50 Schriften aus der Fakultät Wirtschaftsinformatik und Angewandte Informatik der Otto-Friedrich-Universität Bamberg

LLMs and Generative AI for Everything?

A Case Analysis for Applications in Natural Language Processing

Robin Jegan

University of Bamberg Press

50 Schriften aus der Fakultät Wirtschaftsinformatik und Angewandte Informatik der Otto-Friedrich-Universität Bamberg

Contributions of the Faculty Information Systems and Applied Computer Sciences of the Otto-Friedrich-University Bamberg

Schriften aus der Fakultät Wirtschaftsinformatik und Angewandte Informatik der Otto-Friedrich-Universität Bamberg

Contributions of the Faculty Information Systems and Applied Computer Sciences of the Otto-Friedrich-University Bamberg

Band 50

University of Bamberg Press
2026

LLMs and Generative AI for Everything?

A Case Analysis for Applications in Natural Language Processing

Robin Jegan

University
of Bamberg
Press
2026

Bibliographische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de/ abrufbar.

Diese Arbeit hat der Fakultät Wirtschaftsinformatik und Angewandte Informatik der Otto-Friedrich-Universität Bamberg als Dissertation vorgelegen.
1. Gutachter: Prof. Dr. Andreas Henrich
2. Gutachter: Prof. Dr. Gerhard Heyer
Tag der mündlichen Prüfung: 13.07.2026

Herstellung und Druck: docupoint, Magdeburg
Umschlaggestaltung: University of Bamberg Press
Umschlaggraphik: © Robin Jegan

University of Bamberg Press, ubp@uni-bamberg.de, Bamberg 2026
ROR https://ror.org/004fa0x02

ISSN: 1867-7401 (Print) eISSN: 2750-8560 (Online)
ISBN: 978-3-98989-130-2 (Print) eISBN: 978-3-98989-131-9 (Online)

URN: urn:nbn:de:bvb:473-irb-116344x
DOI: https://doi.org/10.20378/irb-116344

For A & K

Acknowledgments

From the beginning, the creation of this thesis has been made possible, likely and in the end also realistic through the support of many people. Your continued and unwavering support has left me grateful and was essential in completing this work. First and foremost, I would like to acknowledge my doctoral supervisor Prof. Dr. Andreas Henrich for his guidance, continued support and constructive feedback. Your role while leading the chair of Media Informatics in a purposeful, supportive and emphatic manner has led to a valuable and enjoyable time even during stressful periods. Always having an open door for any kind of problem felt particularly helpful. As part of my thesis committee, I would further like to thank both Prof. Dr. Gerhard Heyer and Prof. Dr. Stefan Ultes for your time, support, ideas and comprehensive feedback.

I would also like to thank all colleagues at the chair, Aaron, Annette, Claudia, Daniela, Felix, Ines, Leon & Leon, Martin & Martin, Maximilian, Michaela, Siggi, Sonja, Tobias & Tobias, for their support and the pleasant working environment during my nearly seven years there, including but not limited to long, podcast-worthy discussions during lunch break, assistance in any conceivable way, shape or form, or even our rounds at the Kicker. Furthermore, I would like to thank other researchers and colleagues from Bamberg and beyond for their support: Leonie & Christian, Alex & Andrea (and the other members of the Cheese Platter Collective), Alexander (and the Chair of Artificial Intelligence and Data Stream Mining at Coburg University of Applied Sciences) and Adrian, Felix & Markus.

I would be remiss not to mention my students, who enriched my time in teaching and supervising at the University of Bamberg. A special thank you to my tutors Aaron, Johannes, Julia, Lena, Magda, Max and Sebastian in supporting me and also to the students, whose thesis I had the pleasure of supervising, namely Aaron, Anna, Aurelia, Benjamin, Daniel, David, Dominik, Franziska, Frauke, Giulia, Leon, Max, Maximilian, Niklas & Niklas, Pascal, Philipp, Rica, Stefan and Steffen. Your work all directly or indirectly contributed to this thesis, so thank you! For proofreading this thesis, a big thanks to Aaron, Hoschi, Julia, Kristina, Leon, Marco, Maximilian, Mona and Sonja!

Further, I would like to thank my parents and sister for their support across all my life, as well as Patti & Nati, Marco & Julia, Hoschi, Tobi, Jessi & Patrick, Micha, Thomas, Nina & Franky and Mad, Sabrina, Johanna & Jasmin for being part of my life.

Lastly, I would like to thank my wife, Kristina, for her patience, steadfast support and calming presence in my life.

Abstract

The research field of Natural Language Processing (NLP) has experienced a major shift since the introduction of Large Language Models (LLMs). All facets and application scenarios within NLP have been impacted by the use of LLMs. Current research as well as practice of text processing tools is focused mainly on the application and development of LLMs. Major investments, not only by LLM providers but also other companies applying LLMs in their workflows, have only solidified the role of LLMs in NLP – and in other research and application areas – as part of the artificial intelligence boom in recent years.

However, limitations and downsides of the application of LLMs have also emerged. Problems regarding the generated texts as well as the environmental impact of the large-scale use of LLMs are just two of many factors that should be critically analyzed, despite the hype and the prevalence of LLMs for NLP tasks. These restrictions provide the main motivation for this thesis. Traditional models as alternatives to LLMs will be discussed from different perspectives. The characterization of *traditional models* will be progressively developed as features of alternatives to LLMs will emerge during the course of this thesis. This process will be grounded in experiments, observations and evaluations. Several NLP applications will be presented by surveying the state of the art with neural network-based models such as LLMs as well as the current usage of traditional models. The concrete NLP applications comprise information and relation extraction, text classification, text segmentation, text simplification and text summarization.

The first half of this thesis will present the emergence of LLMs contextualized along previous developments within NLP. Characteristics of the selected NLP applications will be collected before a structured literature review will display the prevalence of LLMs regarding each application and will discuss if traditional models are still actively researched. Lessons from domains with long-standing development procedures and processes will also be taken into account to provide a purposeful and structured manner of approaching NLP tasks. A collection of challenges within current

NLP will conclude the first half of the thesis, which will serve as motivation for the analysis of experiments and applications of the latter half.

The second half of this thesis will present observations and evaluations from use cases, aligned towards the challenges recognized in the first half. Through the analysis of these use cases, benefits of applying traditional models will be collected and supported, in particular through the analysis of a text segmentation use case that is purposefully applied with the lessons drawn from the first half of the thesis in mind. The interpretation of these results will conclude in a discussion on the applicability of traditional models in contrast to LLMs and also give recommendations of both model types for different use cases.

Concrete use cases for information extraction, entity matching, text classification and text segmentation will be presented, in which traditional models match or surpass the performance of modern methods. Through improved efficiency as well as enhanced explainability and reproducibility in comparison with neural network-based techniques, these showcases demonstrate the continued relevancy of traditional techniques in today's NLP landscape.

Overall, this thesis discusses the role of traditional models in current NLP research and practice, especially in contrast and comparison to modern neural network-based approaches including LLMs. The applicability of modern and less modern techniques is analyzed through a case-based analysis of NLP tasks in a structured and purposeful manner.

Kurzfassung (German Summary)

Das Forschungsgebiet der natürlichen Sprachverarbeitung – Natural Language Processing (NLP) – hat seit der Einführung großer Sprachmodelle – Large Language Models (LLMs) – einen grundlegenden Wandel erfahren. Alle Aspekte und Anwendungsszenarien in der NLP wurden durch den Einsatz von LLMs beeinflusst. Die aktuelle Forschung sowie die Anwendung von Textverarbeitungswerkzeugen konzentrieren sich hauptsächlich auf die Anwendung und Entwicklung von LLMs. Umfangreiche Investitionen nicht nur von LLM-Anbietern sondern auch von anderen Unternehmen, die LLMs in ihren Arbeitsabläufen und Pipelines einsetzen, haben die Rolle von LLMs in der NLP und darüber hinaus als Teil des Booms der künstlichen Intelligenz in den letzten Jahren nur noch gefestigt.

Allerdings sind auch Einschränkungen und Nachteile von LLMs ersichtlich geworden. Probleme in den generierten Texten sowie die Auswirkungen auf die Umwelt durch den weitreichenden Einsatz von LLMs sind nur zwei von vielen Faktoren, die trotz des Hypes und der Verbreitung von LLMs für NLP-Aufgaben kritisch analysiert werden sollten. Diese Einschränkungen bilden die Hauptmotivation für diese Doktorarbeit. Traditionelle Modelle als Alternativen zu LLMs werden dabei aus verschiedenen Perspektiven beleuchtet. Die Charakterisierung *traditioneller Modelle* wird schrittweise weiterentwickelt, da sich im Laufe der Arbeit Merkmale alternativer Lösungen zu LLMs herauskristallisieren werden. Dieser Prozess wird auf Experimenten, Beobachtungen und Evaluationen basieren. Es werden mehrere NLP-Anwendungen vorgestellt, wobei der aktuelle Stand der Technik sowohl von Modellen basierend auf neuronalen Netzwerken und LLMs als auch die Nutzung traditioneller Modelle in diesen Anwendungen untersucht werden. Die konkreten NLP-Anwendungen umfassen Informations- und Relationsextraktion, Textklassifizierung, Textsegmentierung, Textvereinfachung und Textzusammenfassung.

In der ersten Hälfte dieser Doktorarbeit wird die Entstehung von LLMs im Kontext früherer Entwicklungen innerhalb der NLP vorgestellt. Es werden Merkmale der ausgewählten NLP-Anwendungen gesammelt be-

vor in einer strukturierten Literaturrecherche sowohl die Verbreitung von LLMs in Bezug auf jede der genannten Anwendungen als auch die Frage, ob traditionelle Modelle noch aktiv erforscht werden, einer Diskussion unterzogen werden. Dabei werden auch Erkenntnisse aus Bereichen mit langjährigen Entwicklungsverfahren und -prozessen berücksichtigt, um einen zielgerichteten und strukturierten Ansatz für NLP-Aufgaben zu ermöglichen. Eine Zusammenstellung der Herausforderungen innerhalb der aktuellen NLP bildet den Abschluss der ersten Hälfte der Doktorarbeit und dient als Motivation für die Analyse von Experimenten und Anwendungsfällen in der zweiten Hälfte.

