.44)	teeth (18.242)	embryos (22.966)	conservation (20.398)	selfawareness (16.48)	tail (18.268)
6	iwc (19.226)	blood (16.521)	continue (21.752)	fins (18.641)	meat (16.02)	mya (17.79)
7	canada (18.691)	gills (13.34)	calves (21.25)	new (18.445)	often (15.687)	baleen (17.189)
8	arctic (17.406)	tissue (12.927)	blubber (21.094)	international (18.4)	captivity (15.452)	limbs (16.56)
9	industry (16.837)	body (12.691)	egg (20.735)	drum (17.587)	river (14.68)	allow (16.552)
10	right (16.766)	skeleton (12.52)	fluids (20.662)	finning (17.321)	common (14.389)	toothed (16.489)

Table A.15: For the most significant two words per topic, the four nearest neighbors based on cosine similarity are listed.

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6
0	whaling (1.0)	sharks (1.0)	young (1.0)	killed (1.0)	dolphin (1.0)	mysticetes (1.0)
1	antarctica (1.0)	loan (1.0)	getting (1.0)	alzheimer (1.0)	behaviors (1.0)	digits (1.0)
2	spain (1.0)	leopard (1.0)	insulation (1.0)	queensland (1.0)	familiar (1.0)	streamlined (1.0)
3	caro (1.0)	dogfish (1.0)	harsh (1.0)	als (1.0)	pantropical (1.0)	archaeocete (1.0)
4	excluded (1.0)	lifespans (0.999)	primary (1.0)	control (1.0)	test (1.0)	defines (0.999)
0	whale (1.0)	fish (1.0)	born (1.0)	shark (1.0)	dolphins (1.0)	flippers (1.0)
1	reason (1.0)	like (0.997)	getting (1.0)	figure (1.0)	levels (1.0)	expel (1.0)
2	respected (1.0)	lifetime (0.992)	young (1.0)	sources (1.0)	moderate (0.999)	compress (1.0)
3	divinity (0.999)	content (0.992)	leaner (1.0)	video (0.998)	injuries (0.999)	protocetus (1.0)
4	taken (0.998)	hazardous (0.992)	insulation (1.0)	dogfishes (0.997)	seems (0.998)	nostrils (1.0)



(a) (b)

Figure A.10: (a) 2-dimensional representation of word embeddings $\boldsymbol{H}$ colored by topic assignment. (b) 2-dimensional representation of word embeddings $\boldsymbol{H}$ colored by original Wikipedia article assignment (words that occur in more than one article are excluded).

Articles "Soccer", "Bee", "Johnny Depp" – Tensor NMF

Table A.16: Each column lists the top 10 representative words per dimension of the basis matrix $\boldsymbol{H}$.

	Topic 1 #793	Topic 2 #554	Topic 3 #736	Topic 4 #601	Topic 5 #740	Topic 6 #616
1	film (37.29)	ball (27.588)	honey (29.778)	football (32.591)	heard (37.167)	species (32.973)
2	starred (23.821)	may (25.768)	insects (27.784)	fifa (25.493)	depp (30.275)	eusocial (25.001)
3	role (23.006)	penalty (25.04)	bees (27.679)	world (25.414)	court (20.771)	females (24.24)
4	series (19.563)	players (24.063)	bee (26.936)	cup (24.925)	divorce (17.289)	solitary (21.173)
5	burton (18.694)	referee (23.649)	food (23.44)	association (22.331)	sued (16.105)	nest (20.198)
6	played (17.583)	team (22.9)	flowers (22.374)	national (20.958)	stated (15.984)	males (18.3)
7	character (16.646)	goal (22.859)	pollination (18.09)	women (20.668)	alcohol (15.238)	workers (17.16)
8	success (16.41)	player (22.054)	larvae (17.73)	international (20.16)	stating (15.199)	typically (16.886)
9	films (15.74)	play (21.774)	pollen (17.666)	tournament (18.26)	paradis (14.98)	colonies (16.528)
10	box (15.024)	game (20.471)	predators (17.634)	uefa (18.029)	alleged (14.971)	queens (16.427)

Table A.17: For the most significant two words per topic, the four nearest neighbors based on cosine similarity are listed.

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6
0	film (1.0)	ball (1.0)	honey (1.0)	football (1.0)	heard (1.0)	species (1.0)
1	avril (1.0)	officials (1.0)	triangulum (1.0)	entered (1.0)	obtained (1.0)	progressive (1.0)
2	office (1.0)	invoke (1.0)	consumption (1.0)	most (1.0)	countersued (1.0)	halictidae (1.0)
3	landau (1.0)	heading (1.0)	copper (1.0)	excess (1.0)	depths (1.0)	temperate (1.0)
4	chamberlain (1.0)	twohalves (1.0)	might (1.0)	uk (1.0)	mismanagement (1.0)	spring (1.0)
0	starred (1.0)	may (1.0)	insects (1.0)	fifa (1.0)	depp (1.0)	eusocial (1.0)
1	raimi (1.0)	noninternational (0.992)	blooms (1.0)	oceania (1.0)	city (1.0)	unfertilized (1.0)
2	candidate (1.0)	red (0.991)	eats (1.0)	sudamericana (1.0)	tribute (1.0)	females (1.0)
3	hardwicke (1.0)	required (0.991)	catching (1.0)	widened (1.0)	mick (1.0)	paper (1.0)
4	peter (1.0)	yd (0.989)	disease (1.0)	oversee (1.0)	elvis (1.0)	hibernate (1.0)

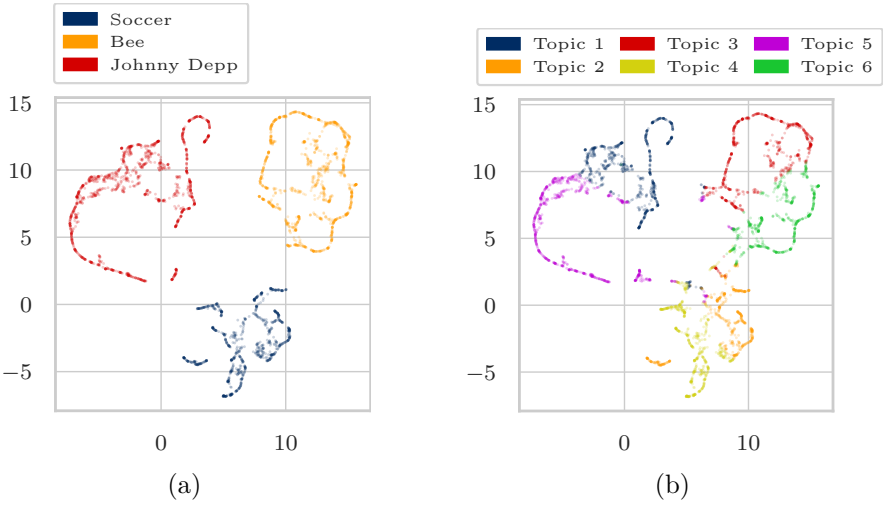


Figure A.11: (a) 2-dimensional representation of word embeddings $\boldsymbol{H}$ colored by topic assignment. (b) 2-dimensional representation of word embeddings $\boldsymbol{H}$ colored by original Wikipedia article assignment (words that occur in more than one article are excluded).

A.2.3 Amazon Reviews – Tensor Input

As described in Section 3.4.1, our Amazon movie review corpus comprises human-written reviews for six famous animation films. Factorizing its PPMI tensor representation with non-negative tensor DEDICOM and the number of topics set to $k = 10$ reveals not only movie-specific sub-topics but also general topics that span over several movies. For example, Topics 1, 9, and 10 in Table A.18 can uniquely be related to the films "Frozen", "Toy Story 1", and "Kung Fu Panda 1", respectively, whereas Topic 5 constitutes bonus material on a DVD which holds true for all films. The latter can also be seen in Figure A.12 where Topic 5 is highlighted in each movie slice (strongly in the top and lightly in the bottom row). In the same sense, one can observe that Topic 3 is present in both, "Kung Fu Panda 1", and "Kung Fu Panda 2", which is reasonable considering the topic depicts the general notion of a fearsome warrior.

DEDICOM Mulitplicative Update Rules

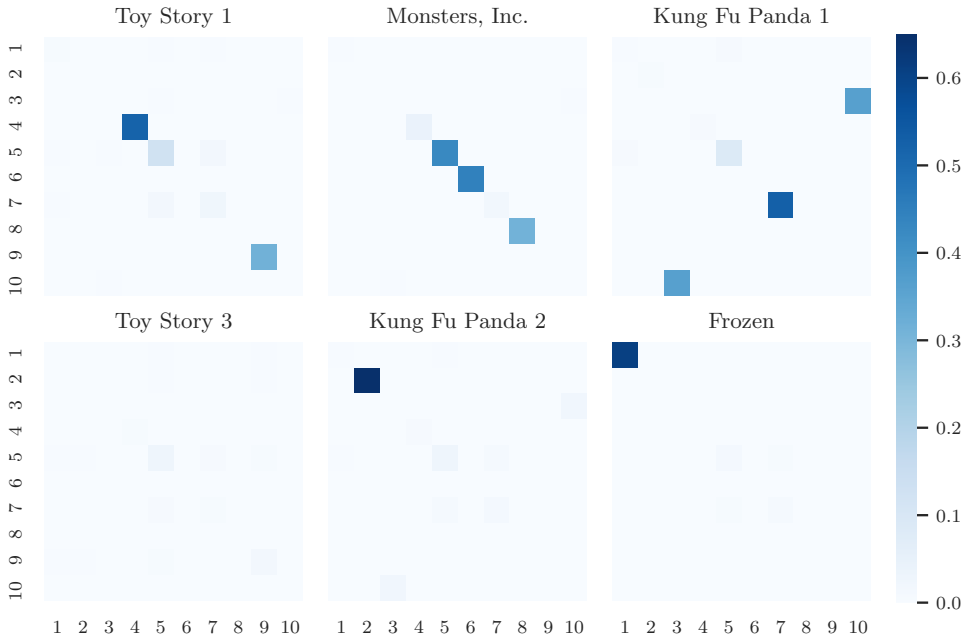


Figure A.12: Colored heatmap of affinity tensor $\underline{\boldsymbol{R}}$, trained on the Amazon review data represented as input tensor using multiplicative update rules.

Table A.18: Top 10 representative words per dimension of the basis matrix $\boldsymbol{A}$, trained on the Amazon review data as input tensor using multiplicative update rules.

	Topic 1 #528	Topic 2 #445	Topic 3 #1477	Topic 4 #1790	Topic 5 #1917	Topic 6 #597	Topic 7 #789	Topic 8 #670	Topic 9 #1599	Topic 10 #188
1	anna (4.215)	shen (4.21)	legendary (1.459)	lasseter (2.887)	disc (1.367)	screams (3.779)	code (3.292)	mike (4.325)	woody (6.12)	po (5.737)
2	elsa (4.087)	peacock (2.668)	valley (1.448)	director (2.392)	birds (1.343)	energy (3.315)	email (2.781)	crystal (4.055)	buzz (5.484)	master (5.276)
3	olaf (2.315)	oldman (2.627)	temple (1.31)	andrew (2.158)	widescreen (1.327)	monstropolis (3.13)	promo (2.645)	billy (3.911)	andy (4.355)	shifu (4.707)
4	trolls (2.241)	gary (2.423)	kim (1.307)	stanton (2.119)	outtakes (1.238)	world (3.109)	free (2.343)	goodman (3.812)	toys (4.119)	dragon (4.344)
5	frozen (2.196)	lord (2.201)	fearsome (1.288)	special (1.892)	extras (1.185)	monsters (3.047)	promotion (2.279)	sully (3.728)	lightyear (3.334)	warrior (4.274)
6	kristoff (2.155)	weapon (1.469)	teacher (1.288)	pete (1.612)	dvd (1.142)	city (2.994)	promotional (2.266)	wazowski (3.513)	allen (2.752)	tai (4.082)
7	queen (2.055)	wolf (1.405)	battle (1.264)	ranft (1.564)	included (1.13)	monster (2.978)	amazon (2.129)	randall (3.404)	tim (2.609)	lung (3.993)
8	hans (2.054)	inner (1.38)	duk (1.257)	joe (1.564)	short (1.101)	power (2.919)	click (2.024)	sulley (3.396)	hanks (2.471)	five (2.952)
9	sister (1.904)	yeoh (1.359)	train (1.221)	feature (1.555)	games (1.1)	scare (2.614)	download (1.889)	buscemi (3.266)	cowboy (2.407)	oogway (2.918)
10	ice (1.839)	michelle (1.354)	warriors (1.22)	ralph (1.518)	tour (1.031)	closet (2.555)	instructions (1.877)	james (3.264)	space (2.375)	furious (2.806)

Table A.19: For the most significant two words per topic, the four nearest neighbors based on cosine similarity are listed.