Die zweite Hälfte dieser Doktorarbeit präsentiert Beobachtungen und Evaluationen aus Anwendungsfällen, die auf die oben erwähnten Herausforderungen abgestimmt sind. Durch die Analyse dieser Anwendungsfälle werden Vorteile gesammelt, die durch die Verwendung traditioneller Modelle beobachtet werden, insbesondere durch die Anwendung auf ein Textsegmentierungsproblem, welches gezielt unter Berücksichtigung der Erkenntnisse aus der ersten Hälfte der Doktorarbeit bearbeitet wird. Die Interpretation dieser Ergebnisse mündet in einer Diskussion der Anwendbarkeit traditioneller Modelle gegenüber LLMs und Empfehlungen für beide Modelltypen für verschiedene Anwendungsfälle.

Konkrete Anwendungsfälle zur Informationsextraktion, Entitätenabgleich, Textklassifizierung und Textsegmentierung werden vorgestellt, in denen traditionelle Modelle die Leistung moderner Methoden erreichen oder übertreffen. Durch eine verbesserte Effizienz sowie erhöhte Erklärbarkeit und Reproduzierbarkeit im Vergleich zu neuronalen netzwerkbasierten Techniken zeigen diese Beispiele die anhaltende Relevanz traditioneller Techniken in der heutigen NLP-Forschungslandschaft.

Insgesamt befasst sich diese Doktorarbeit mit der Rolle traditioneller Modelle in der aktuellen NLP-Forschung und -Praxis, insbesondere im Kontrast und Vergleich zu modernen, auf neuronalen Netzwerken basierenden Ansätzen und LLMs. Anhand einer fallbasierten Analyse von NLP-Aufgaben wird die Anwendbarkeit moderner und weniger moderner Techniken analysiert und auf strukturierte und zielgerichtete Weise dargestellt.

Contents

1 Introduction

> "Writing is perhaps the greatest of human inventions, binding together people who never knew each other, citizens of distant epochs." [Sagan, 1995, *p.* 281]

Text & Language

Text as a medium is everywhere around us. The role of text in all facets of life cannot be overstated, may it be written text, ranging from personal letters to news articles all the way to scientific publication. Even spoken language is nowadays easily transformed into written text. While the use of language, especially written text, is seen as a distinctive human accomplishment, recently a new communication partner has emerged with capabilities of producing language, often with human-like semblance in terms of quality of the produced text.[1]

Large Language Models

This *new partner* refers to Large Language Models (LLMs). Prior to such models, a form of language was already produced by earlier techniques and systems, e.g., by applying statistical methods or rule-based approaches, but in a much more limited and constrained manner. Through LLMs, however, the kind of written text referred to as the greatest of human *inventions* as characterized in the above quote, while still based on that invention, was now available for non-humans as well. The ability to produce text in a generative and abstractive manner, most prominently through a chat-based interface, has enabled new and impressive capabilities and application scenarios through the use of such LLMs.

Natural Language Processing

The manner in which earlier techniques of Natural Language Processing (NLP) were able to process text was much more analytical and less generative. Approaches for information extraction or text classification purposes, or even when new text was created in text summarization, usually produced output in a fixed or extractive way. Instead, the abstractive and creative generation of new text was only recently advanced through the introduction of neural network-based systems trained on large datasets. Thus, the analogy between the invention of writing as a human

[1] References are largely omitted in this chapter due to the general nature of the introduction and to improve readability. However, references will be included in the subsequent chapters.

achievement and the capabilities of neural networks that are modeled after neurons and the human brain might be somewhat farfetched. However, it cannot be denied that the introduction of neural networks and systems based on the transformer architecture [Vaswani et al., 2017] has impacted every facet of NLP and, through the rise of LLMs, many parts of our daily life. While the capabilities of LLMs are impressive and the hype initialized by LLM providers is – at least in 2025 – still ongoing, drawbacks, limitations and clear disadvantages through the training, use and application of LLMs have become apparent.

Traditional Models

As a consequence, an analysis is justified, if and when LLMs are required for NLP tasks and if the generative nature of LLMs is the desired technique of approaching every problem. Other and often older techniques are a potential alternative. These techniques, which will be referred to as *traditional models*,[2] exhibit widely differing features, but in most cases present advantages in several dimensions in contrast to LLMs, such as efficiency, explainability and reproducibility. Additionally, traditional models are intended for a dedicated desired use case and not all-purpose software as their modern counterparts. Therefore, traditional models are more constrained and built for a specific NLP task, which at first might seem like a limiting factor but can also work as an advantage due to the features listed above. This thesis will attempt to provide an overview of traditional models in contrast to modern neural network-based methods, with limitations, benefits and recommendations of both types of models. Through the analysis in this thesis, LLMs as a *human invention* in reference to the introductory quote will be critically reviewed regarding several NLP tasks, in particular in contrast to applicable traditional models. The research questions and problem statement will be presented in the next Section 1.1, before the thesis structure is laid out in Section 1.2.

1.1 Problem Statement

Narrowing of Scope

NLP is a broad research field, with dozens of different tasks and research perspectives. Not only due to the rise of LLMs, but certainly contributed by this development, the interest in NLP has increased in recent years with a sharp uptick in published papers. This growth has also led to boosted contributions in software projects concerned with text processing and to a

[2] A definition of traditional techniques, as used in this thesis, will be attempted in Section 2.5 and once again later in the thesis in a renewed attempt of the same definition in Section 8.6.

surge in conferences and their size. While this development is not always met with appreciation, higher visibility regarding NLP has certainly also led to increased opportunities regarding research and practice, for individuals, research institutes and businesses altogether. Due to the breadth of this field, it is necessary to narrow the research goals for this thesis. Several applications will be defined below in Chapter 3, which will be more closely analyzed for this thesis according to the following research questions:

Research Questions

1. Are traditional models still in use in current NLP applications and NLP research?
2. Which NLP tasks can be processed with traditional models yielding performance competitive with modern neural network-based approaches?
3. What are potential benefits of using traditional models?
4. How can a systematic approach support the selection of appropriate methods for different NLP tasks?

Extent of Research Questions

Research question 1 will be conducted using a Structured Literature Review (SLR), which will be presented in Section 4.1. First learnings regarding the second research question 2 can also be noted through the SLR. The analysis of use cases and corresponding limitations of the respective tasks, as well as observations as part of the implementations, are mainly referenced in Chapters 7 and 9. Research question 3 is the natural follow-up to research question 2, i.e., the advantages of traditional models in use today, especially in contrast to modern LLMs. Potential avenues for improvement by using traditional techniques can be envisioned when considering the drawbacks and limitations of using LLMs for NLP tasks, as presented in Chapter 6, as well as in Chapter 8 for the potential benefits when using traditional models. Similarly to research question 2, research question 3 is also referenced in Chapter 9 regarding the use case of text segmentation and the discussion on applicability of techniques for different tasks as part of Chapter 10. The final research question 4 will be analyzed in several parts of this thesis, but mainly in Chapters 5, 9 and 10, as other disciplines and their perspectives present opportunities for NLP. Further details regarding the structure of this thesis will be presented in the next section.

1.2 Thesis Structure

Broad Structure

This thesis can be split into two halves. The first half, from Chapter 1 to Chapter 6, is focused on setting the stage for the observations, evaluations and experiments presented in the latter half, through focused literature work, interdisciplinary research and the identification of challenges for NLP tasks. The second half, from Chapter 7 to Chapter 11, is intended to utilize the learnings from the former half, through use cases and interpretation of results.

Detailed Structure of First Half

After the introduction in the current chapter, the evolution towards the recent explosive emergence of LLMs will be presented in Chapter 2, focused on applications and milestones of this development. The following Chapter 3 will define the applications that will be analyzed in this thesis, with brief definitions and descriptions of each application. Subsequently, Chapter 4 includes the SLR on the use of traditional models in current NLP research and a high-level comparison of text processing methods. Connected to this application-focused work is Chapter 5, in which different domains will take center stage, in particular engineering and design theory. Perspectives and learnings from these domains will be transferred to NLP tasks and problems. Afterwards, and concluding the first half of this thesis, challenges in current NLP research and practice will be presented in Chapter 6 such as hallucinations, lacking metrics and the focus on research and datasets concerned with the English language.

Detailed Structure of Second Half

Chapter 7 marks the beginning of the second half of this thesis and builds on the previous chapter, i.e., the challenges presented there will be discussed from the perspective of selected use cases and to what extent those challenges are observed. Benefits of using traditional approaches will be detailed in Chapter 8, supported by observations and evaluations of the applications from the prior Chapters 6 and 7. To support the findings of the chapter on benefits of traditional models, an analysis on a new use case is given in Chapter 9. Direct references to Chapter 5 and its perspectives from other domains and learnings thereof will be applied in this Chapter 9, especially for the initialization phase of the project. The lessons learned from all analyzed use cases in this thesis will be highlighted in Chapter 10 and the applicability of techniques for different NLP tasks will be given, both from the perspective of the techniques and as recommendations for different tasks. Finally, Chapter 11 will close this thesis.

2 Present Landscape of NLP Techniques

In this chapter, an overview will be presented regarding the major shifts of NLP methods. This overview will help to classify the overarching theme of this thesis, namely the contrast between traditional techniques and the state of the art, which in the middle of the 2020s mainly consists of LLMs. In Section 2.1 a short look at some linguistic background will be provided as well as at its influence on the ability to process textual information. After laying the groundwork with some earlier techniques and models in Section 2.2, the shift towards approaches using machine learning techniques will be discussed in Section 2.3, in which a certain gray area between traditional techniques and modern LLMs will be identified. Furthermore, the modern state of the art is the focus of Section 2.4, in which the release of ChatGPT, based on the Generative Pre-trained Transformer (GPT) language model, in 2022 is characterized as a milestone, changing the research field of NLP in a nearly all-encompassing way. Finally, an attempt at defining the term *traditional technique* will be made in Section 2.5.

2.1 Foundations of Natural Language Processing

Linguistic Background

The research discipline of linguistics incorporates the study of different properties of languages and subdisciplines including but not limited to phonetics, morphology, syntax, semantics and pragmatics. All these research fields, and more, are used in NLP. While a complete overview is not in scope, the discoveries and insights of the linguistic disciplines are still part of the very foundation of this thesis. As a brief reference to the importance of linguistics for NLP, a few applications will be mentioned here, before later being discussed in the relevant parts of the thesis in more detail. Within these applications, linguistic properties such as morphology are a part of widely used tools of processing pipelines, e.g., stem-

mers [Porter, 2001] or lemmatizers [Jurafsky and Martin, 2024, *p.* 23f.].[3] Other linguistic properties include syntactical analysis[4], which is used to gauge the cohesiveness of text units, or semantics[5], for determining characteristics such as text complexity.

Unstructured Text

Working with text as a medium enables certain possibilities while also increasing difficulty compared to other mediums. Text, as is common for articles, chat messages or publications, is unstructured in a myriad of ways due to the freedom of human use of language, especially in terms of syntax, semantics and pragmatics. However, this freedom results in problems when transforming text in a representation that enables the processing of the data. These problems can become apparent when normalizing the data or when applying typical processing steps such as tokenization [Jurafsky and Martin, 2024, *p.* 4f.]. Simple models such as Bag of Words (BoW), referenced as early as the 1950s in Harris [1954], or more complex vector representation such as word2vec [Mikolov, 2013] enable calculations in order to compare, contrast and further analyze text. These approaches provide the groundwork for even more sophisticated LLMs.

Text Representation

The transformation of text into a mathematical representation results in certain restrictions that need to be taken into account. One such restriction is the problem of dimensionality when processing huge corpora, which in a simple BoW model would result in a prohibitively large number of dimensions since each unique word would represent one such dimension.[6] Another problem is data sparsity. This problem occurs, e.g., in similarity computation between sentences represented by BoW models: Vector representations of sentences contain mostly zero-entries due to the small number of unique words per sentence compared to the full corpus so that comparisons between two such vectors yield little information. Consecutive words may be in a variety of relations and the most important other word for a given word may not be in its immediate neighborhood or even in other sentences altogether.[7]

[3] In use for nearly all discussed applications, such as text classification (e.g., in Raab [2023, *p.* 30ff.]), information extraction (e.g., in Pulver [2023, *p.* 27]) and text simplification (e.g., in Fruth [2022, *p.* 18]). Many of the cited applications here and in the following are part of supervised students' theses. An introduction to those theses will be given in Chapter 7.

[4] See text segmentation (e.g., in Jegan and Henrich [2025b, *p.* 276]) or text summarization (e.g., in Dal Cin [2025, *p.* 29]).

[5] See text simplification (e.g., in Fruth [2022, *p.* 17]) and text summarization (e.g., in Dal Cin [2025, *p.* 29]).

[6] The distinction between discrete data types, such as natural language, and continuous data types, such as images when viewed from a computer vision perspective, is analyzed, e.g., in Cartuyvels et al. [2021], but will not be discussed further in this thesis.

[7] The restrictions noted here will be covered once again in Section 2.3.

In textual data, processing pipelines include aspects such as tokenization, lemmatization, stemming and stop word elimination, which can all be challenging in their own right or must be adapted for languages and use cases. Additionally, the choice of how to represent units of text for further computation, using words, phrases, sentences, n-grams, Term Frequency-Inverse Document Frequency (TF-IDF) representations [Jones, 1972; Luhn, 1957] among others or a combination of the above is crucial for the performance of the selected models [Jurafsky and Martin, 2024, *p*. 33ff. and *p*. 111ff.]. NLP Pipeline

While the focus of this thesis will not be on the creation of text representations or pipeline aspects such as tokenization or feature selection, they are nevertheless an important part of each of the use cases described in more detail in Chapter 4. Furthermore, some selection criteria for choosing an appropriate method or technique for a certain application will be described in Chapter 5 and later exemplary in Chapter 9 for a selected use case.

2.2 Early Approaches in NLP

Advancements in architectures and techniques enabled a different scale of model creation and training. These developments, starting at the beginning of this millennium, will be the focus of Section 2.3. However, prior to using machine learning approaches built on neural networks, linguistic problems were tackled computationally with a variety of different methods, a small selection of which will be presented here due to their relevancy for the applications shown later in this thesis. First, the analysis of linguistic properties by hand enabled various NLP applications, see below in Section 2.2.1, and second, machine learning-based processes introduced more capable computational resources to NLP, see Section 2.2.2.

2.2.1 Linguistic Properties

The analysis of textual properties based on linguistic studies was the main contributor to the early computational approaches in NLP. One example of such studies is syntactic distributions, classically done by using context-free grammars,[8] which enabled the parsing of sentences and thus further Syntactic Distributions

[8] Proposed in both Backus [1959] and Chomsky [1956] in the 1950s.

analysis, e.g., for downstream tasks[9] such as information extraction or semantic parsing.

Rules & Regular Expressions

A further example comprises the use of hand-crafted rules or expressions. This class of approaches includes regular expressions, rule-based systems as well as decision trees, which can be incorporated as part of a pipeline or stand for themselves, for instance in information extraction tasks, see Section 3.2 and Pulver [2023]. These approaches from above, i.e., regular expressions, rule-based systems and decision trees, require a manual analysis of the respective dataset. Regular expressions are created, e.g., in order to filter out certain information from a text such as Hypertext Markup Language (HTML) tags when using a preprocessing pipeline to extract the full text of a website or from structured data in eXtensible Markup Language (XML) files in general. A manual study of those files is usually recommended to prevent losing context information that is included in metadata fields or other parts of the structured text. With a broader application scope, rule-based systems can apply regular expressions or other rules on problems such as early part-of-speech tagging scenarios [Karlsson et al., 1995], information extraction [Nahm and Mooney, 2002] or as part of decision trees for coreference resolution [McCarthy and Lehnert, 1995].

Matching Rules

The rules can originate from diverse linguistic fields but defining them still requires manual work. On the one hand, simple string matching rules are employed to identify connected elements or discard irrelevant words [Chiticariu et al., 2010]. On the other hand, morphosyntactic properties such as part-of-speech tags and numbers as well as lexical features and more expansive actions, such as lookups within gazetteers, can help in extracting phrases. All these actions combined can then be applied, e.g., in coreference resolution [Eisenstein, 2018, *p.* 367f.].

Challenges for Rule-Based Approaches

Two challenges arise from the vast possibilities offered by such linguistically motivated procedures. First, a manual and time-consuming analysis of the data is the main requirement in order to start rule-based processing of text, which can vary depending on the length, variety and the degree of structuredness of the dataset. Second, after a set of properties has been identified that can represent the rules, the order or the weighting of these rules need to be investigated and experimented upon. Without going into

[9] Although the term *downstream task* is used here for older models, the term itself would only rise to a larger prominence much later in the context of (large) language models. There, pre-trained models based on huge datasets serve as a foundation for such downstream tasks, e.g., by fine-tuning on tasks including but not limited to classification or Named-Entity Recognition (NER) [Eisenstein, 2018, *p.* 339f.].

too much detail here, more expansive systems are conceivable such as hybrid systems, combining morphosyntactic features as well as lookups in dictionaries and gazetteers [Farmakiotou et al., 2000] or the identification of such features as a learning task [Paliouras et al., 2000].

2.2.2 Towards Machine Learning

Early Learning Approaches

The learning task of decision trees serves as a transition point in order to introduce machine learning into the equation. One big advantage of decision trees and rule-based approaches is their interpretability. However, the construction of rules, as mentioned above, is time-consuming. This is why early research in information theory, dating back to the 1980s, see Quinlan [1986], had begun to shift towards the study of the automatic synthesis of decision trees from examples.[10] The core concepts behind these early systems revolve around the classification of entities within a dataset, which is treated as a supervised learning problem and thus necessitates a training dataset with labeled data or a domain expert manually crafting the rules [Quinlan, 1986, *p.* 83f.]. Classification of the data entities is then achieved by identifying attributes that can be separated according to some information gain, which enables the classification for unseen samples [Quinlan, 1986, *p.* 92ff.].

Boosting & Bagging

While traditionally only one decision tree is created as the output by the models developed (see above and Quinlan [1986, 1993]), the natural next step in order to improve the learning of decision trees was utilizing a large set of suitable decision trees. These iteratively learned decision trees, which were by design not as optimized as the one produced by, e.g., C4.5, are intended as "weak learners" [Freund et al., 1996, *p.* 148ff.], which can be combined into one composite and improved classifier. Different algorithms were created for these procedures, most famously titled boosting [Freund et al., 1996]. Conversely, instead of training different decision trees, splitting the dataset was another method innovated around the same time, called bagging [Breiman, 1996]. New training datasets of equal size are created by sampling with replacement from the old training data and decision trees are generated for each newly created training dataset. The results of each decision tree are weighted equally when calculating the final decision. Both procedures, bagging as well as boosting,

[10] Some famous representatives of such systems are ID3 [Quinlan, 1986] and C4.5 [Quinlan, 1993].

are intended to decrease the error rates of classifiers based on a single decision tree [Quinlan et al., 1996].[11]

Decision Tree Learning

Taking a step back towards applications within NLP, the learning of decision trees is practically used, e.g., in coreference resolution among others. By including morphosyntactic features such as Part of Speech (PoS) tags, phrase structure, string matching and more as features, decision trees can be constructed to identify relationships between entities within a sentence [Soon et al., 2001]. One big advantage of such systems, especially when compared to modern neural networks, is the explainability of decision trees and the influence of certain features on a decision. For example, the impact of the string matching feature can be directly calculated and compared to other features within the same approach [Soon et al., 2001, *p.* 534f.].