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
0	anna (1.0)	shen (1.0)	legendary (1.0)	lasseter (1.0)	disc (1.0)	screams (1.0)	code (1.0)	mike (1.0)	woody (1.0)	po (1.0)
1	christoph (1.0)	canons (1.0)	sacred (0.999)	andrew (1.0)	thxcertified (1.0)	harvested (1.0)	confirm (1.0)	bogg (1.0)	rips (1.0)	panda (0.985)
2	readiness (1.0)	yeoh (1.0)	fulfill (0.999)	stanton (1.0)	presentation (0.999)	speciallytrained (1.0)	discount (1.0)	chased (0.996)	spy (1.0)	fu (0.983)
3	carrots (1.0)	wolf (1.0)	roster (0.999)	eggleston (1.0)	upgrade (0.999)	screamprocessing (1.0)	browser (1.0)	flair (0.996)	josie (1.0)	black (0.983)
4	poverty (1.0)	weapon (1.0)	megafan (0.999)	uncredited (1.0)	featurettes (0.998)	corporation (1.0)	popup (1.0)	slot (0.995)	supurb (1.0)	kung (0.981)
0	elsa (1.0)	peacock (1.0)	valley (1.0)	director (1.0)	birds (1.0)	energy (1.0)	email (1.0)	crystal (1.0)	buzz (1.0)	master (1.0)
1	shipwreck (1.0)	shen (1.0)	praying (0.999)	producer (0.998)	pressed (1.0)	powered (0.998)	confirm (1.0)	oz (1.0)	hist (1.0)	shifu (0.999)
2	marriage (1.0)	mcbride (1.0)	kim (0.997)	teaser (0.997)	gadget (1.0)	frightened (0.997)	code (1.0)	mae (1.0)	wayne (1.0)	warrior (0.999)
3	idena (1.0)	yeoh (1.0)	chorgum (0.997)	globes (0.997)	classically (1.0)	screams (0.996)	fwiw (1.0)	celia (1.0)	reunited (1.0)	dragon (0.998)
4	prodding (1.0)	michelle (1.0)	preying (0.997)	rousing (0.997)	starz (1.0)	scarry (0.996)	android (1.0)	cristal (1.0)	hockey (1.0)	martial (0.994)

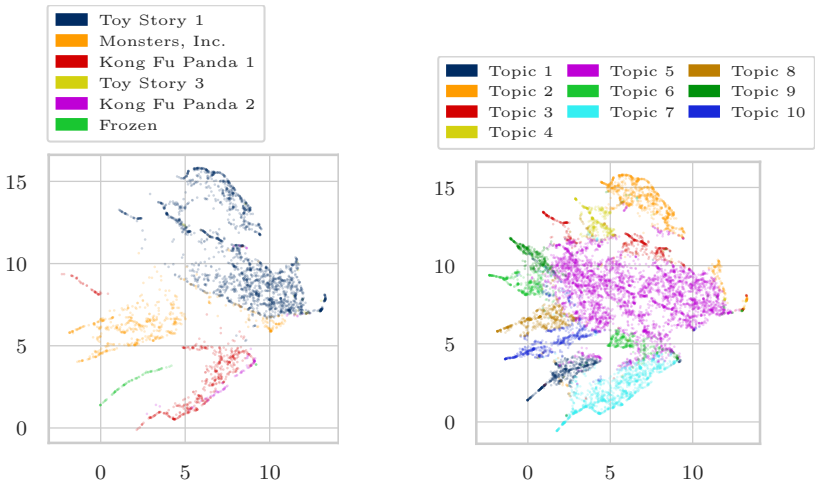


Figure A.13: (a) 2-dimensional representation of word embeddings $\boldsymbol{A}$ colored by topic assignment. (b) 2-dimensional representation of word embeddings $\boldsymbol{A}$ colored by original review article.

Tensor NMF

Table A.20: Each column lists the top 10 representative words per dimension of the basis matrix $\boldsymbol{H}$.

	Topic 1 #590	Topic 2 #1052	Topic 3 #456	Topic 4 #350	Topic 5 #4069	Topic 6 #733	Topic 7 #1140	Topic 8 #582	Topic 9 #423	Topic 10 #605
1	anna (109.81)	woody (134.366)	director (100.622)	allen (83.875)	widescreen (34.484)	code (88.94)	master (89.686)	mike (88.628)	film (58.514)	screams (87.782)
2	elsa (106.148)	buzz (120.93)	lasseter (93.134)	hanks (77.313)	outtakes (30.724)	email (73.645)	po (85.79)	crystal (82.472)	animation (53.628)	energy (78.113)
3	olaf (59.353)	andy (105.728)	andrew (81.452)	tim (75.511)	disc (30.688)	promo (67.483)	shifu (82.465)	billy (79.244)	characters (46.861)	world (73.888)
4	trolls (58.811)	toys (98.523)	stanton (80.132)	rickles (74.217)	extras (30.09)	amazon (64.978)	warrior (75.609)	goodman (76.831)	films (44.937)	monstropolis (73.484)
5	kristoff (56.309)	lightyear (68.336)	john (73.004)	tom (72.401)	versions (27.894)	promotion (58.631)	dragon (74.235)	sully (75.612)	pixar (44.873)	monsters (71.721)
6	hans (55.628)	sid (52.34)	pete (70.556)	jim (69.776)	included (27.455)	free (58.207)	tai (71.721)	wazowski (71.588)	even (44.176)	city (71.352)
7	frozen (54.257)	cowboy (48.588)	docter (64.734)	varney (66.053)	material (26.887)	promotional (57.738)	lung (70.786)	randall (69.695)	animated (43.492)	power (70.642)
8	queen (53.956)	space (47.88)	ralph (54.884)	slinky (62.326)	edition (26.546)	click (55.373)	furious (63.232)	sulley (69.604)	also (43.484)	monster (70.197)
9	sister (52.749)	room (42.655)	joe (53.7)	potato (62.237)	contains (25.386)	download (50.788)	oogway (60.879)	james (68.574)	dvd (42.736)	closet (61.451)
10	ice (49.71)	toy (42.042)	ranft (53.41)	mr (61.801)	extra (25.144)	purchase (50.327)	five (59.259)	buscemi (66.028)	well (40.124)	scare (61.243)

Table A.21: For the most significant two words per topic, the four nearest neighbors based on cosine similarity are listed.

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
0	anna (1.0)	woody (1.0)	director (1.0)	allen (1.0)	widescreen (1.0)	code (1.0)	master (1.0)	mike (1.0)	film (1.0)	screams (1.0)
1	marriage (1.0)	acciently (1.0)	producer (1.0)	trustworthy (1.0)	benefactors (1.0)	card (1.0)	furious (1.0)	longtime (0.995)	films (1.0)	electrical (1.0)
2	trolls (1.0)	limp (1.0)	jackson (1.0)	arguments (1.0)	pioneers (1.0)	confirmation (1.0)	dragon (1.0)	cyclops (0.995)	first (0.994)	screamprocessing (1.0)
3	flees (1.0)	jealousey (0.999)	rabson (1.0)	hanks (1.0)	keepcase (1.0)	assuming (1.0)	shifu (1.0)	slot (0.995)	animated (0.993)	chlid (1.0)
4	christian (1.0)	swells (0.999)	composer (1.0)	knowitall (1.0)	redone (1.0)	android (1.0)	warrior (1.0)	humanlike (0.994)	animation (0.989)	shortage (1.0)
0	elsa (1.0)	buzz (1.0)	lasseter (1.0)	hanks (1.0)	outtakes (1.0)	email (1.0)	po (1.0)	crystal (1.0)	animation (1.0)	energy (1.0)
1	marriage (1.0)	recive (0.999)	nathan (1.0)	trustworthy (1.0)	storyboarding (0.993)	promo (1.0)	martial (0.998)	billy (1.0)	film (0.989)	supply (1.0)
2	heals (1.0)	zorg (0.999)	officer (1.0)	allen (1.0)	informative (0.991)	avail (1.0)	fight (0.997)	talkative (0.999)	story (0.987)	powered (1.0)
3	marrying (1.0)	limp (0.997)	cunningham (1.0)	tom (1.0)	contents (0.99)	flixster (1.0)	arts (0.996)	competitor (0.999)	scenes (0.987)	collect (1.0)
4	feminist (1.0)	acciently (0.997)	derryberry (1.0)	arguments (1.0)	logo (0.99)	confirming (1.0)	adopted (0.995)	devilishly (0.999)	also (0.984)	screams (0.999)

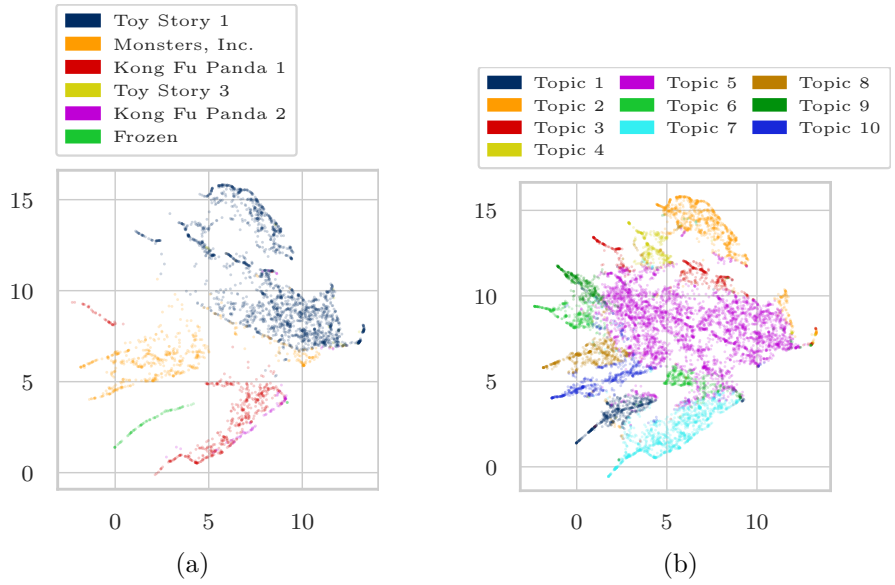


Figure A.14: (a) 2-dimensional representation of word embeddings $\boldsymbol{H}$ colored by topic assignment. (b) 2-dimensional representation of word embeddings $\boldsymbol{H}$ colored by original review article.

A.2.4 New York Times News – Tensor Input

DEDICOM Mulitplicative Update Rules

Table A.22: For the most significant two words per topic, the four nearest neighbors based on cosine similarity are listed.