Random Forest

Boosting as well as bagging can be characterized as ensemble methods since they both use multiple learning attributes, either in the form of multiple models or datasets. Another important ensemble method, and a more often referenced method in later parts of this thesis, is the random forest algorithm [Ho, 1995; Breiman, 2001]. Expanding from the previous approaches, random forests introduce random feature selection combined with a bagging approach, i.e., random features are calculated for each subset of the data, increasing robustness across classification and regression tasks [Breiman, 2001, *p.* 21ff.]. In the end, the multitude of trees comprises the forest, each of them contributing to the result by voting, in case of a classification task [Breiman, 2001, *p.* 5f.], or by averaging across the predictions, when working on a regression task [Breiman, 2001, *p.* 25f.].

Naive Bayes Classifier

The previously mentioned machine learning models are rooted in the tradition of decision trees. However, Breiman, most known for the introduction of bagging [Breiman, 1996] and random forest procedures [Breiman, 2001], observed similar performance for classification tasks for a Naive Bayes classifier [Breiman, 2001, *p.* 23] and further noted the possibility of even categorizing a random forest system "[...] as a Bayesian procedure" [Breiman, 2001, *p.* 29]. Thus, a closer look at other traditional machine learning techniques seems natural, Naive Bayes being the first representative. The Naive Bayes classifier, also a supervised learning method, is based on Bayes' theorem, dating back to the 18th century [Bayes, 1763]. The principles of calculating probabilities based on a num-

[11] Both algorithms can also be characterized as voting algorithms, which are still in use in a multitude of facets within machine learning and NLP research [Henrich and Wegmann, 2021; Jia et al., 2023].

ber of conditions have been applied for classification purposes since the 1960s, e.g., in the context of indexing and search library systems [Maron and Kuhns, 1960]. The independence assumption of Naive Bayes classifiers is noteworthy here, i.e., features that represent a certain text are assumed to be independent of each other and attributes such as word positions are of no importance for this type of classifier [Jurafsky and Martin, 2024, *p.* 59]. Thus, correlations between features are irrelevant for the models' probabilities and the BoW representation is the usual feature space for the Naive Bayes approach [Eisenstein, 2018, *p.* 17ff.].

Generative vs. Discriminative Classifiers

Naive Bayes classifiers can be categorized as generative classifiers, i.e., a model is built up that can generate synthetic data, thus providing a class candidate [Bishop and Nasrabadi, 2006, *p.* 196ff.]. In contrast, discriminative classifiers are the other type of category of classification systems. The name of this category is nearly self-explanatory since features are identified that can help in discriminating between possible classes for a certain input without necessarily modeling the distribution of objects within each class [Bishop and Nasrabadi, 2006, *p.* 204ff.].[12]

Logistic Regression

The most famous approach within discriminative classifiers is logistic regression. It should be noted that the goal of logistic regression classifiers is classification instead of regression, as the name of the method suggests [Bishop and Nasrabadi, 2006, *p.* 205]. Logistic regression dates back to the 18th century when it was applied in the analysis of population growth, before resurfacing in the field of statistics at the end of the 1960s [Cramer, 2002] and being applied in NLP on word representation [Schütze et al., 1995] as well as machine translation [Berger et al., 1996] tasks. Classifiers using logistic regression analyze the input data, extract features, apply a weight and use the result within a sigmoid function in order to produce a classification probability [Jurafsky and Martin, 2024, *p.* 77ff.; Ng and Jordan, 2001].[13]

Support Vector Machines

Even though the advancements through the machine learning techniques mentioned above enabled new statistic and probabilistic approaches in NLP, some attributes of texts, and specifically longer texts, have made the processing of natural language challenging. One problem is the high dimensionality of approaches based on BoW text representa-

[12] The concept of generating and discriminating data will be of importance later in this thesis in the use case of text simplification as part of a model using Generative Adversarial Networks (GANs) and its main components, a generator and a discriminator, see Section 7.1.1.

[13] Also, logistic regression serves as one important component of many neural networks and thus can be seen as a precursor for many of the more complex and computationally expensive models in recent years [Jurafsky and Martin, 2024, *p.* 77].

tions, even though strategies exist, such as stop word filtering or a minimum occurrence factor of words in order to be considered as features in the vector space [Joachims, 1998, *p.* 138]. The creation of Support Vector Machines (SVMs) in the 1990s introduced a mechanism to handle high-dimensional feature spaces in text classification and other machine learning approaches [Cortes, 1995].[14] With SVMs, the potential high number of features is of lesser importance, due to their characteristic of generalizing during the learning process, i.e., being independent of the dimensionality of the feature space [Joachims, 1998, *p.* 138f.]. In essence, SVMs calculate hyperplanes that separate vectors with potentially high dimensionality into classes, and thus can classify text that is represented by those vectors [Kudo and Matsumoto, 2000, *p.* 142ff.].

As will be discussed throughout the thesis, these older machine learning techniques can still be found in current NLP research. Such observations are illustrated in Jegan and Henrich [2025a], Section 4.1 and use cases such as Pulver [2023] for decision trees and Naive Bayes classifiers, Raab [2023] for SVMs, and Hebeis [2025] for decision tree learning and random forests.

2.3 Rise of Embeddings and Neural Network-Based Methods

Role of Semantics

One area of linguistics, which was not directly included as the main component of any of the previously mentioned approaches, is semantics. The meaning of words is indirectly included in characteristics such as dictionary- and synonymy-lookups as part of decision trees, but not as the main calculated aspect of a learning method. This changed at the turn of the millennium, when vector space embeddings and especially vector semantics emerged as an alternative representation type of text (for further processing).

Latent Semantic Analysis

The offspring of this school of thought can be traced back to the field of information retrieval and the experiments resulting in Latent Semantic Analysis (LSA). In LSA, techniques similar to the future embeddings concept were created using singular value decomposition on a term-document matrix [Deerwester et al., 1990; Jurafsky and Martin, 2024,

[14] The foundations of the optimized approaches currently applied as SVMs for machine learning purposes can be traced further back to the 1960s and the work in Vapnik [1963] for pattern recognition.

p. 129ff.].[15] In this system, low-dimensional representations were calculated in order to better differentiate the data (in this case documents), with the goal of improving information retrieval tasks. Previously problematic aspects such as synonyms were more successfully handled using these dimensions, however not yet fully providing an approach to more complex issues such as polysemy [Deerwester et al., 1990, *p.* 405f.].

N-Grams

A more generalized application of LSA concepts on language modeling was attempted through a distributional representation of words, capable of calculating similarity scores between them as well as probabilities based on neural networks [Bengio et al., 2000]. This development enabled a new way of processing text, going one step further compared to previous approaches. The context of words, i.e., the semantic frame the words are placed within in a sentence or section, as found in techniques such as previous Naive Bayes methods, was limited, often based on BoW representations. A slight improvement, depending on the use case, was possible by using n-gram-based representations of words, restricting the context to the value of n tokens. Thus, an enriched representation through n-grams was the first step in order to include the context in the processing of such techniques, while vector space embeddings, as introduced in neural language modeling, provided the next iteration.

Curse of Dimensionality

The introduction of neural network architectures was the next step to handle *the curse of dimensionality* [Bengio and Bengio, 1999, *p.* 400], especially regarding computational resources. Two developments were central to this approach: First the introduction of the distributed representation for each word, i.e., textual regularities and patterns [Mikolov et al., 2013], enabling similarity comparisons between them. Second, an improved probability function based on neural networks in order to process word sequences through the distributed representations [Bengio et al., 2000, *p.* 1ff.]. The semantic and syntactic relations between words are captured with feature vectors, which, in contrast to previous approaches, are of substantially lower dimension than the size of the vocabulary. These features are based on the frequency of words in a certain context with other words, following the assumption that similar words appear often in the same contexts [Bengio et al., 2000, *p.* 5].

Downstream Tasks

These advancements improved the state of the art for word prediction tasks, before embeddings were more generally applied on other NLP tasks

[15] The term *embedding* was first used in this LSA context and refers to the mapping between different vector representations, which differs from the current usage of the term as a dense vector representing a word in latent space, see below in this section when discussing word2vec [Jurafsky and Martin, 2024, *p.* 131].

[Collobert et al., 2011]. The innovation of producing a neural network that can represent text in a way such that different text processing applications are possible without specifically designing the architecture for the downstream task can be seen as a predecessor of modern language models, which are of general purpose and can be applied to many downstream tasks.

Word2vec

Word2vec, as introduced in 2013, signified the next word representation breakthrough, that reduced the number of dimensions even further [Mikolov, 2013]. The innovation enabled by word2vec was embeddings, meaning dense vector representations of substantially lower dimension than previously possible. In contrast to BoW representations, dimensions do not directly correspond to words or phrases. The dimensions of a vector were therefore reduced to a number in the hundreds whereas previously dimensions could easily reach tens of thousands [Jurafsky and Martin, 2024, *p.* 109f.]. Instead of sparse values in the vectors, due to, e.g., term-document matrices and the co-occurrence of words with each other resulting in a large rate of zero entries for each vector, virtually all values in the vectors of a word2vec vector space representation are now filled with non-zero real numbers. Consequently, the reduced dimensionality resulted in a speed up when creating the vector space embeddings since the number of learnable weights was also decreased accordingly when compared to previous word representations, in addition to reduced memory usage and the ability to compute semantic similarity [Jurafsky and Martin, 2024, *p.* 117ff.].

Neural Networks

In parallel to the rise to prominence of word2vec, research into other techniques was also pushed further due to enhancements in processing capabilities through modern hardware. Neural networks had been worked on since the early 1990s. However, deep neural networks, characterized by a large number of neural processing layers, were now rapidly transforming the research field of NLP. This development is a turning point,[16] due to the computational complexity and resource demands introduced by deep learning approaches. The combination of word embeddings with neural networks led to the increase in hardware requirements and thus started the race towards ever larger models, still ongoing for present-day LLMs. A categorization between traditional and modern techniques, and in-between such as word2vec, is a complex problem and will be attempted in Sections 2.5 and 8.6.

[16] Not only in general for NLP, but also for the scope of this thesis.

Static vs. Contextual Embeddings

Still, word2vec as a landmark for signifying a shift in approaches should be acknowledged here, not only due to the use of the term *word embeddings*.[17] One of the key characteristics of this shift is the use of word embeddings as just the first step, i.e., as input representation, now called pre-training [Jurafsky and Martin, 2024, *p.* 145ff.]. This processing step can be applied either by using static embeddings, e.g., word2vec, or contextual embeddings, most prominently by Bidirectional Encoder Representations from Transformers (BERT), see below. For static embeddings, one vocabulary word is represented by one vector in the vector space, whereas contextual embeddings can have different vectors depending on the context of the word.

Transformer Architecture

Word2vec can therefore be seen as a simpler variant, and thus an initial step towards more complex and computationally intensive techniques, even when considered only as a pre-training step. While many deep learning techniques achieved impressive results across the past decade – exemplified by Recurrent Neural Networks (RNN) and its extension, the Long Short-Term Memory (LSTM) network [Hochreiter and Schmidhuber, 1997] – the focus of this thesis does not lie in the details of these neural networks. The transition towards LLMs, however, requires the short introduction of one more technique, the Transformer architecture [Vaswani et al., 2017].

Encoder-Decoder & Attention

As prerequisite for the transformer architecture, the encoder-decoder architecture as well as the attention mechanism should be briefly mentioned. The former structures an NLP problem into two tasks, i.e., first, creating a transformed version of the input data with the encoder as a contextualized representation, and second, feeding this representation into a decoder that generates the output, which is adapted depending on the desired task [Kalchbrenner and Blunsom, 2013; Cho et al., 2014]. Furthermore, attention was introduced as another groundbreaking feature for language modeling [Graves, 2013; Bahdanau et al., 2016]. Attention added the capability of incorporating information from all positions of the input as well as output sequences more evenly than was previously possible, thus enabling the modeling of dependencies and references within longer sections of text [Vaswani et al., 2017, *p.* 6001f.]. Encoder-decoder models in combination with attention were already present in LSTMs and other networks with recurrent mechanisms. However, the advantage of the transformer architecture of easier parallelized computation, which was the result of a reduced complexity when building the attention lay-

[17] And other embeddings such as GloVe [Pennington et al., 2014] and FastText [Bojanowski et al., 2017] among others.

ers,[18] led to a quick adoption of this architecture for many NLP tasks [Jurafsky and Martin, 2024, *p.* 184ff.].

BERT

While the transformer architecture proved to be highly influential for generative tasks, further solidified by their use in modern LLMs, another modern architecture is commonplace in many NLP tasks and should be briefly mentioned since its applications as well as comparisons to BERT will be relevant in several parts of this thesis. The architecture in question is the masked language model, implemented as a bidirectional transformer encoder model, most prominently implemented in the BERT model [Devlin, 2018]. The major difference compared to transformer-based architectures, which, very broadly, generate new tokens based on previous tokens, is the capability of including both past and future contexts of a masked token in the processing. With this capability of using left and right contexts of a text, contextual embeddings are created, with a focus on a deeper and richer understanding of the information presented in the analyzed text [Devlin, 2018, *p.* 4173f.]. Various downstream tasks are the focus of such embedding-based models, separating these models from transformer models mainly used for text generation, especially in the present LLM-focused NLP landscape. Contextual embeddings, in which the importance of the whole context surrounding some token can influence the generated output, especially in the form of BERT and its offshoots such as RoBERTa [Liu, 2019], were an important innovation for many NLP tasks, including question answering, classification and Named-Entity Recognition (NER) [Devlin, 2018, *p.* 4175ff.].

Shift towards LLMs

Once again, like the earlier models from Section 2.2.2, models based on word2vec and other static embeddings have been applied in use cases discussed below.[19] While the models mentioned in this section already used impressive amounts of data for training, the use of giant datasets in order to create language models thus far not seen in terms of scale prior to 2019/2020 was a paradigm shift. The innovation of modeling language and knowledge about our world through vast amounts of information and data started a trend that is continuing as of 2025. A closer look at LLMs is necessary to contextualize the further parts of this thesis.

[18] For transformer-based architectures, an adapted variant of attention is mostly used, called self-attention, which focuses on a single sequence and the relationship of tokens within that sequence [Cheng, 2016].

[19] See Pulver [2023] for contextualized embeddings such as BERT, other transformer-based systems for many text summarization tasks, see Gottwald [2023], Harms [2024] or Dal Cin [2025], and for many further tasks, see Jegan and Henrich [2025a] and Section 4.1.

2.4 ChatGPT and the Advent of LLMs

The emergence of LLMs in late 2022, specifically prompt-based LLMs, introduced NLP applications to a wide audience. Prominent examples like ChatGPT[20] applied prior developments, as detailed in the previous Section 2.3, at a larger scale and moved the application into a new user interface.

Pre-Training

The evolution towards such general-purpose interfaces for all kinds of NLP applications necessitated a reconsideration of established techniques, mainly pre-training, as well as some advancements within language model research. Pre-training, as already clearly stated in the paper presenting BERT [Devlin, 2018, *p.* 1ff.], lies at the core of large and influential language models. Prior to BERT, pre-training was already at the center of word2vec models, see above in Section 2.3, in order to learn representations of words or phrases (or even whole sentences in further model derivations) within a high-dimensional vector space, intended to define semantic relatedness. However, with larger datasets, pre-training enables new generalizable architectures without task-specific adaptations during training.

Hardware Demands

The size and scale of modern datasets, usually based on crawls of a huge number of websites,[21] paved the way for unprecedented training runs. The GPT-3 model, famously being the basis for GPT-3.5 and the first versions of ChatGPT [Wu et al., 2023, *p.* 1123f.], required over 1,000 Nvidia A100 Graphics Processing Units (GPUs) running for over a month in order to train the model [Brown et al., 2020; Narayanan et al., 2021, *p.* 2f.].[22]

Conditional Generation

Despite the increase in training costs, in time and hardware dimensions, a new representation of information within text is now possible to be captured by such models. Due to the restricted interpretability of LLMs and of deep learning models in general, it is still unclear how much knowledge about the world in general can be externalized through the text in the training data [Roberts et al., 2020]. Still, the capabilities of LLMs, especially as a tool which can be adapted for a myriad of use cases and applications, are demonstrably impressive. Generation of text in general

[20] See `https://openai.com/index/chatgpt/` (all web resources last accessed on: 11.02.2026).

[21] One of the most famous being the Common Crawl dataset, as of 2025 with over 2.5 trillion websites crawled, see `https://commoncrawl.github.io/cc-crawl-statistics/`.

[22] OpenAI generally does not publish details for the training runs of their models, but approximations due to model size can be applied and thus rough estimates can be calculated.

as well as classification, extraction and other tasks can all be approached with LLMs through conditional generation, i.e., a request as a prompt can include all the input needed in order for the LLM to receive the request as a word prediction task [Jurafsky and Martin, 2024, *p.* 204ff.]. For text generation tasks, the prompt can simply be the task description (e.g., "Summarize the following text"), while tasks such as question answering can be defined by providing a template within the prompt (e.g., the prompt starting with "Q: " followed by the question and concluding with "A: ").

Retrieval-Augmented Generation

The last example is the first step towards one of the most researched recent trends, Retrieval-Augmented Generation (RAG), which adds domain knowledge through a retriever, analogous to information retrieval and its documents that are retrieved based on a search query, to the context of the LLM [Lewis et al., 2020]. Thus, the initially static data,[23] which an LLM is trained on, can be enriched by relevant additional and potentially more up-to-date information and in the end can help in preventing hallucinations from appearing in the generated texts [Lewis et al., 2020, *p.* 9465ff.]. While adaptations of RAG approaches at first required architectural changes, new variations included the contextual information into the prompt, or rather the input for the LLM itself [Ram et al., 2023].

Prompt Engineering

One further aspect related to the usage of LLMs, due to the enormous attention it received within NLP research and applications, is the study and optimization of prompts itself, commonly referred to as prompt engineering [Liu et al., 2023]. While templates for various tasks have been identified, see above in discussing the conditional generation of requests, task-dependent notations and templates were studied as well as changing the prompt usage itself, e.g., by automated template learning or multi-prompt learning [Liu et al., 2023, *p.* 5ff.]. The optimization of prompts is however hindered by several factors, such as complex evaluations across different LLM models due to size and architecture differences or the fact that updates or changes within ChatGPT or other LLMs are often not communicated to users by the providers. This black-box nature regarding parameters, training data and other facets of the models impedes the reproducibility of results. As has been noticed multiple times already, updates from LLM providers can be applied without prior notice and are prone

[23] Although beginning with OpenAI and ChatGPT in late 2023, LLMs were able to access sources from the internet and thus are no longer restricted to trained and thus static data, see `https://www.bbc.com/news/technology-66940771/`.

to breaking workflows.[24] Additionally, changes in style and quality of the texts produced via the chat interface were noted and discussed similarly.[25]

Chat-Based LLMs

While the problems of LLM evaluation have now been mentioned, and will be further discussed across this thesis, e.g., in the respective sections within NLP applications in Chapter 3 and the observed use cases in Chapter 7, one major success is still remarkable, i.e., the ease of use of LLMs for new users. A previous problem for NLP tools based on deep learning was the lack of reproducibility of results as well as the lack of documentation or even the difficulty of installing the system due to non-public datasets, dependency problems or other hindrances [Arvan et al., 2022; Aumiller et al., 2023]. While traditional models require only some rudimentary programming skills, more advanced models such as BERT need the necessary hardware requirements to be fulfilled in order to run the models, not to mention the requirements when training or fine-tuning. Still, even the necessity of running local scripts for simple classification or data extraction tasks was prohibitive for users without prior NLP knowledge. Therefore, the interaction via prompts is the biggest advantage of LLMs. This development can be called transformative. Chat-based interaction enables any user accessing the model through its web interface or via a Command-Line Interface (CLI) to execute NLP tasks on their own, without installing any dedicated task-specific software.

Local LLMs

The previously referenced LLMs are usually operated by one of several large corporations or providers, such as OpenAI, Google, Anthropic or Mistral. Both the use of the Application Programming Interface (API) and the chat-based interaction are dependent on these corporations, their services and web servers, with server overload and availability in general as potential error sources. An alternative is locally run LLMs with one of the first major releases of such models being LLaMA[26] by Meta [Touvron et al., 2023]. The motivation for releasing this family of models is

[24] One such observation has been widely documented for the release of GPT-5 in August 2025, see, e.g., `https://www.wsj.com/tech/ai/openais-rocky-gpt-5-rollout-shows-struggle-to-remain-undisputed-ai-leader-04897686/`. The release and accompanying removal of the previous GPT-4o model broke workflows for users due to different model behavior and also changed the billing procedure preventing previous processes from completing, which comprise just two of the documented problems for this release.