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
0	suleimani (1.0)	loans (1.0)	masks (1.0)	floyd (1.0)	contributed (1.0)	confederate (1.0)	ukraine (1.0)	storm (1.0)	restaurants (1.0)	weinstein (1.0)
1	qassim (1.0)	spend (1.0)	sanitizer (1.0)	brutality (1.0)	alan (1.0)	statue (1.0)	lutsenko (1.0)	storms (1.0)	salons (1.0)	raped (1.0)
2	iran (1.0)	smallbusiness (1.0)	wipes (1.0)	police (1.0)	edmondson (1.0)	monuments (1.0)	ukrainians (1.0)	isaias (1.0)	cafes (1.0)	predatory (1.0)
3	iranian (1.0)	rent (1.0)	cloth (1.0)	systemic (1.0)	mervosh (1.0)	statues (1.0)	yovanovitch (1.0)	landfall (1.0)	pubs (1.0)	mann (1.0)
4	militias (1.0)	incentives (1.0)	homemade (1.0)	knee (1.0)	emily (1.0)	honoring (1.0)	burisma (1.0)	forecasters (1.0)	nightclubs (1.0)	sciorra (1.0)
0	iran (1.0)	university (1.0)	protective (1.0)	minneapolis (1.0)	reporting (1.0)	statue (1.0)	sondland (1.0)	hurricane (1.0)	bars (1.0)	sexual (1.0)
1	qassim (1.0)	jerome (0.999)	gowns (1.0)	breonna (1.0)	rabin (1.0)	monuments (1.0)	zelensky (1.0)	bahamas (1.0)	dining (1.0)	rape (1.0)
2	suleimani (1.0)	oxford (0.998)	ventilators (1.0)	kueng (1.0)	contributed (1.0)	statues (1.0)	volker (1.0)	hurricanes (1.0)	theaters (1.0)	metoo (1.0)
3	iranian (1.0)	columbia (0.998)	respirators (1.0)	floyd (1.0)	keith (1.0)	confederate (1.0)	giuliani (1.0)	forecasters (1.0)	venues (1.0)	sexually (1.0)
4	militias (1.0)	economics (0.998)	supplies (1.0)	police (1.0)	chokshi (1.0)	honoring (1.0)	quid (1.0)	landfall (1.0)	malls (1.0)	mann (1.0)

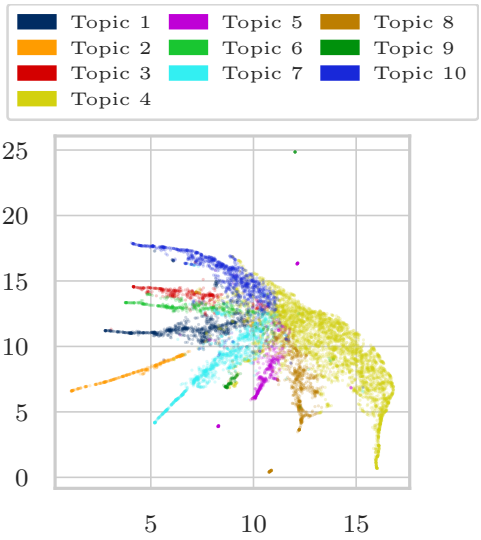


Figure A.15: 2-dimensional representation of word embeddings $\boldsymbol{A}$ colored by topic assignment.

Tensor NMF

Table A.23: Each column lists the top 10 representative words per dimension of the basis matrix $\boldsymbol{A}$.

	Topic 1 #977	Topic 2 #360	Topic 3 #420	Topic 4 #4192	Topic 5 #489	Topic 6 #405	Topic 7 #1135	Topic 8 #748	Topic 9 #108	Topic 10 #1166
1	floyd (82.861)	contributed (137.058)	iran (80.007)	masks (40.463)	ship (87.178)	syria (94.686)	senator (77.285)	restaurants (93.511)	bloom (110.77)	ukraine (79.611)
2	police (64.649)	reporting (84.889)	suleimani (78.78)	patients (34.77)	crew (70.694)	syrian (82.565)	storm (43.439)	bars (64.889)	julie (103.282)	sondland (61.099)
3	protesters (63.588)	michael (76.156)	iranian (72.581)	ventilators (34.299)	aboard (67.464)	kurdish (82.013)	hurricane (42.213)	reopen (57.541)	edited (100.159)	testimony (49.982)
4	minneapolis (63.216)	katie (63.146)	iraq (63.27)	protective (33.719)	passengers (65.895)	turkey (80.374)	iowa (41.985)	stores (55.435)	los (95.747)	testified (49.959)
5	protests (61.585)	emily (60.696)	gen (50.966)	loans (28.178)	cruise (63.535)	turkish (75.912)	republican (40.993)	gyms (50.487)	graduated (93.51)	zelensky (48.427)
6	george (53.378)	alan (59.499)	strike (49.026)	supplies (27.15)	princess (45.714)	kurds (63.639)	gov (37.689)	theaters (49.866)	angeles (92.349)	ambassador (46.086)
7	brutality (44.051)	nicholas (55.899)	iraqi (46.027)	gloves (26.724)	flight (45.375)	fighters (62.313)	buttigieg (37.1)	closed (46.544)	berkeley (85.653)	weinstein (45.053)
8	officers (43.581)	cochrane (52.045)	qassim (45.861)	equipment (26.252)	nasa (43.306)	forces (57.659)	democrat (37.087)	indoor (44.541)	grew (84.145)	ukrainian (43.716)
9	racism (43.457)	ben (41.414)	maj (44.921)	respiratory (25.447)	navy (40.396)	troops (54.045)	representative (35.807)	salons (41.99)	today (41.818)	giuliani (42.674)
10	demonstrations (42.547)	maggie (41.328)	baghdad (44.867)	testing (24.179)	astronauts (37.073)	isis (53.743)	bernie (35.228)	shops (40.311)	california (38.807)	sexual (39.966)

Table A.24: For the most significant two words per topic, the four nearest neighbors based on cosine similarity are listed.

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
0	floyd (1.0)	contributed (1.0)	iran (1.0)	masks (1.0)	ship (1.0)	syria (1.0)	senator (1.0)	restaurants (1.0)	bloom (1.0)	ukraine (1.0)
1	demonstrations (1.0)	shear (1.0)	suleimani (1.0)	providers (1.0)	aboard (1.0)	isis (1.0)	wyden (1.0)	shops (1.0)	graduated (1.0)	volker (1.0)
2	systemic (1.0)	annie (1.0)	retaliation (1.0)	distressed (1.0)	capsule (1.0)	ceasefire (0.999)	iowa (1.0)	takeout (1.0)	edited (1.0)	inquiry (1.0)
3	protests (1.0)	mazzei (1.0)	qassim (1.0)	tobacco (1.0)	diamond (1.0)	fighters (0.999)	steyer (1.0)	nightclubs (1.0)	berkeley (1.0)	transcript (1.0)
4	defund (1.0)	kitty (1.0)	revenge (1.0)	selfemployed (1.0)	dragon (1.0)	syrian (0.999)	klobuchar (1.0)	pubs (1.0)	grew (1.0)	investigations (1.0)
0	police (1.0)	reporting (1.0)	suleimani (1.0)	patients (1.0)	crew (1.0)	syrian (1.0)	storm (1.0)	bars (1.0)	julie (1.0)	sondland (1.0)
1	systemic (1.0)	luis (1.0)	strike (1.0)	treating (1.0)	aboard (1.0)	alassad (1.0)	carolina (1.0)	reopen (1.0)	garcetti (0.993)	testifying (1.0)
2	peaceful (1.0)	beachy (1.0)	maj (1.0)	infection (1.0)	capsule (1.0)	recep (1.0)	rubio (1.0)	nonessential (1.0)	graduated (0.988)	mick (1.0)
3	peacefully (1.0)	kaplan (1.0)	iran (1.0)	develop (1.0)	princess (1.0)	erdogan (1.0)	hampshire (1.0)	nail (1.0)	edited (0.988)	quid (1.0)
4	knee (1.0)	glueck (1.0)	retaliation (1.0)	repay (1.0)	cruise (1.0)	kurds (1.0)	landfall (1.0)	takeout (1.0)	berkeley (0.988)	impeachment (1.0)

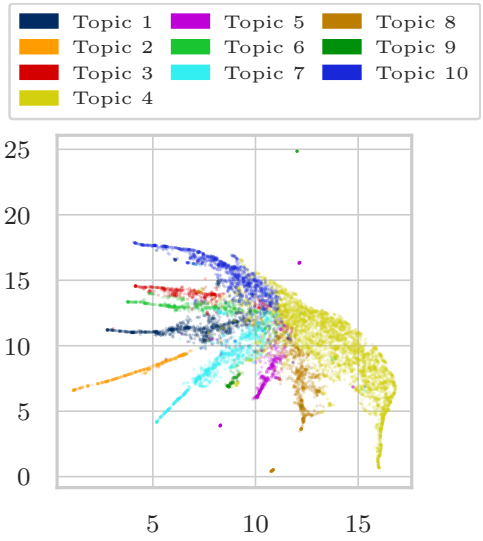


Figure A.16: 2-dimensional representation of word embeddings $\boldsymbol{H}$ colored by topic assignment.

APPENDIX B

Large Language Models for Compliance Verification

B.1 Prompt evaluation

Table B.1 shows the detailed evaluation of all prompt configurations per dataset and model based on the micro F_1 score.

Table B.1: Micro F_1-Scores for all models per dataset and prompt configuration. Note: Due to the verbose nature of Llama-2 Models and their poor performance in the German language, Llama-2 was incapable of generating any machine-readable consistent outputs that are interpretable with a heuristic for some prompt formats. This leads to some Micro F_1-Scores being 0.

Prompt	Dataset	GPT-3.5	GPT-3.5-16K	GPT-4	Llama-2-7b	Llama-2-13b	Llama-2-70b
I	HGB	45.88	45.88	**66.67**	49.23	0.00	0.00
	IFRS	76.42	**77.21**	73.60	58.58	24.57	42.58
II	HGB	35.49	35.10	40.07	**46.89**	36.20	13.68
	IFRS	69.69	10.32	**77.05**	33.13	52.83	70.04
III	HGB	43.44	43.44	**66.67**	36.20	0.00	24.62
	IFRS	58.22	57.22	**71.73**	29.74	22.59	50.69
IV	HGB	37.87	37.87	41.03	43.44	43.44	**49.23**
	IFRS	74.27	11.03	**75.23**	35.83	56.49	66.57
V	HGB	43.96	33.57	**53.85**	38.46	0.00	13.68
	IFRS	**72.00**	60.77	71.37	35.68	65.41	69.65
VI	HGB	37.87	37.87	**75.60**	0.00	46.89	43.08
	IFRS	77.56	**77.58**	66.38	68.02	65.58	70.00
VII	HGB	43.96	43.96	**63.95**	13.68	24.62	0.00
	IFRS	70.04	70.04	**70.13**	65.82	61.62	24.76
VIII	HGB	46.15	46.15	**66.67**	0.00	47.34	0.00
	IFRS	35.21	35.21	**67.74**	53.69	35.02	29.87

B.2 Prompt configurations

The following features the list of evaluated prompts as outlined in Section 9.4.2. First, the prompts used for reports written under the IFRS are shown. Second, prompts used for German reports written for the HGB are listed.

B.2.1 IFRS Prompts

I In-Out-Sub-Template

System: You are an expert auditor with perfect knowledge of the IFRS accounting standard. You always answer truthfully whether a given regulatory requirement is fully complied in the following line ids.

Is the following IFRS sub-requirement fully complied in the following input document?

Answer with "yes", if the sub-requirement is fully complied.
Answer with "no", if it is not fully complied.

Format your output complying to the following json schema:
{{"answer": <"yes"|"no">}}

requirement: "{requirement}"
document: "{document}"

II Cot-Sub-Template

System: You are an expert auditor with perfect knowledge of the IFRS accounting standard. You always answer truthfully whether a given regulatory requirement is fully complied in the following line ids.

Is the following IFRS sub-requirement fully complied in the following input document? Think step by step: Explain whether it is fully complied and reference relevant line ids from the input document. Based on your explanation, determine whether the requirement is fully complied by answering with "yes" or "no".

requirement: "{requirement}"
document: "{document}"

III In-Out-Template

System: You are an expert auditor with perfect knowledge of the IFRS accounting standard. You always answer truthfully whether a given regulatory requirement is fully complied in the following line ids.