[25] See, e.g., `https://arstechnica.com/ai/2025/04/openai-rolls-back-update-that-made-chatgpt-a-sycophantic-mess/`.

[26] These models are available in different sizes, usually referred to by the number of parameters such as *70B*. This abbreviation refers to seventy billion parameters, e.g., *LLaMA 3.1 70B*, see `https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_3/`.

to provide users and researchers data to experiment upon and to study the models.[27] While hardware demands are a restrictive factor in executing these local models, often requiring dedicated GPUs, the availability of LLaMA, its successors provided by Meta[28] and competing open weights by Mistral[29] or DeepSeek[30], are an important milestone in enabling researchers and users to experiment and in general use LLMs. This measure thus benefits those who are not able to pay the fees by LLM providers or who are not willing to conform to closed source systems. More details regarding the hardware requirements and restrictions of locally run LLMs in contrast to models hosted by larger providers are given in Tuggener et al. [2024].

This brief presentation of LLMs shall suffice as an introduction to this family of models. LLMs shall further serve as a contrast to the other class of models, which both together represent the main theme of this thesis, i.e., the analysis of traditional models in contrast with modern models. Before a definition of traditional models will be proposed in Section 2.5, a brief review of the current chapter will be given as a timeline.

Timeline

A collection of models and architectures from this chapter is visualized in Figure 2.1, accompanied by a rough categorization aligned towards Sections 2.2 to 2.4 from this chapter. In order to keep a clean layout, some concessions were made, e.g., the exclusion of prominent extractive text summarization techniques like LexRank [Erkan and Radev, 2004], which will be featured throughout this thesis, see Sections 3.6, 8.1 and 10.1, or the traditional text segmentation approach of TextTiling [Hearst, 1997], which will play a major role in Chapter 9. Furthermore, rule-based methods, e.g. as defined by Karlsson et al. [1995] in the 1990s, are not directly referenced in the timeline, but they can be traced back to context-free grammars, see Section 2.2.1. A clear separation of categories is also not always possible, e.g., for Naive Bayes classifiers, which can be categorized as machine learning systems used for text classification tasks [Koller and Sahami, 1997].

[27] See the initial blogpost at `https://ai.meta.com/blog/large-language-model-llama-meta-ai/`.
[28] See `https://www.llama.com/docs/overview/`.
[29] See `https://platform-docs-public.pages.dev/getting-started/models/weights/`.
[30] See `https://api-docs.deepseek.com/`.

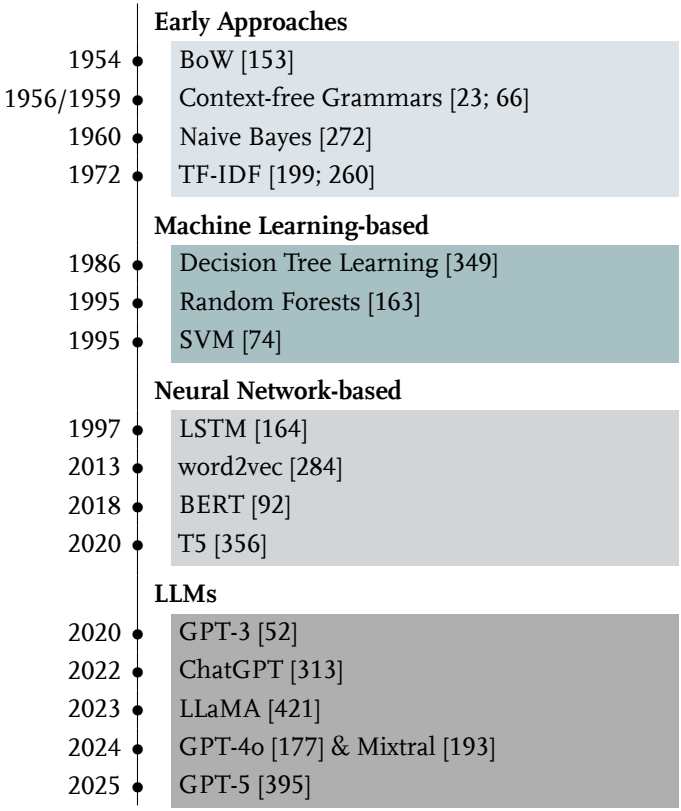


Figure 2.1: Timeline of NLP approaches.

After presenting an overview of early and modern approaches in Figure 2.1, a first definition of traditional models will be given next.

2.5 Traditional Techniques: An Attempt at an Early Definition

NLP History

Before applications are detailed in Chapter 3, a closer look towards traditional models is necessary to contextualize the techniques used there within the broader realm of NLP applications. When looking back at the development of NLP, starting from the early linguistic foundations (as seen in Section 2.2) up towards LLMs (as seen in Section 2.4), noteworthy shifts have occurred in how text processing as a computational task is

approached. A major factor for these shifts has been the increase in processing capabilities and thus the ability of processing large amounts of (textual) data.

Vagueness of Terminology

While the rise of neural networks and more complex language models has already been described, a clear separation between either simple and complex models or between traditional and modern techniques is not easily done. However, some common characteristics need to be identified that allow a systematic categorization of techniques based on shared attributes. These characteristics will be mentioned here briefly, before they will be discussed in later parts of this thesis in Chapters 8 and 10 and Section 9.3.

Features of Traditional Models

The four tentative characteristics of traditional models are: Reproducibility, efficiency, explainability and documentation. *Reproducibility*, especially in the age of LLMs, is a key factor when talking about the definition of *traditional techniques* in this thesis, i.e., the capability of producing the same output given the same input and the same parameters and setup, see Sections 6.4.1, 7.4.1 and 8.2. *Efficiency*, as shortly referenced above, is another factor, which can be approached from different angles. One dimension is efficiency in terms of time, meaning execution time (and also training time, if the applied systems are trained individually), see Sections 6.4.3, 7.4.3 and 8.1.1. Another dimension is cost, meaning certain hardware requirements such as dedicated GPUs in order to train and use LLMs more efficiently, see Sections 6.4.3, 7.4.3 and 8.1.2. Costs can also be seen as licensing, subscription fees or fixed costs in terms of API requests or token counts, when looking at using hosted LLMs at providers such as OpenAI. *Explainability*, see Sections 6.4.2, 7.4.2 and 8.4, as well as *documentation*, see Section 8.3, are further dimensions that can be applied here. Due to the simpler structure of traditional techniques – at least in most cases – and their long-standing application in use cases, a wide array of tutorials, handbooks or other documentation exists for a variety of programming languages. The simpler structure of traditional techniques makes them highly explainable due to, e.g., the case-based nature of decision trees for information extraction tasks or extractive approaches for text summarization purposes.

Examples of Traditional Models

The techniques just mentioned, decision trees and extractive summarization approaches such as LexRank [Erkan and Radev, 2004], are two examples of traditional techniques. They all exhibit the characteristics mentioned above. Other older techniques exhibit most, rather than all characteristics. Two examples are SVMs, which are only explainable in a certain sense, see Núñez et al. [2002], and random forest classifiers, which are

not easily reproducible if the random seed, with which those classifiers are initialized, is not known. Still, both techniques would be classified as traditional for this thesis. A borderline case is word2vec and vector representations that were introduced after word2vec. The architectural setup of word2vec, based on the vector space embedding with reduced dimensions, allows for comparatively efficient computation, which would be fitting for the efficiency criteria. However, techniques after the introduction of word2vec and other related techniques use such embeddings in many systems only as initializing steps as a form of pre-training. Categorizing word2vec as a standalone system is ambiguous regarding the inclusion as a traditional technique. Other, more complex systems using word2vec only as part of a larger system would surely not be seen as traditional due to reduced efficiency, lacking explainability and other drawbacks, see below in Chapter 6. A more comprehensive discussion on the applicability of traditional techniques, and their actual use in practice, will be done in Chapter 10, where more concrete examples will show what kind of techniques are considered *traditional* for this thesis. A refined definition after an analysis of challenges, observations and evaluations as well as use cases will be given in Section 8.6, which will further discuss and contextualize the criteria given in this section.

3 NLP Applications

After having displayed the earlier as well as current NLP landscape in Chapter 2, this chapter will present an overview on core techniques in use for different NLP tasks in Section 3.1, before brief descriptions on application scenarios including historical contexts, approaches and evaluations will be introduced. The tasks specified are: Information and relation extraction (in Section 3.2), text classification (in Section 3.3), text segmentation (in Section 3.4), text simplification (in Section 3.5) and text summarization (in Section 3.6). While the state of the art will be shortly reported, a closer look at traditional techniques in the respective application scenarios will also be given, since they, i.e., the traditional techniques, will be contextualized in later parts of this thesis in terms of applicability in the present NLP landscape.

3.1 Overview of NLP Applications

Some parts of this subsection have been inspired by supervised theses, in particular the selection of models, see Dal Cin [2025], Fruth [2022], Gottwald [2023], Harms [2024], Pulver [2023] and Raab [2023].[31]

Reference to NLP History

The development of NLP across the past 80 years, when starting with context-free grammars from Chomsky [1956] or statistical n-gram analysis from Luhn [1957], has seen several paradigm shifts conceptually. Transitions from linguistic and syntactical foundations towards statistical, probabilistic and later neural network-based methods, see Chapter 2, have occurred several times and have impacted widely differing applications within NLP. The shift towards LLMs, which has impacted the whole research field, is thus just the latest in a number of extensive development leaps.

[31] References to the theses are listed in the Bibliography, including an Uniform Resource Locator (URL) to the thesis if applicable, hosted at the research information system of the University of Bamberg, see `https://fis.uni-bamberg.de/`.