Answer with "yes", if all sub-requirements are fully complied.
Answer with "no", if at least one of the sub-requirements is not fully complied.

Format your output complying to the following json schema:
{{"answer": <"yes"|"no">}}

requirement: "{requirement}"
document: "{document}"

IV Cot-Template

System: You are an expert auditor with perfect knowledge of the IFRS accounting standard. You always answer truthfully whether a given regulatory requirement is fully complied in the following line ids.

Think step by step: Explain for each sub-requirement whether it is fully complied and reference relevant line ids from the input document in each explanation. Based on your explanations, determine whether the overall requirement is fully complied by answering with "yes" or "no".

requirement: "{requirement}"
document: "{document}"

V In-Out-Tot-Template

System: Imagine three different experts in the field of auditing answering this question. Each expert has perfect knowledge of the IFRS accounting standard. All experts always answer truthfully whether a particular regulatory requirement in the following line numbers is is completely fulfilled.

Each expert writes down 1 step of their thought process and shares it with the group. Then all experts move to the next step, and so on. Show each step and each expert's thinking process. If at any time an expert realizes he is wrong, he is eliminated.

Answer with "yes", if all sub-requirements are fully complied.
Answer with "no", if at least one of the sub-requirements is not fully complied.

Format your output complying to the following json schema:
{{"answer": <"yes"|"no">}}

requirement: "{requirement}"
document: "{document}"

VI In-Out-Tot-One-Shot-Template

System: Imagine three different experts in the field of auditing answering this question. Each expert has perfect knowledge of the IFRS accounting standard. All experts always answer truthfully whether a particular regulatory requirement in the following line numbers is is completely fulfilled.

Each expert writes down 1 step of their thought process and shares it with the group. Then all experts move to the next step, and so on. Show each step and each expert's thinking process. If at any time an expert realizes he is wrong, he is eliminated.

Answer with "yes", if all sub-requirements are fully complied.
Answer with "no", if at least one of the sub-requirements is not fully complied.

Format your output complying to the following json schema:
{{"answer": <"yes"|"no">}}

requirement: "{requirement}"
document: "{document}"

Example:

Requirement: “Disclose the amount of receivables with a remaining maturity of more than one year; separately for each item reported.” Document: “544: All receivables have a remaining maturity of less than one year.”

Expert 1: “The requirement asks for the amount of receivables with a remaining maturity of more than one year. However, line 544 states that all receivables have a remaining maturity of less than one year. This seems to imply that there are no receivables with a remaining maturity of more than one year and the requirement has been met.”

Expert 2: “I see Expert 1’s point. Since the document indicates that all receivables have a remaining term of less than one year, we can assume that the requirement has been met.”

Expert 3: “I agree with my colleagues. Although the document does not provide any other specific information, we can conclude from the context that the requirement has been met.”

{{“answer”: “yes”}}

VII In-Out-One-Shot-Template

System: You are an expert auditor with perfect knowledge of the IFRS accounting standard. You always answer truthfully whether a given regulatory requirement is fully complied in the following line ids.

Answer with “yes”, if all sub-requirements are fully complied.
Answer with ”no”, if at least one of the sub-requirements is not fully complied.

Format your output complying to the following json schema:
{{“answer”: <“yes”|“no”>}}

requirement: “{requirement}”
document: “{document}”

Example:

Requirement: “Disclose the amount of receivables due in more than one year; separately for each item shown.” Document: “544: All receivables have a remaining maturity of less than one year.”

{{“answer”: “yes”}}

VIII In-Out-One-Shot-No-Template

System: You are an expert auditor with perfect knowledge of the IFRS accounting standard. You always answer truthfully whether a given regulatory requirement is fully complied in the following line ids.

Answer with “yes”, if all sub-requirements are fully complied.
Answer with “no”, if at least one of the sub-requirements is not fully complied.

Format your output complying to the following json schema:
{{“answer”: <“yes”|“no”>}}

requirement: “{requirement}”
document: “{document}”

Example:

Requirement: “Disclose the amount of receivables due in less than one year; separately for each item shown.” Document: “544: All receivables have a remaining maturity of less than one year.”

{{“answer”: “no”}}

B.2.2 HGB Prompts (German)

I In-Out-Sub-Template

System: Sie sind ein Experte im Bereich Wirtschaftsprüfung und haben perfekte Kenntnisse des HGB-Bilanzierungstandards. Sie antworten immer wahrheitsgemäß, ob eine bestimmte behördliche Anforderung in den folgenden Zeilennummern vollständig erfüllt ist.

Ist die folgende HGB-Anforderung im unten genannten Dokument vollständig erfüllt?
Antworten Sie mit:
- “yes”, wenn die Anforderung erfüllt ist,
- “no”, wenn die Anforderung nicht erfüllt ist,
- “unclear”, wenn die Erfüllung der Anforderung nicht beantwortet werden kann, da Kontextinformationen fehlen und
- “not applicable”, wenn die Anforderung für das Dokument nicht relevant ist.

Formatieren Sie Ihre Ausgabe gemäß dem folgenden JSON-Schema:
{{“answer”: <“yes”|“no”|“unclear”|“not applicable”>}}

Anforderung: “{`requirement`}”
Dokument: “{`document`}”

II Cot-Sub-Template

System: Sie sind ein Experte im Bereich Wirtschaftsprüfung und haben perfekte Kenntnisse des HGB-Bilanzierungstandards. Sie antworten immer wahrheitsgemäß, ob eine bestimmte behördliche Anforderung in den folgenden Zeilennummern vollständig erfüllt ist.

Ist die folgende HGB-Teilanforderung im gegebenen Eingangsdokument vollständig erfüllt? Denke Schritt für Schritt: Erklären Sie, ob sie vollständig erfüllt ist, und geben Sie die relevanten Zeilennummern aus dem vorhanden Dokument an. Basierend auf Ihrer Erklärung bestimmen Sie, ob die Anforderung vollständig erfüllt ist.
Beenden Sie ihre Antwort mit
- “yes”, wenn die Anforderung erfüllt ist,
- “no”, wenn die Anforderung nicht erfüllt ist,
- “unclear”, wenn die Erfüllung der Anforderung nicht beantwortet werden kann, da Kontextinformationen fehlen und
- “not applicable”, wenn die Anforderung für das Dokument nicht relevant ist.

Anforderung: “{`requirement`}”
Dokument: “{`document`}”

III In-Out-Template

System: Sie sind ein Experte im Bereich Wirtschaftsprüfung und haben perfekte Kenntnisse des HGB-Bilanzierungstandards. Sie antworten immer wahrheitsgemäß, ob eine bestimmte behördliche Anforderung in den folgenden Zeilennummern vollständig erfüllt ist.

Antworten Sie mit:
- “yes”, wenn die Anforderung erfüllt ist,
- “no”, wenn die Anforderung nicht erfüllt ist,
- “unclear”, wenn die Erfüllung der Anforderung nicht beantwortet werden kann, da Kontextinformationen fehlen und
- “not applicable”, wenn die Anforderung für das Dokument nicht relevant ist.

Formatieren Sie Ihre Ausgabe gemäß dem folgenden JSON-Schema:
{{“answer”: <“yes”|“no”|“unclear”|“not applicable”>}}

Anforderung: “{`requirement`}”
Dokument: “{`document`}”

IV Cot-Template

System: Sie sind ein Experte im Bereich Wirtschaftsprüfung und haben perfekte Kenntnisse des HGB-Bilanzierungstandards. Sie antworten immer wahrheitsgemäß, ob eine bestimmte behördliche Anforderung in den folgenden Zeilennummern vollständig erfüllt ist.

Denke Schritt für Schritt: Erkläre für jede Teilanforderung, ob sie vollständig erfüllt ist, und verweise in jeder Erklärung auf die relevanten Zeilennummern aus dem gegebenen Dokument. Basierend auf deinen Erklärungen entscheide, ob jede Teilanforderung vollständig erfüllt ist. Beenden Sie ihre Antwort mit
- “yes”, wenn die Anforderung erfüllt ist,
- “no”, wenn die Anforderung nicht erfüllt ist,
- “unclear”, wenn die Erfüllung der Anforderung nicht beantwortet werden kann, da Kontextinformationen fehlen und
- “not applicable”, wenn die Anforderung für das Dokument nicht relevant ist.

Anforderung: “{`requirement`}”
Dokument: “{`document`}”

V In-Out-Tot-Template

System: Stellen Sie sich vor, drei verschiedene Experten im Bereich Wirtschaftsprüfung beantworten diese Frage. Jeder Experte hat perfekte Kenntnisse des HGB-Bilanzierungstandards. Alle Experten antworten immer wahrheitsgemäß, ob eine bestimmte behördliche Anforderung in den folgenden Zeilennummern vollständig erfüllt ist.

Jeder Experte schreibt 1 Schritt seines Denkprozesses nieder und teilt ihn mit der Gruppe. Dann gehen alle Experten zum nächsten Schritt über, usw. Zeige jeden Schritt und den Denkprozess jedes Experten. Wenn ein Experte zu irgendeinem Zeitpunkt feststellt, dass er falsch liegt, scheidet er aus.

Geben Sie bei einer Mehrheitsabstimmung unter den Experten nur eine Antwort in diesem Format zurück:
- “yes”, wenn die Anforderung erfüllt ist,

- “no”, wenn die Anforderung nicht erfüllt ist,
- “unclear”, wenn die Erfüllung der Anforderung nicht beantwortet werden kann, da Kontextinformationen fehlen und
- “not applicable”, wenn die Anforderung für das Dokument nicht relevant ist.

Formatieren Sie Ihre Ausgabe gemäß dem folgenden JSON-Schema:
{{“answer”: <“yes”|“no”|“unclear”|“not applicable”>}}

Anforderung: “{requirement}”
Dokument: “{document}”

VI In-Out-Tot-One-Shot-Template

System: Stellen Sie sich vor, drei verschiedene Experten im Bereich Wirtschaftsprüfung beantworten diese Frage. Jeder Experte hat perfekte Kenntnisse des HGB-Bilanzierungstandards. Alle Experten antworten immer wahrheitsgemäß, ob eine bestimmte behördliche Anforderung in den folgenden Zeilennummern vollständig erfüllt ist.

Jeder Experte schreibt 1 Schritt seines Denkprozesses nieder und teilt ihn mit der Gruppe. Dann gehen alle Experten zum nächsten Schritt über, usw. Zeige jeden Schritt und den Denkprozess jedes Experten. Wenn ein Experte zu irgendeinem Zeitpunkt feststellt, dass er falsch liegt, scheidet er aus.

Geben Sie bei einer Mehrheitsabstimmung unter den Experten nur eine Antwort in diesem Format zurück:
- “yes”, wenn die Anforderung erfüllt ist,
- “no”, wenn die Anforderung nicht erfüllt ist,
- “unclear”, wenn die Erfüllung der Anforderung nicht beantwortet werden kann, da Kontextinformationen fehlen und
- “not applicable”, wenn die Anforderung für das Dokument nicht relevant ist.

Formatieren Sie Ihre Ausgabe gemäß dem folgenden JSON-Schema:
{{“answer”: <“yes”|“no”|“unclear”|“not applicable”>}}

Anforderung: “{requirement}”
Dokument: “{document}”

Example:

Anforderung: “Angabe des Betrags der Forderungen mit einer Restlaufzeit von mehr als einem Jahr; gesondert für jeden ausgewiesenen Posten” Dokument: “544: Sämtliche Forderungen haben eine Restlaufzeit von weniger als einem Jahr.”

Experte 1: “Die Anforderung verlangt die Angabe des Betrags der Forderungen mit einer Restlaufzeit von mehr als einem Jahr. In Zeile 544 wird jedoch angegeben, dass alle Forderungen eine Restlaufzeit von weniger als einem Jahr haben. Das scheint zu implizieren, dass es keine Forderungen mit einer Restlaufzeit von mehr als einem Jahr gibt und die Anforderung erfüllt worden ist.”

Experte 2: “Ich sehe den Punkt von Experte 1. Da das Dokument angibt, dass alle Forderungen eine Restlaufzeit von weniger als einem Jahr haben, können wir davon ausgehen, dass die Anforderung erfüllt worden ist.”

Experte 3: “Ich stimme meinen Kollegen zu. Obwohl das Dokument keine weiteren spezifischen Informationen enthält, können wir aus dem Kontext schließen, dass die Anforderung erfüllt worden

ist."

{{"answer": "yes"}}

VII In-Out-One-Shot-Template

System: Sie sind ein Experte im Bereich Wirtschaftsprüfung und haben perfekte Kenntnisse des HGB-Bilanzierungstandards. Sie antworten immer wahrheitsgemäß, ob eine bestimmte behördliche Anforderung in den folgenden Zeilennummern vollständig erfüllt ist.

Antworten Sie mit:
- "yes", wenn die Anforderung erfüllt ist,
- "no", wenn die Anforderung nicht erfüllt ist,
- "unclear", wenn die Erfüllung der Anforderung nicht beantwortet werden kann, da Kontextinformationen fehlen und
- "not applicable", wenn die Anforderung für das Dokument nicht relevant ist.

Formatieren Sie Ihre Ausgabe gemäß dem folgenden JSON-Schema:
{{"answer": <"yes"|"no"|"unclear"|"not applicable">}}

Anforderung: "{`requirement`}"
Dokument: "{`document`}"

Example:

Anforderung: "Angabe des Betrags der Forderungen mit einer Restlaufzeit von mehr als einem Jahr; gesondert für jeden ausgewiesenen Posten" Dokument: "544: Sämtliche Forderungen haben eine Restlaufzeit von weniger als einem Jahr."

{{"answer": "yes"}}

VIII In-Out-One-Shot-No-Template

System: Sie sind ein Experte im Bereich Wirtschaftsprüfung und haben perfekte Kenntnisse des HGB-Bilanzierungstandards. Sie antworten immer wahrheitsgemäß, ob eine bestimmte behördliche Anforderung in den folgenden Zeilennummern vollständig erfüllt ist.

Antworten Sie mit:
- "yes", wenn die Anforderung erfüllt ist, - "no", wenn die Anforderung nicht erfüllt ist,
- "unclear", wenn die Erfüllung der Anforderung nicht beantwortet werden kann, da Kontextinformationen fehlen und
- "not applicable", wenn die Anforderung für das Dokument nicht relevant ist.

Formatieren Sie Ihre Ausgabe gemäß dem folgenden JSON-Schema:
{{"answer": <"yes"|"no"|"unclear"|"not applicable">}}

Anforderung: "{`requirement`}"
Dokument: "{`document`}"

Example:

Anforderung: "Angabe des Betrags der Forderungen mit einer Restlaufzeit von weniger als

einem Jahr; gesondert für jeden ausgewiesenen Posten" Dokument: "544: Sämtliche Forderungen haben eine Restlaufzeit von weniger als einem Jahr."

{{"answer": "no"}}

Bibliography

[1] H. Jo, A. Hsu, R. Llanos-Popolizio, and J. Vergara-Vega,
Corporate governance and financial fraud of wirecard,
European Journal of Business and Management Research (2021) (cit. on p. 1).

[2] P. Sikka, *Financial crisis and the silence of the auditors*,
Accounting, organizations and society (2009) (cit. on p. 1).

[3] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, nature (2015) (cit. on pp. 1, 2).

[4] Y. Bengio, A. Courville, and P. Vincent,
Representation learning: A review and new perspectives,
IEEE transactions on pattern analysis and machine intelligence (2013) (cit. on p. 2).

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016,
URL: `http://www.deeplearningbook.org` (cit. on p. 2).

[6] S. Linnainmaa, *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors*,
Master's Thesis (in Finnish), University of Helsinki (1970) (cit. on p. 2).

[7] S. Linnainmaa, *Taylor expansion of the accumulated rounding error*,
BIT Numerical Mathematics (1976) (cit. on p. 2).

[8] D. E. Rumelhart, G. E. Hinton, and R. J. Williams,
Learning representations by back-propagating errors, nature (1986) (cit. on pp. 2, 46).

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei,
"Imagenet: A large-scale hierarchical image database," *Proc. CVPR*, 2009 (cit. on p. 2).

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton,
Imagenet classification with deep convolutional neural networks,
Advances in neural information processing systems (2012) (cit. on p. 3).

[11] S. Ren, K. He, R. Girshick, and J. Sun,
Faster r-cnn: Towards real-time object detection with region proposal networks (2015)
(cit. on pp. 3, 90).

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair,
A. Courville, and Y. Bengio, *Generative adversarial nets*,
Advances in neural information processing systems (2014) (cit. on p. 3).

[13] J. Ho, A. Jain, and P. Abbeel, *Denoising diffusion probabilistic models*,
Advances in neural information processing systems (2020) (cit. on p. 3).

[14] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al., *Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups*, IEEE Signal processing magazine (2012) (cit. on p. 3).

[15] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al., *Mastering the game of Go with deep neural networks and tree search*, nature (2016) (cit. on p. 3).

[16] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al., *Highly accurate protein structure prediction with AlphaFold*, nature (2021) (cit. on p. 3).

[17] A. Radford and K. Narasimhan, "Improving Language Understanding by Generative Pre-Training," 2018 (cit. on pp. 3, 13, 44, 45, 48, 49, 55, 58, 72).

[18] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al., *Gpt-4 technical report*, arXiv:2303.08774 (2023) (cit. on pp. 3, 5, 48, 49, 98, 100, 112, 113, 128–130).

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Proc. NeurIPS*, 2017 (cit. on pp. 3, 4, 14, 47, 48, 55, 97, 101, 112).

[20] G. Adams, A. R. Fabbri, F. Ladhak, E. Lehman, and N. Elhadad, "From sparse to dense: GPT-4 summarization with chain of density prompting," *Proc. EMNLP*, 2023 (cit. on p. 3).

[21] S. Frieder, L. Pinchetti, R.-R. Griffiths, T. Salvatori, T. Lukasiewicz, P. Petersen, and J. Berner, *Mathematical capabilities of chatgpt*, Advances in neural information processing systems (2024) (cit. on p. 3).

[22] D. M. Katz, M. J. Bommarito, S. Gao, and P. Arredondo, *Gpt-4 passes the bar exam*, Philosophical Transactions of the Royal Society A (2024) (cit. on p. 3).

[23] I. Auditing and A. S. B. (IAASB), *2023-2024 Handbook of International Quality Management, Auditing, Review, Other Assurance, and Related Services Pronouncements*, International Federation of Accountants (IFAC), 2024 (cit. on p. 3).

[24] Bundesministerium der Justiz, *German Commercial Code (Handelsgesetzbuch, HGB)*, URL: `https://www.gesetze-im-internet.de/hgb/` (visited on 02/21/2025) (cit. on p. 3).

[25] IFRS Foundation, *International Financial Reporting Standards*, URL: `https://www.ifrs.org/issued-standards/list-of-standards/` (visited on 02/21/2025) (cit. on p. 3).

[26] Federal Accounting Standards Advisory Board,
FASAB Handbook of Accounting Standards and Other Pronouncements,
URL: https://fasab.gov/accounting-standards/ (visited on 02/21/2025)
(cit. on p. 3).

[27] R. Sifa, A. Ladi, M. Pielka, R. Ramamurthy, L. Hillebrand, B. Kirsch, D. Biesner, R. Stenzel, T. Bell, M. Lübbering, et al.,
"Towards automated auditing with machine learning," *Proc. DocEng*, 2019
(cit. on pp. 4, 56, 58, 73, 89, 93, 115, 120).

[28] R. Ramamurthy, M. Pielka, R. Stenzel, C. Bauckhage, R. Sifa, T. D. Khameneh, U. Warning, B. Kliem, and R. Loitz,
"ALiBERT: improved automated list inspection (ALI) with BERT," *Proc. DocEng*, 2021
(cit. on pp. 4, 73, 89, 115).

[29] L. Hillebrand, M. Pielka, D. Leonhard, T. Deußer, T. Dilmaghani, B. Kliem, R. Loitz, M. Morad, C. Temath, T. Bell, R. Stenzel, and R. Sifa,
"sustain.AI: a Recommender System to analyze Sustainability Reports," *Proc. ICAIL*, 2023, DOI: 10.1145/3594536.3595131 (cit. on pp. 4, 6, 87, 88, 100, 106, 108–110).

[30] L. Hillebrand, T. Deußer, T. Dilmaghani, B. Kliem, R. Loitz, C. Bauckhage, and R. Sifa,
"KPI-BERT: A Joint Named Entity Recognition and Relation Extraction Model for Financial Reports," *Proc. ICPR*, 2022, DOI: 10.1109/ICPR56361.2022.9956191
(cit. on pp. 4, 6, 56, 70, 72–74, 77, 78, 80, 109, 115).

[31] L. Hillebrand, T. Deußer, T. Dilmaghani, B. Kliem, R. Loitz, C. Bauckhage, and R. Sifa,
"Towards automating Numerical Consistency Checks in Financial Reports,"
Proc. BigData, 2022, DOI: 10.1109/BigData55660.2022.10020308
(cit. on pp. 4, 6, 69, 115, 130).

[32] T. Deußer, M. Pielka, L. Pucknat, B. Jacob, T. Dilmaghani, M. Nourimand, B. Kliem, R. Loitz, C. Bauckhage, and R. Sifa, "Contradiction Detection in Financial Reports,"
Proc. NLDL, 2023 (cit. on pp. 4, 115, 130).

[33] A. Berger, L. Hillebrand, D. Leonhard, T. Deußer, T. B. F. De Oliveira, T. Dilmaghani, M. Khaled, B. Kliem, R. Loitz, C. Bauckhage, et al., "Towards automated regulatory compliance verification in financial auditing with large language models,"
Proc. BigData, 2023, DOI: 10.1109/BigData59044.2023.10386518
(cit. on pp. 4, 7, 113).

[34] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova,
"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,"
Proc. NAACL-HLT, 2019 (cit. on pp. 4, 5, 13, 44, 45, 48–50, 55, 57, 58, 70, 72, 87–89, 91, 97, 100, 101, 104, 112, 115, 116).

[35] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al.,
"Retrieval-augmented generation for knowledge-intensive nlp tasks," *Proc. NeurIPS*, 2020 (cit. on pp. 5, 45, 52, 98, 111, 130).

[36] L. Hillebrand, D. Biesner, C. Bauckhage, and R. Sifa, "Interpretable Topic Extraction and Word Embedding Learning Using Row-Stochastic DEDICOM," *Proc. CD-MAKE*, 2020, DOI: 10.1007/978-3-030-57321-8_22 (cit. on pp. 6, 21, 22, 32, 33).

[37] L. Hillebrand, D. Biesner, C. Bauckhage, and R. Sifa, *Interpretable Topic Extraction and Word Embedding Learning Using Non-Negative Tensor DEDICOM*, Machine Learning and Knowledge Extraction (2021), DOI: 10.3390/make3010007 (cit. on pp. 6, 21, 22).

[38] L. Hillebrand, P. Pradhan, C. Bauckhage, and R. Sifa, "Pointer-Guided Pre-Training: Infusing Large Language Models with Paragraph-Level Contextual Awareness," *Proc. ECML PKDD*, 2024, DOI: 10.1007/978-3-031-70359-1_23 (cit. on pp. 7, 98).

[39] L. Hillebrand, A. Berger, T. Deußer, T. Dilmaghani, M. Khaled, B. Kliem, R. Loitz, M. Pielka, D. Leonhard, C. Bauckhage, et al., "Improving Zero-Shot Text Matching for Financial Auditing with Large Language Models," *Proc. DocEng*, 2023, DOI: 10.1145/3573128.3609344 (cit. on pp. 7, 113).