Chomsky

Different trends of linguistic and computational linguistic development can easily shift the dominant way of thinking towards one kind of approach, while marginalizing previous methods. One such example would be the influence of Chomsky's work in the 1950s [Chomsky, 1956; Chomsky, 1976] and 1960s [Miller and Chomsky, 1963]. This work supplanted the previous statistical-based processing of information by using n-grams and Markov models [Jurafsky and Martin, 2024, *p.* 52f.]. Instead, Chomsky and colleagues claimed that Markov models and statistical methods were incapable of representing human language, while proposing the Chomsky hierarchy [Chomsky, 1956]. This hierarchy formalized language within four classes of grammar including concepts such as finite-state automatons, which were highly influential. The shift described in the 1950s and 1960s in terms of scope is similar to the shift in recent years towards LLMs.

Machine Translation

When narrowing the scope, a task-specific perspective is helpful in showing the evolution of NLP applications. The evolution of a concrete application within NLP will be briefly presented through the research progress achieved in machine translation. This application scenario was chosen, since it will not be explicitly approached as a main NLP task for this thesis but is still one of the premier use cases for text processing in general. Statistical machine translation using machine learning algorithms has been in use since the 1990s, in which extraction rules were identified by processing parallel corpora of the two respective languages [Lopez, 2008]. Earlier machine translation systems can be grouped into direct translation approaches, transfer approaches and interlingua approaches [Jurafsky and Martin, 2024, *p.* 286f.]. First, direct translation approaches apply the translation simply by processing each word through bilingual dictionaries. Second, transfer approaches apply translation rules through a parse tree. Finally, interlingua approaches extract an abstract meaning representation from the source text and generate the translated text from the interlingual representation. Challenges with these models and statistical machine learning approaches in general were recognized as limited word alignment and the dependence on translation rules. Both issues severely impact the quality of the translation and were later addressed by using neural machine translation [Dabre et al., 2020, *p.* 2].[32]

[32] Neural networks based on RNN, Convolutional Neural Network (CNN), self-attention or feed-forward architectures have all been used for machine translation before the shift towards LLMs [Dabre et al., 2020, *p.* 5].

LLMs for Machine Translation

Ironically, language models[33] themselves have become the major contributor to the more recent shift in the NLP landscape, particularly through the emergence of LLMs. This development also affected research in machine translation. By scaling several dimensions up, such as data, compute power and model parameters, tremendous achievements have been reached since the release of LLMs. These advances became widely visible with the release of ChatGPT [Wei et al., 2022] in 2022. Just as one example, the shared task on general machine translation from the conference on machine translation (WMT) in 2024 has found that nearly all submissions have been using LLMs as part of their setup, in most cases applying LLMs through fine-tuned LLaMA models [Kocmi et al., 2024, *p.* 8]. As one successful example from the 2024 shared task, Liao et al. [2024] use continuous pre-training of a LLaMA model on monolingual data to enhance the capabilities of the model within one language and then fine-tune on high-quality parallel data for the translation task. The special consideration of multilingual LLMs has been a focus of Artificial Intelligence (AI) providers as well as in research. This is evident in the release of Mixtral[34] as an LLM with explicitly multilingual capabilities and in studies like Rathje et al. [2024] as well as surveys such as Xu et al. [2025].

Related Research Fields

Apart from the major NLP task of machine translation and other tasks that will be presented below starting with information and relation extraction in Section 3.2, many more could be mentioned here. An exhaustive list is certainly not possible, but information retrieval and clustering are two broad research fields as well as commercial avenues, which both should be mentioned, since they have close ties to other NLP tasks discussed in this thesis.

Information Retrieval

Information retrieval has been mentioned briefly in Section 2.3 during the introduction of neural networks for NLP approaches. There, the advancement of LSA approaches was noted for its initial application on retrieval tasks, which in turn led to further developments for other text processing applications. Furthermore, techniques such as traditional models like TextRank [Mihalcea and Tarau, 2004] for text summarization scenarios – which will be referenced several times throughout this thesis, see Table 4.1 and Sections 3.6, 4.1 and 10.1 – is based on the information retrieval ranking algorithm PageRank [Brin and Page, 1998; Page

[33] The term *language model* was introduced in Jelinek et al. [1975]. In their paper, statistical techniques using n-grams were resurfaced in a speech recognition model, which recognized language models as encompassing more information such as grammar and semantics when compared to simple n-gram approaches [Jurafsky and Martin, 2024, *p.* 53].

[34] See `https://mistral.ai/news/mixtral-of-experts/`.

et al., 1999]. This algorithm is best known as a core component of the Google search system. Historically, many different techniques were applied to information retrieval: Simple pattern matching approaches and Boolean retrieval methods were created early on before vector space models allowed for more complex document representations [Henrich, 2008, *p.* 137ff. and *p.* 185ff.]. Probabilistic retrieval models, with BM 25 [Robertson et al., 2009] as one of the most famous ranking functions currently available, are further noteworthy [Henrich, 2008, *p.* 263ff.]. Nowadays, when discussing information retrieval in a web search context in particular, the role of classical information retrieval algorithms in contrast or in conjunction with LLMs is unclear and part of ongoing research. In particular, it is uncertain whether chat-based LLM systems will replace traditional search systems or complement them [Hersh, 2024].

Clustering

Another, potentially even broader research field, is clustering. Similarities are certainly possible between classification and clustering approaches. For details on the former application. see Section 3.3. A distinct difference can be found in the initial goal and dataset. Whereas classification tasks usually concern a fixed number of classes that are predetermined [Aggarwal and Zhai, 2012a, *p.* 163ff.], clustering scenarios do not share this feature of having a controlled number of classes. Instead, they attempt to extract groups of similar objects that share some common features [Aggarwal and Zhai, 2012b, *p.*77ff.]. Concrete clustering approaches are diverse and numerous. They range from traditional methods such as partitioning-based methods – most famously implemented as the k-means clustering algorithm proposed in the 1960s [MacQueen, 1967] – to hierarchical-, density- or grid-based methods. For overviews of approaches not based on neural networks, see Xu and Tian [2015] and Nguyen et al. [2015]. The influence of LLMs for this research field is maybe not as all-encompassing as in other NLP tasks, but studies, comparative analyses and use cases using LLM-assisted clustering approaches have also been conducted [Keraghel et al., 2024; Viswanathan et al., 2024].

Scope of Information Retrieval & Clustering

Broadly speaking, information retrieval as well as clustering scenarios are not limited to textual data as their input dimension. Meaningful advances within their respective research fields have been achieved in different disciplines. In multimedia retrieval, for example, multimodal semantic models are capable of processing images, video and text or a combination of these data types [Zhu et al., 2023]. Clustering techniques have also seen progress in areas such as bioinformatics or object recognition [Ezugwu et al., 2022]. The focus of this thesis will remain on textual data, but the capabilities of both application scenarios briefly presented here

are noteworthy, mainly enabled through the use of neural networks [Hambarde and Proenca, 2023; Oyewole and Thopil, 2023].

Attempting an Overview

The processing of text is a broad research field, which prevents a comprehensive and all-encompassing discussion on all techniques and methods from the past and present. Instead, this thesis will attempt to select the techniques that are prevalent for the use cases discussed starting with the next Section 3.2. Even then, heavily researched application areas such as text classification and text summarization are too large to comprehensively review. The focus will be therefore on a snapshot of current techniques of the past several years, largely defined by LLMs, and in contrast the selection of traditional models, outlined above in Section 2.5.

Method Compilation

Following this methodology, two main sources were used in gathering the methods and techniques. First, as a slight lookahead to the SLR covered in Section 4.1, the survey presented in Jegan and Henrich [2025a] will serve as a basis in both representing state-of-the-art methods as well as traditional models still in use today for all use cases except text segmentation.[35] Second, the experiments conducted for this thesis as well as in supervised students' theses will offer an overview of applied methods from not only a literature review perspective, but also from a practical standpoint. All use cases described in this thesis include experiments in adapting and executing methods on the respective scenario. Therefore, methods were selected and further details can be found either in the respective section of this thesis, the SLR in Jegan and Henrich [2025a] and the referenced sources:

- *Information and Relation Extraction:* See Section 3.2 and Pulver [2023].
- *Text Classification:* See Section 3.3 and Raab [2023].
- *Text Segmentation:* See Section 3.4 and Jegan and Henrich [2025b].
- *Text Simplification:* See Section 3.5 and Fruth [2022].
- *Text Summarization:* See Section 3.6 and Dal Cin [2025], Gottwald [2023] and Harms [2024].

After presenting an overview of major tasks within NLP such as machine translation, clustering and information retrieval, which will not be further addressed in this thesis, the applications listed just above will be discussed in the next section.

[35] Due to the limited scope of the SLR, the "smallest" application scenario was not analyzed there.

3.2 Information and Relation Extraction

Common Features

Both information extraction and relation extraction share many common characteristics. They differ, however, in one key aspect: The number of cohesive units connected to a single relation. Thus, entities produced by information extraction techniques can be semantically independent of surrounding parts of text. In contrast, entities detected by relation extraction consist of two or more connected units of text. Still, the commonality between both information and relation extraction is the identification of information based on some class or criteria.

Micro-Reading vs. Macro-Reading

While many NLP tasks focus on the exact processing token by token, meaning a concentrated effort to analyze each sentence, information and relation extraction are allowed to only focus on the relevant entities and thus can skip sentences without such entities. The first type of tasks can be characterized as micro-reading, while information and relation extraction systems, and others like them, can be seen as macro-reading [Eisenstein, 2018, *p.* 404f]. The categorization into macro-reading results in a different goal for such approaches, especially when compared to tasks such as simplification or summarization, where each token is analyzed.

Relation extraction in particular is connected to further research branches such as linked data, knowledge graphs and template filling. The focus will however remain on the two applications information extraction (in Section 3.2.1) and relation extraction (in Section 3.2.2).

3.2.1 Information Extraction

Named Entities

The NLP task information extraction includes previously mentioned applications such as NER, but also the identification of custom entities, defined or declared by and for the task they are relevant for. Named entities – commonly places, people, time information and organizations – are typical representatives for NER. However, the distinction between information and relation extraction becomes less clear since knowledge bases such as WikiData are also clearly defined by the relations between entities therein.