[40] L. Hillebrand, A. Berger, D. Uedelhoven, D. Berghaus, U. Warning, T. Dilmaghani, B. Kliem, T. Schmid, R. Loitz, and R. Sifa, "Advancing Risk and Quality Assurance: A RAG Chatbot for Improved Regulatory Compliance," *Proc. BigData*, 2024, DOI: 10.1109/BigData62323.2024.10825431 (cit. on pp. 7, 127).

[41] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*, The MIT Press, 2009 (cit. on p. 10).

[42] J. Lafferty, A. McCallum, F. Pereira, et al., "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," *Proc. ICML*, 2001 (cit. on pp. 11, 63).

[43] G. Koch, R. Zemel, R. Salakhutdinov, et al., "Siamese neural networks for one-shot image recognition," *Proc. ICML*, 2015 (cit. on p. 11).

[44] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," *Proc. EMNLP-IJCNLP*, 2019 (cit. on pp. 11, 111, 113, 116).

[45] W. Research, *CosineDistance – Wolfram Language & System Documentation Center*, 2007, URL: https://reference.wolfram.com/language/ref/CosineDistance.html (visited on 02/21/2025) (cit. on p. 11).

[46] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," *Proc. CVPR*, 2006 (cit. on p. 12).

[47] D. M. Blei, A. Y. Ng, and M. I. Jordan, *Latent Dirichlet Allocation*, JMLR (2003) (cit. on pp. 12, 24).

[48] D. D. Lee and H. S. Seung, "Algorithms for Non-Negative Matrix Factorization," *Proc. NeurIPS*, 2000 (cit. on pp. 12, 24, 36).

[49] R. Sennrich, B. Haddow, and A. Birch,
"Neural Machine Translation of Rare Words with Subword Units," *Proc. ACL*, 2016
(cit. on p. 13).

[50] Y. Wu, *Google's neural machine translation system: Bridging the gap between human and machine translation*, arXiv:1609.08144 (2016) (cit. on p. 13).

[51] M. Ali, M. Fromm, K. Thellmann, R. Rutmann, M. Lübbering, J. Leveling, K. Klug, J. Ebert, N. Doll, J. S. Buschhoff, C. Jain, A. A. Weber, L. Jurkschat, H. Abdelwahab, C. John, P. O. Suarez, M. Ostendorff, S. Weinbach, R. Sifa, S. Kesselheim, and N. Flores-Herr, "Tokenizer Choice For LLM Training: Negligible or Crucial?"
Proc. Findings of NAACL, 2024 (cit. on p. 13).

[52] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean,
Distributed representations of words and phrases and their compositionality,
Advances in neural information processing systems (2013) (cit. on pp. 14, 17, 24).

[53] J. Pennington, R. Socher, and C. Manning,
"Glove: Global Vectors for Word Representation," *Proc. EMNLP*, 2014
(cit. on pp. 14, 17–19, 22, 24, 33, 90).

[54] R. Harshman, P. Green, Y. Wind, and M. Lundy,
A Model for the Analysis of Asymmetric Data in Marketing Research,
Marketing Science (1982) (cit. on pp. 14, 21, 23).

[55] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer,
"Deep contextualized word representations," *Proc. NAACL-HLT*, 2018
(cit. on pp. 14, 44).

[56] Z. S. Harris, *Distributional structure*, 1954 (cit. on p. 15).

[57] K. Sparck Jones,
A statistical interpretation of term specificity and its application in retrieval,
Journal of documentation (1972) (cit. on pp. 16, 89).

[58] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, et al.,
Okapi at TREC-3, Nist Special Publication Sp (1995) (cit. on p. 16).

[59] T. Mikolov, W.-t. Yih, and G. Zweig,
"Linguistic regularities in continuous space word representations," *Proc. NAACL*, 2013
(cit. on p. 17).

[60] J. R. Firth, *A synopsis of linguistic theory 1930-1955*,
Studies in Linguistic Analysis, Special Volume/Blackwell (1957) (cit. on p. 16).

[61] T. Mikolov, K. Chen, G. Corrado, and J. Dean,
"Efficient Estimation of Word Representations in Vector Space," *Proc. ICLR*, 2013
(cit. on p. 18).

[62] O. Levy and Y. Goldberg, "Neural Word Embedding as Implicit Matrix Factorization,"
Proc. NeurIPS, 2014 (cit. on pp. 22, 24).

[63] P. Symeonidis and A. Zioupos,
Matrix and Tensor Factorization Techniques for Recommender Systems, 2016
(cit. on p. 23).

[64] A. H. Andrzej, A. Cichocki, and T. V. Dinh, *Nonnegative DEDICOM Based On Tensor Decompositions for Social Networks Exploration*,
Australian Journal of Intelligent Information Processing Systems (2010)
(cit. on pp. 23, 24).

[65] B. W. Bader, R. A. Harshman, and T. G. Kolda,
Pattern analysis of directed graphs using DEDICOM: an application to Enron email.,
Office of Scientific & Technical Information Technical Reports (2006)
(cit. on pp. 23, 24).

[66] R. Sifa, Ojeda, K. Cvejoski, and C. Bauckhage, "Interpretable Matrix Factorization with Stochasticity Constrained Nonnegative DEDICOM," *Proc. KDML-LWDA*, 2018
(cit. on pp. 23, 26).

[67] R. Sifa, C. Ojeda, and C. Bauckage, "User Churn Migration Analysis with DEDICOM," *Proc. RecSys*, 2015 (cit. on p. 23).

[68] P. Chew, B. Bader, and A. Rozovskaya,
"Using DEDICOM for Completely Unsupervised Part-of-Speech Tagging,"
Proc. UMSLLS, 2009 (cit. on p. 23).

[69] M. Nickel, V. Tresp, and H.-P. Kriegel,
"A Three-Way Model for Collective Learning on Multi-Relational Data.," *Proc. ICML*, 2011 (cit. on p. 24).

[70] R. Sifa, R. Yawar, R. Ramamurthy, C. Bauckhage, and K. Kersting,
Matrix- and Tensor Factorization for Game Content Recommendation,
KI - Künstliche Intelligenz (2019) (cit. on p. 24).

[71] G. W. Furnas, S. Deerwester, S. T. Dumais, T. K. Landauer, R. A. Harshman, L. A. Streeter, and K. E. Lochbaum, "Information Retrieval Using A Singular Value Decomposition Model of Latent Semantic Structure," *Proc. SIGIR*, 1988 (cit. on p. 24).

[72] YongchangWang and L. Zhu,
"Research and implementation of SVD in machine learning," *Proc. ICIS*, 2017 471
(cit. on p. 24).

[73] I. Jolliffe, *Principal component analysis*, John Wiley and Sons Ltd, 2005 (cit. on p. 24).

[74] R. Lebret and R. Collobert, "Word Embeddings through Hellinger PCA,"
Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, 2014 (cit. on p. 24).

[75] D. Q. Nguyen, R. Billingsley, L. Du, and M. Johnson,
Improving Topic Models with Latent Feature Word Representations,
Transactions of the Association for Computational Linguistics (2015) (cit. on p. 24).

[76] M. Frank and P. Wolfe, *An algorithm for quadratic programming*,
Naval Research Logistics Quarterly (1956) (cit. on p. 28).

[77] J. Ni, J. Li, and J. McAuley,
"Justifying recommendations using distantly-labeled reviews and fine-grained aspects,"
Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019 (cit. on p. 32).

[78] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*,
arXiv:1412.6980 (2014) (cit. on p. 34).

[79] L. McInnes, J. Healy, and J. Melville,
UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction,
arXiv:1802.03426 (2018) (cit. on p. 38).

[80] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, *Survey of hallucination in natural language generation*,
ACM Computing Surveys (2023) (cit. on pp. 45, 52).

[81] R. Pascanu, T. Mikolov, and Y. Bengio,
"On the difficulty of training recurrent neural networks," *Proc. ICML*, 2013
(cit. on p. 46).

[82] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, Neural computation (1997)
(cit. on p. 46).

[83] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," *Proc. EMNLP*, 2014 (cit. on pp. 46, 55, 109).

[84] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever,
"Language Models are Unsupervised Multitask Learners," 2019 (cit. on pp. 48, 58).

[85] I. Beltagy, M. E. Peters, and A. Cohan, *Longformer: The long-document transformer*,
arXiv:2004.05150 (2020) (cit. on p. 48).

[86] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang, et al., *Big bird: Transformers for longer sequences*,
Advances in neural information processing systems (2020) (cit. on p. 48).

[87] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré,
"FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness,"
Proc. NeurIPS, 2022 (cit. on pp. 48, 111).

[88] A. Gu and T. Dao, *Mamba: Linear-time sequence modeling with selective state spaces*,
arXiv:2312.00752 (2023) (cit. on p. 48).

[89] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford,
D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand, et al., *Mixtral of experts*,
arXiv:2401.04088 (2024) (cit. on pp. 48, 98).

[90] G. Team, R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk,
A. M. Dai, A. Hauth, K. Millican, et al.,
Gemini: a family of highly capable multimodal models, arXiv:2312.11805 (2023)
(cit. on p. 48).

[91] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., "Language models are few-shot learners," *Proc. NeurIPS*, 2020 (cit. on pp. 49, 51, 98, 100, 128, 129).

[92] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, *RoBERTa: A Robustly Optimized BERT Pretraining Approach*, arXiv:1907.11692 (2019) (cit. on pp. 50, 58, 72, 90, 100, 104).

[93] J. Wei, M. Bosma, V. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, "Finetuned Language Models are Zero-Shot Learners," *Proc. ICLR*, 2022 (cit. on p. 51).

[94] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al.,
Pytorch: An imperative style, high-performance deep learning library,
Advances in neural information processing systems (2019)
(cit. on pp. 56, 62, 80, 98, 103).

[95] H.-U. Krause and D. Arora, *Controlling-Kennzahlen-key performance indicators*, Oldenbourg Wissenschaftsverlag, 2009 (cit. on p. 56).

[96] E. Brito, R. Sifa, C. Bauckhage, R. Loitz, U. Lohmeier, and C. Pünt, *A Hybrid AI Tool to Extract Key Performance Indicators from Financial Reports for Benchmarking*, Proc. DocEng (2019) (cit. on p. 56).

[97] D. Farmakiotou, V. Karkaletsis, J. Koutsias, G. Sigletos, C. D. Spyropoulos, and P. Stamatopoulos, "Rule-based named entity recognition for Greek financial texts," *Proc. COMLEX*, 2000 (cit. on pp. 56, 58).

[98] Y. Cao, H. Li, P. Luo, and J. Yao,
"Towards Automatic Numerical Cross-Checking: Extracting Formulas from Text," *Proc. WWW*, 2018 (cit. on pp. 56, 58, 73, 115, 130).

[99] M. Eberts and A. Ulges,
"Span-based Joint Entity and Relation Extraction with Transformer Pre-training," *Proc. ECAI*, 2020 (cit. on pp. 57, 58, 61, 63, 66, 72).

[100] Z. Huang, W. Xu, and K. Yu, *Bidirectional LSTM-CRF models for sequence tagging*, arXiv:1508.01991 (2015) (cit. on p. 57).

[101] B. Taillé, V. Guigue, G. Scoutheeten, and P. Gallinari,
"Let's Stop Incorrect Comparisons in End-to-end Relation Extraction!" *Proc. EMNLP*, 2020 (cit. on pp. 57, 58, 63, 66).

[102] B. Lester, D. Pressel, A. Hemmeter, S. R. Choudhury, and S. Bangalore, "Constrained Decoding for Computationally Efficient Named Entity Recognition Taggers," *Proc. EMNLP*, 2020 (cit. on p. 58).

[103] A. Ushio and J. Camacho-Collados, "T-NER: An All-Round Python Library for Transformer-based Named Entity Recognition," *Proc. EACL*, 2021 (cit. on p. 58).

[104] X. Wang, Y. Jiang, N. Bach, T. Wang, Z.-y. Huang, F. Huang, and K. Tu, "Improving Named Entity Recognition by External Context Retrieving and Cooperative Learning," *Proc. ACL/IJCNLP*, 2021 (cit. on pp. 58, 72).

[105] X. Wang, Y. Jiang, N. Bach, T. Wang, Z. Huang, F. Huang, and K. Tu, "Automated Concatenation of Embeddings for Structured Prediction," *Proc. ACL*, 2021 (cit. on pp. 58, 72).

[106] B. Xu, Q. Wang, Y. Lyu, Y. Zhu, and Z. Mao, "Entity Structure Within and Throughout: Modeling Mention Dependencies for Document-Level Relation Extraction," *Proc. AAAI*, 2021 (cit. on p. 58).

[107] S. Zeng, R. Xu, B. Chang, and L. Li, "Double Graph Based Reasoning for Document-level Relation Extraction," *Proc. EMNLP*, 2020 (cit. on p. 58).

[108] N. Zhang, X. Chen, X. Xie, S. Deng, C. Tan, M. Chen, F. Huang, L. Si, and H. Chen, "Document-level Relation Extraction as Semantic Segmentation," *Proc. IJCAI*, 2021 (cit. on p. 58).

[109] C. Liang, Y. Yu, H. Jiang, S. Er, R. Wang, T. Zhao, and C. Zhang, "BOND: BERT-Assisted Open-Domain Named Entity Recognition with Distant Supervision," *Proc. KDD*, 2020 (cit. on p. 58).

[110] Y. Shen, X. Ma, Y. Tang, and W. Lu, "A Trigger-Sense Memory Flow Framework for Joint Entity and Relation Extraction," *Proc. Web Conference*, 2021 (cit. on pp. 58, 72).

[111] J. Wang and W. Lu, "Two Are Better than One: Joint Entity and Relation Extraction with Table-Sequence Encoders," *Proc. EMNLP*, 2020 (cit. on pp. 58, 72).

[112] D. Ye, Y. Lin, and M. Sun, *Pack Together: Entity and Relation Extraction with Levitated Marker*, arXiv:2109.06067 (2021) (cit. on pp. 58, 72).

[113] Z. Zhong and D. Chen, "A Frustratingly Easy Approach for Entity and Relation Extraction," *Proc. NAACL*, 2021 (cit. on pp. 58, 72).

[114] K. Fundel, R. Küffner, and R. Zimmer, *RelEx—Relation extraction using dependency parse trees*, Bioinformatics (2007) (cit. on p. 58).

[115] H. Gurulingappa, A. Mateen-Rajpu, and L. Toldo, *Extraction of potential adverse drug events from medical case reports*, Journal of Biomedical Semantics (2012) (cit. on p. 58).

[116] J. Giorgi, X. Wang, N. Sahar, W. Y. Shin, G. D. Bader, and B. Wang, *End-to-end Named Entity Recognition and Relation Extraction using Pre-trained Language Models*, arXiv:1912.13415 (2019) (cit. on p. 58).

[117] A. Pappu, R. Blanco, Y. Mehdad, A. Stent, and K. Thadani, "Lightweight Multilingual Entity Extraction and Linking," *Proc. WSDM*, 2017 (cit. on p. 58).

[118] D. Treigueiros and R. Berry, "The application of neural network based methods to the extraction of knowledge from accounting reports," *Proc. HICSS*, 1991 (cit. on p. 58).

[119] D. Biesner, R. Ramamurthy, R. Stenzel, M. Lübbering, L. Hillebrand, A. Ladi, M. Pielka, R. Stenzel, R. Loitz, C. Bauckhage, and R. Sifa, *Anonymization of German financial documents using neural network-based language models with contextual word representations*, Springer International Journal of Data Science and Analytics (2021) (cit. on p. 58).

[120] M. Schuster and K. Nakajima, "Japanese and korean voice search," *Proc. ICASSP*, 2012 (cit. on pp. 58, 91, 101).

[121] A. Viterbi, *Error bounds for convolutional codes and an asymptotically optimum decoding algorithm*, IEEE Transactions on Information Theory (1967) 260 (cit. on p. 63).

[122] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," *Proc. ICLR*, 2018 (cit. on pp. 64, 80, 94, 105).

[123] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: a simple way to prevent neural networks from overfitting*, The journal of machine learning research (2014) (cit. on pp. 64, 75, 76, 80, 81, 92, 94).

[124] A. Lawrence, *Individual investors and financial disclosure*, Journal of Accounting and Economics (2013) (cit. on p. 70).

[125] A. Nwaobia, J. Kwarbai, J. Olajumoke, and A. Ajibade, *Financial reporting quality on investors' decisions*, International Journal of Economics and Financial Research (2013) (cit. on p. 70).

[126] P. Choudhary, K. Merkley, and K. Schipper, *Immaterial error corrections and financial reporting reliability*, Contemporary Accounting Research (2021) (cit. on p. 70).

[127] V. W. Fang, A. H. Huang, and W. Wang, *Imperfect accounting and reporting bias*, Journal of Accounting Research (2017) (cit. on p. 70).

[128] M. B. Nuijten, C. H. Hartgerink, M. A. Van Assen, S. Epskamp, and J. M. Wicherts, *The prevalence of statistical reporting errors in psychology (1985–2013)*, Behavior research methods (2016) (cit. on p. 72).

[129] N. Hassan, F. Arslan, C. Li, and M. Tremayne, "Toward automated fact-checking: Detecting check-worthy factual claims by claimbuster," *Proc. KDD*, 2017 (cit. on p. 72).

[130] M. Kobayashi, A. Ishii, C. Hoshino, H. Miyashita, and T. Matsuzaki, "Automated historical fact-checking by passage retrieval, word statistics, and virtual question-answering," *Proc. IJCNLP*, 2017 (cit. on p. 72).

[131] M. Nadeem, W. Fang, B. Xu, M. Mohtarami, and J. Glass, "FAKTA: An Automatic End-to-End Fact Checking System," *Proc. NAACL*, 2019 (cit. on p. 72).

[132] E. S. Parolin, Y. Hu, L. Khan, J. Osorio, P. T. Brandt, and V. D'Orazio, "CoMe-KE: A New Transformers Based Approach for Knowledge Extraction in Conflict and Mediation Domain," *Proc. BigData*, 2021 (cit. on p. 72).

[133] I. Yamada, A. Asai, H. Shindo, H. Takeda, and Y. Matsumoto, "LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention," *Proc. EMNLP*, 2020 (cit. on p. 72).

[134] J. Yu, B. Bohnet, and M. Poesio, "Named Entity Recognition as Dependency Parsing," *Proc. ACL*, 2020 (cit. on p. 72).

[135] P. Crone, *Deeper Task-Specificity Improves Joint Entity and Relation Extraction*, arXiv:2002.06424 (2020) (cit. on p. 72).

[136] Z. Xu, Y. Cao, R. Cao, G. Li, X. Liu, Y. Pang, Y. Wang, J. Zhang, A. Cheung, M. Tam, et al., "Jura: Towards Automatic Compliance Assessment for Annual Reports of Listed Companies," *Proc. CIKM*, 2021 (cit. on p. 73).

[137] H. Li, Q. Yang, Y. Cao, J. Yao, and P. Luo, "Cracking tabular presentation diversity for automatic cross-checking over numerical facts," *Proc. KDD*, 2020 (cit. on p. 73).

[138] M. Adkisson, J. C. Kimmell, M. Gupta, and M. Abdelsalam, "Autoencoder-based Anomaly Detection in Smart Farming Ecosystem," *Proc. BigData*, 2021 (cit. on p. 73).

[139] W. Chen, H. Li, J. Li, and A. Arshad, "Autoencoder-based outlier detection for sparse, high dimensional data," *Proc. BigData*, 2020 (cit. on p. 73).

[140] J. Li, J. Zhang, J. Wang, Y. Zhu, M. J. Bah, G. Yang, and Y. Gan, "VAGA: Towards Accurate and Interpretable Outlier Detection Based on Variational Auto-Encoder and Genetic Algorithm for High-Dimensional Data," *Proc. BigData*, 2021 (cit. on p. 73).

[141] M. Lübbering, M. Gebauer, R. Ramamurthy, R. Sifa, and C. Bauckhage, "Supervised autoencoder variants for end to end anomaly detection," *Proc. ICPR*, 2021 (cit. on p. 73).

[142] M. Lübbering, R. Ramamurthy, M. Gebauer, T. Bell, R. Sifa, and C. Bauckhage, "From imbalanced classification to supervised outlier detection problems: adversarially trained auto encoders," *Proc. ICANN*, 2020 (cit. on p. 73).

[143] A. F. Agarap, *Deep learning using rectified linear units (relu)*, arXiv preprint arXiv:1803.08375 (2018) (cit. on pp. 76, 81, 92).

[144] V. Levenshtein, *Binary codes capable of correcting spurious insertions and deletion of ones*, Problems of information Transmission (1965) (cit. on pp. 79, 81).

[145] D. Rolnick, A. Veit, S. Belongie, and N. Shavit, *Deep learning is robust to massive label noise*, arXiv:1705.10694 (2017) (cit. on p. 79).

[146] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, *Signature verification using a" siamese" time delay neural network* (1993) (cit. on p. 82).

[147] M. Gutierrez-Bustamante and L. Espinosa-Leal, *Natural Language Processing Methods for Scoring Sustainability Reports—A Study of Nordic Listed Companies*, Sustainability (2022) (cit. on p. 90).

[148] T. Goel, P. Jain, I. Verma, L. Dey, and S. Paliwal,
"Mining company sustainability reports to aid financial decision-making,"
Proc. Workshop on Knowledge Discovery from Unstructured Data in Financial Services, AAAI, 2020 (cit. on p. 90).

[149] M. Angin, B. Taşdemir, C. A. Yılmaz, G. Demiralp, M. Atay, P. Angin, and G. Dikmener,
A RoBERTa Approach for Automated Processing of Sustainability Reports,
Sustainability (2022) (cit. on p. 90).

[150] A.-I. Moreno and T. Caminero,
Application of text mining to the analysis of climate-related disclosures,
International Review of Financial Analysis (2022) (cit. on p. 90).

[151] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," *Proc. KDD*, 1996 (cit. on p. 90).

[152] R. Agombar, M. Luebbering, and R. Sifa,
"A clustering backed deep learning approach for document layout analysis,"
Proc. CD-MAKE, 2020 (cit. on p. 90).

[153] J. Ramos, *Using tf-idf to determine word relevance in document queries*, tech. rep., Rutgers University, Dept. of CS., 2003 (cit. on p. 95).

[154] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al.,
Llama 2: Open foundation and fine-tuned chat models, arXiv:2307.09288 (2023) (cit. on pp. 98, 100, 120, 124).

[155] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning,
"ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators,"
Proc. ICLR, 2020 (cit. on pp. 100, 104).

[156] M. Yasunaga, J. Leskovec, and P. Liang,
"LinkBERT: Pretraining Language Models with Document Links," *Proc. ACL*, 2022 (cit. on p. 100).

[157] M. Yasunaga, A. Bosselut, H. Ren, X. Zhang, C. D. Manning, P. S. Liang, and J. Leskovec, "Deep bidirectional language-knowledge graph pretraining," *Proc. NeurIPS*, 2022 (cit. on p. 100).

[158] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer Networks," *Proc. NeurIPS*, 2015 (cit. on pp. 100, 101).

[159] S. B. R. Chowdhury, F. Brahman, and S. Chaturvedi,
"Is Everything in Order? A Simple Way to Order Sentences," *Proc. EMNLP*, 2021 (cit. on p. 100).

[160] B. Cui, Y. Li, M. Chen, and Z. Zhang, "Deep Attentive Sentence Ordering Network," *Proc. EMNLP*, 2018 (cit. on p. 100).

[161] L. Logeswaran, H. Lee, and D. Radev,
"Sentence ordering and coherence modeling using recurrent neural networks,"
Proc. AAAI, 2018 (cit. on p. 100).

[162] A. Cohan, I. Beltagy, D. King, B. Dalvi, and D. Weld,
"Pretrained Language Models for Sequential Sentence Classification," *Proc. EMNLP*,
2019 (cit. on pp. 100, 106, 108, 110).

[163] A. Brack, A. Hoppe, P. Buschermöhle, and R. Ewerth, "Cross-domain multi-task
learning for sequential sentence classification in research papers," *Proc. JCDL*, 2022
(cit. on pp. 100, 108, 110).

[164] D. Jin and P. Szolovits, "Hierarchical Neural Networks for Sequential Sentence
Classification in Medical Scientific Abstracts," *Proc. EMNLP*, 2018
(cit. on pp. 100, 106, 108, 110).

[165] X. Shang, Q. Ma, Z. Lin, J. Yan, and Z. Chen,
"A span-based dynamic local attention model for sequential sentence classification,"
Proc. ACL/IJCNLP, 2021 (cit. on pp. 100, 108, 110).

[166] K. Yamada, T. Hirao, R. Sasano, K. Takeda, and M. Nagata,
"Sequential span classification with neural semi-Markov CRFs for biomedical abstracts,"
Proc. EMNLP, 2020 (cit. on pp. 100, 108, 110).

[167] Y. Cui, W. Che, T. Liu, B. Qin, and Z. Yang,
Pre-training with whole word masking for chinese bert, IEEE/ACM TASLP (2021)
(cit. on p. 101).

[168] A. Wettig, T. Gao, Z. Zhong, and D. Chen,
"Should You Mask 15% in Masked Language Modeling?" *Proc. EACL*, 2023
(cit. on p. 101).

[169] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio,
"Empirical evaluation of gated recurrent neural networks on sequence modeling,"
Proc. NeurIPS, 2014 (cit. on p. 104).

[170] P. He, J. Gao, and W. Chen, "DeBERTaV3: Improving DeBERTa using
ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing,"
Proc. ICLR, 2023 (cit. on p. 104).

[171] I. Beltagy, K. Lo, and A. Cohan,
"SciBERT: A Pretrained Language Model for Scientific Text," *Proc. EMNLP*, 2019
(cit. on pp. 104, 108).

[172] F. Dernoncourt and J. Y. Lee, "PubMed 200k RCT: a Dataset for Sequential Sentence
Classification in Medical Abstracts," *Proc. IJCNLP*, 2017 (cit. on p. 106).

[173] S. N. Kim, D. Martinez, L. Cavedon, and L. Yencken,
"Automatic classification of sentences to support evidence based medicine,"
BMC Bioinformatics, 2011 (cit. on p. 106).

[174] W. S. Richardson, M. C. Wilson, J. Nishikawa, and R. S. Hayward,
The well-built clinical question: a key to evidence-based decisions,
ACP journal club (1995) (cit. on p. 106).

[175] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, *Roformer: Enhanced transformer with rotary position embedding*, Neurocomputing (2024) (cit. on p. 111).

[176] O. Khattab and M. Zaharia, "Colbert: Efficient and effective passage search via contextualized late interaction over bert," *Proc. SIGIR*, 2020 (cit. on p. 111).

[177] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al., *Chain-of-thought prompting elicits reasoning in large language models*, Advances in Neural Information Processing Systems (2022) (cit. on pp. 112, 126).

[178] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan, *Tree of thoughts: Deliberate problem solving with large language models*, arXiv:2305.10601 (2023) (cit. on pp. 112, 122, 126).

[179] D. Biesner, M. Pielka, R. Ramamurthy, T. Dilmaghani, B. Kliem, R. Loitz, and R. Sifa, "Zero-Shot Text Matching for Automated Auditing using Sentence Transformers," *Proc. ICMLA*, 2022 (cit. on pp. 113, 115–117, 119).

[180] T. Deußer, S. M. Ali, L. Hillebrand, D. Nurchalifah, B. Jacob, C. Bauckhage, and R. Sifa, "KPI-EDGAR: A Novel Dataset and Accompanying Metric for Relation Extraction from Financial Documents," *Proc. ICMLA*, 2022 (cit. on p. 115).

[181] Y. Cao and J. Zhai, *Bridging the gap–the impact of ChatGPT on financial research*, Journal of Chinese Economic and Business Studies (2023) (cit. on p. 115).

[182] B. Neilson, *Artificial Intelligence Authoring Financial Recommendations: Comparative Australian Evidence*, Journal of Financial Regulation (2023) (cit. on p. 115).

[183] C. Yue, X. Xu, X. Ma, L. Du, H. Liu, Z. Ding, Y. Jiang, S. Han, and D. Zhang, *Leveraging LLMs for KPIs Retrieval from Hybrid Long-Document: A Comprehensive Framework and Dataset*, arXiv:2305.16344 (2023) (cit. on p. 115).

[184] E. Callanan, A. Mbakwe, A. Papadimitriou, Y. Pei, M. Sibue, X. Zhu, Z. Ma, X. Liu, and S. Shah, *Can GPT models be Financial Analysts? An Evaluation of ChatGPT and GPT-4 on mock CFA Exams*, arXiv:2310.08678 (2023) (cit. on p. 115).

[185] S. Wu, O. Irsoy, S. Lu, V. Dabravolski, M. Dredze, S. Gehrmann, P. Kambadur, D. Rosenberg, and G. Mann, *Bloomberggpt: A large language model for finance*, arXiv:2303.17564 (2023) (cit. on p. 115).

[186] D. Araci, *Finbert: Financial sentiment analysis with pre-trained language models*, arXiv:1908.10063 (2019) (cit. on p. 115).

[187] Y. Yang, M. C. S. Uy, and A. Huang, *Finbert: A pretrained language model for financial communications*, arXiv:2006.08097 (2020) (cit. on p. 115).

[188] Z. Liu, D. Huang, K. Huang, Z. Li, and J. Zhao, "Finbert: A pre-trained financial language representation model for financial text mining," *Proc. IJCAI*, 2021 (cit. on p. 115).

[189] Y. Arslan, K. Allix, L. Veiber, C. Lothritz, T. F. Bissyandé, J. Klein, and A. Goujon, "A comparison of pre-trained language models for multi-class text classification in the financial domain," *Proc. WWW*, 2021 (cit. on p. 115).

[190] A. H. Huang, H. Wang, and Y. Yang,
FinBERT: A large language model for extracting information from financial text,
Contemporary Accounting Research (2022) (cit. on p. 115).

[191] G. S. Temponeras, S.-A. N. Alexandropoulos, S. B. Kotsiantis, and M. N. Vrahatis,
"Financial fraudulent statements detection through a deep dense artificial neural network," *Proc. IISA*, 2019 (cit. on p. 115).

[192] F. Zhu, D. Ning, Y. Wang, and S. Liu,
"A Novel Cost-sensitive Capsule Network for Audit Fraud Detection," *Proc. IUCC*, 2021 (cit. on p. 115).

[193] M. Leippold, *Sentiment spin: Attacking financial sentiment with GPT-3*,
Finance Research Letters (2023) (cit. on p. 115).

[194] M. Aumüller, E. Bernhardsson, and A. Faithfull,
ANN-Benchmarks: A Benchmarking Tool for Approximate Nearest Neighbor Algorithms,
Information Systems (2020) (cit. on p. 117).

[195] X. Liu, Y. He, J. Chen, J. Gao, and L. Deng, *GPT Understands, Too*,
arXiv:2103.10385 (2021) (cit. on p. 118).

[196] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, *Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing*,
ACM Computing Surveys (2023) (cit. on p. 121).

[197] J. Long, *Large Language Model Guided Tree-of-Thought*, arXiv:2305.08291 (2023) (cit. on p. 122).

[198] D. Hulbert,
Tree of Knowledge: ToK aka Tree of Knowledge dataset for Large Language Models LLM,
2023, URL: `https://github.com/dave1010/tree-of-thought-prompting` (visited on 02/21/2025) (cit. on p. 122).

[199] I. Rodgers, J. Armour, and M. Sako,
How technology is (or is not) transforming law firms,
Annual Review of Law and Social Science (2023) (cit. on p. 128).

[200] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, et al., *Judging llm-as-a-judge with mt-bench and chatbot arena*,
Advances in Neural Information Processing Systems (2023) (cit. on pp. 129, 133).

[201] J. Guo, Y. Cai, Y. Fan, F. Sun, R. Zhang, and X. Cheng,
Semantic models for the first-stage retrieval: A comprehensive review,
ACM Transactions on Information Systems (TOIS) (2022) (cit. on p. 130).

[202] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W. T. Yih,
"Dense passage retrieval for open-domain question answering," *Proc. EMNLP*, 2020 (cit. on p. 130).

[203] J. Choi, K. Hickman, A. Monahan, and D. Schwarcz, *ChatGPT Goes to Law School*,
Journal of Legal Education (2022) (cit. on p. 130).

[204] D. M. Katz, M. J. Bommarito, S. Gao, and P. Arredondo, *GPT-4 Passes the Bar Exam*,
2023 (cit. on p. 130).

[205] I. Chalkidis, I. Androutsopoulos, and A. Michos, "Extracting contract elements," *Proc. ICAIL*, 2017 (cit. on p. 130).

[206] A. Louis, G. van Dijck, and G. Spanakis, "Interpretable Long-Form Legal Question Answering with Retrieval-Augmented Large Language Models," *Proc. AAAI*, 2023 (cit. on p. 130).

[207] OpenAI, *New and improved embedding model*, 2022, URL: `https://openai.com/index/new-and-improved-embedding-model/` (visited on 02/21/2025) (cit. on pp. 131, 132).

[208] OpenAI, *New embedding models and API updates*, 2024, URL: `https://openai.com/index/new-embedding-models-and-api-updates/` (visited on 02/21/2025) (cit. on pp. 131, 132).

[209] S. Robertson, H. Zaragoza, et al., *The probabilistic relevance framework: BM25 and beyond*, Foundations and Trends® in Information Retrieval (2009) (cit. on p. 132).

[210] G. V. Cormack, C. L. Clarke, and S. Buettcher, "Reciprocal rank fusion outperforms condorcet and individual rank learning methods," *Proc. ACM SIGIR*, 2009 758 (cit. on p. 132).

[211] Y. Liu, D. Iter, Y. Xu, S. Wang, R. Xu, and C. Zhu, *G-eval: Nlg evaluation using gpt-4 with better human alignment*, arXiv:2303.16634 (2023) (cit. on p. 132).

[212] A. Zhao, D. Huang, Q. Xu, M. Lin, Y.-J. Liu, and G. Huang, "Expel: Llm agents are experiential learners," *Proc. AAAI*, 2024 (cit. on p. 139).

[213] Y. Talebirad and A. Nadiri, *Multi-agent collaboration: Harnessing the power of intelligent llm agents*, arXiv:2306.03314 (2023) (cit. on p. 139).

[214] Y. Gu, L. Dong, F. Wei, and M. Huang, *MiniLLM: Knowledge distillation of large language models*, arXiv:2306.08543 (2023) (cit. on p. 139).

[215] D. Cheng, S. Huang, and F. Wei, "Adapting Large Language Models to Domains via Reading Comprehension," *Proc. ICLR*, 2024 (cit. on p. 140).

[216] C. Li, Z. Gan, Z. Yang, J. Yang, L. Li, L. Wang, J. Gao, et al., *Multimodal foundation models: From specialists to general-purpose assistants*, Foundations and Trends® in Computer Graphics and Vision (2024) (cit. on p. 140).

[217] OpenAI, *Learning to reason with LLMs*, 2024, URL: `https://openai.com/index/learning-to-reason-with-llms/` (visited on 02/21/2025) (cit. on p. 140).

[218] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al., *Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning*, arXiv:2501.12948 (2025) (cit. on p. 140).

[219] X. He, Y. Tian, Y. Sun, N. Chawla, T. Laurent, Y. LeCun, X. Bresson, and B. Hooi, *G-retriever: Retrieval-augmented generation for textual graph understanding and question answering*, Advances in Neural Information Processing Systems **37** (2025) (cit. on p. 140).

List of Figures

List of Tables