Application & Evaluation

In practice, different linguistic properties are relevant for information extraction. Pattern matching by using regular expressions, analysis of syntactic structures and lexical analysis can all contribute when constructing an information extraction model [Grishman, 1997, *p.* 13ff.]. Evaluation for information extraction systems, and relation extraction as well, is

usually done by applying the metrics precision, recall and f1-score [Grishman, 1997, *p.* 21f.].

Early Systems

In terms of techniques or methods, early information extraction systems were realized by template filling approaches. First, the template structures were defined and then extraction rules were applied in order to extract the relevant information [Cowie and Lehnert, 1996]. Applying machine learning techniques yielded systems with SVMs or Hidden Markov Models (HMMs) as a basis, while the tagging schemes such as BIO tags[36] or other schemes were widely used [Grishman, 2012, *p.* 4ff.].

Modern Systems

In more recent years, following the shift introduced in Section 2.3, transformer-based approaches, both encoder-decoder-based transformer models and encoder-only models such as BERT,[37] represented the state of the art before the advent of LLMs. Through the capabilities of modern deep neural nets, new approaches and new datasets were created. For example, the Few-NERD dataset was introduced. It contains the common NER entities but also includes more fine-grained categories within the usual classes such as person or organization, e.g., the occupation of a person and the type of organization respectively [Ding et al., 2021]. The Few-NERD dataset was tested by Ding et al. [2021] with various approaches, mainly implemented with BERT as the backbone of models and further adaptations such as both supervised and few-shot settings and architectural changes like prototypical networks [Ding et al., 2021, *p.* 3203ff.].

Advanced Use Cases

While the previous paper tackled the widely known NER application, more advanced information extraction use cases were also successfully adapted through modern transformer-based approaches. One such example is the extraction of aspect-based sentiment information from review datasets on different domains such as restaurant and laptop reviews [Lv et al., 2023]. There, various categories depending on the dataset have to be analyzed in order to extract semantically relevant information for each aspect (e.g., "food quality" or "service quality" in case of the restaurant dataset). The authors applied the transformer-based model T5 [Raffel et al., 2020], widely used not only in information extraction use cases [Lv et al., 2023, *p.* 1012].

The transformative nature of LLM-based approaches was also noticed in the information extraction research field. An early analysis noted vary-

[36] The first token of a recognized entity is tagged with a "B", meaning the "beginning" part, while all following tokens of the same entity are tagged with an "I", i.e., "inside the entity" and all tokens that are not part of an entity are tagged "O", logically "outside" of entities [Ramshaw and Marcus, 1999].

[37] See Section 2.3 for more information on this distinction.

ing performancewhen compared to the prior state-of-the-art models that are mainly based on the BERT architecture, claimed however strong performance on a more open task definition in terms of accuracy and faithfulness [Li et al., 2023]. Faithfulness in this case relied on the responses given by ChatGPT, which was prompted to explain the provided result and if that explanation adhered to the original prompt and its context [Li et al., 2023, *p.* 6ff.]. A larger survey on current LLM usage for information extraction purposes displayed the competitive performance of many LLM-based methods on various domains and the near-total domination of the research field by LLMs [Xu et al., 2024b]. While relation extraction was also a part of the previous two publications focused on LLMs [Li et al., 2023; Xu et al., 2024b], relation extraction as an NLP task will be presented further in the next section. This separate consideration is due to the differences between the two tasks, especially when looking at more traditional approaches.

LLMs in Information Extraction

3.2.2 Relation Extraction

The NLP task relation extraction, as already mentioned, can be seen as an extension of information extraction. The relationship between different entities, e.g., modeled as triples within a knowledge graph, each triple consisting of two entities connected by one node or relation,[38] is one typical use case of relation extraction. Linguistic relationships such as hypernyms and hyponyms can also be modeled as relation extraction tasks, which capture a hierarchical part-of relation between a supertype (the hypernym) and the subordinate subtype (the hyponym).

Contrast to Information Extraction

The latter application, i.e., automatic detection of hyponymy, was an early typical NLP use case, achieved by using hand-crafted rules like

Early Systems

$$NP_0 \; such \; as \; \{NP_1, \; NP_2 \; ..., (and \mid or)\} \; NP_n$$

with NP being a noun phrase [Hearst, 1992, *p.* 539]. While hand-crafted rules[39] exploit the lexico-syntactic knowledge of typical entities through linguistic analysis, more complex relation types require other approaches. Early machine learning-based techniques represented relation extraction as a binary classification task, in which a pair of entities were either labeled as one class of a relationship, or not [Zhang et al., 2017a, *p.* 179]. SVMs and other machine learning classifiers were applied in this supervised set-

[38] Usually characterized as a subject-predicate-object relationship.
[39] Often referred to as *Hearst patterns* after the main author of Hearst [1992].

ting in order to build a model [Zhou et al., 2005], with advances by including, for example, dependency parse trees [Culotta and Sorensen, 2004]. Semi-supervised methods such as co-training were applied in the same manner as a classification task [Zhang et al., 2017a, *p.* 184ff.], but the problem of even more complex relation types arises once again.

Complex Relations

One such example of a more complex relation is event extraction with multiple entities such as location, organization and time information and even relations between events. This task can vary widely in terms of complexity and was modeled early on as a template filing or multi-class classification problem [Jurafsky and Martin, 2024, *p.* 446f. and *p.* 456ff.]. Similarly to information extraction, a shift towards deep learning-based approaches can be noticed here as well, starting with CNN- and LSTM-based techniques [Feng et al., 2018; Wang et al., 2023a].

Further Dimensions

While event extraction increases the amount and types of entities, other dimensions can vary for more complex relation extraction tasks. One such further dimension is the extent of the input that needs to be analyzed. In all previous examples, sentences or at most a few sentences were the focus of the extraction task. However, document-level relation extraction as a research scenario is a difficult problem for traditional relation extraction systems. The challenge for those systems is the distribution of relevant entities for the relationship across, at least, multiple sentences [Yao et al., 2019b]. More dimensions exist for relation extraction tasks, such as cross-lingual/multilingual [Faruqui and Kumar, 2015] or multimodal [Hu et al., 2023b] relation extraction applications. However, the focus here will remain on the simpler relation extraction between entities, introduced above.

Evaluation

Evaluation, as mentioned above for information extraction in Section 3.2.1, is usually quantitatively done using metrics such as precision, recall and f1-score [Cowie and Lehnert, 1996]. However, the standard way of evaluating with these three metrics necessitates datasets with accompanying labels for comparison. Oftentimes such labels are not available and thus semi-supervised approaches are implemented, which can be applied on datasets that do not include such labels. Both large-scale datasets and domain-specific datasets, with far less text, are afflicted by this issue. The evaluation for such datasets needs to be adapted accordingly. Usually, it is done by extracting a smaller subset of results from the extracted relations and manually labeling them. This process allows a qualitative evaluation and enables a quantitative summary based on the manual assignment of fitting relations [Etzioni et al., 2008, *p.* 71ff.]. The resulting values are then used to compute a precision score for this smaller sample. This score is

intended to be representative for the whole corpus, similar to the usual evaluation of supervised approaches. In contrast, recall and thus also f1-scores are not easily achievable in this manner [Bach and Badaskar, 2007, *p.* 12].

Advanced Approaches

The development not only of complex relation extraction but relation extraction in general also rapidly involved transformer-based approaches. One noteworthy implementation involved the prediction of spans – the identification and classification of contiguous tokens as, e.g., NER tokens – using an adapted BERT architecture [Joshi et al., 2020]. The task in this case, the prediction of the type of relation between subject and object, defined in the TACRED dataset as 42 types [Zhang et al., 2017b], was modeled as a downstream classification task. The more recent state-of-the-art approaches include more complex architectures to handle positioning problems inherent in previous systems. These architectures allow triples – which combine both entities and their relation[40] – to be extracted irrespective of their order [Sui et al., 2023].

LLMs in Relation Extraction

The rise of LLMs has also impacted relation extraction. Various prompt techniques were applied on temporal relation extraction and analyzed in different settings. These analyses revealed some drawbacks of prompt-based methods when compared to other transformer-based state-of-the-art approaches. LLM-based approaches also showed problems such as limitations with long dependencies between entities [Yuan et al., 2023]. A further study by Wadhwa et al. [2023] modeled the relation extraction problem as a conditional text generation task (see Section 2.4) for named entities and standard relations, which yielded near comparable results to state-of-the-art approaches. Wadhwa et al. [2023, *p.* 15567ff.], however, also note a shortcoming of evaluating prompt-based models for relation extraction. Because the generative nature of LLMs tends to produce results that differ from the requested result, comparisons with other techniques and the evaluation in general are complicated.

Both information and relation extraction tasks can and have been modeled as classification tasks, see the categorization of different techniques as part of the SLR from Section 4.1, especially in Tables 4.1 to 4.5. However, the base classification task – when discussing text classification – usually involves classifying a whole text into one or more categories. This application scenario will be presented in the next section.

[40] Similar to knowledge graph generation.

3.3 Text Classification

Brief Definition

The NLP task text classification revolves around the labeling of texts with a certain class value from a given set of different class values [Aggarwal and Zhai, 2012a]. Also known as text categorization, the classes or categories are usually pre-defined, which is the primary distinction contrasted with the task topic modeling or clustering, wherein topics are generated by analyzing texts themselves without providing pre-defined labels, see Section 3.1. Thus, text classification is usually defined as a supervised learning task, depending on a training dataset with texts and corresponding labels.

Use Cases for Classification

Application scenarios within text classification are diverse and are widespread in mainstream software systems. Spam detection is one of the main use cases, common in email programs and one of the earlier text classification approaches in use in consumer software [Sahami et al., 1998]. Sentiment analysis is another common application, used in determining the sentiment, i.e., a positive or negative stance of a text.[41] Closely related, and in the age of social media of larger importance, is the text classification approach for hate speech detection. This scenario can be implemented as a binary text classification task. The task is then to decide if one social media post can be categorized as hate speech, or not. Alternatively, it can be posed as a multi-class problem, in which the type of hateful comment can be categorized [Yin and Zubiaga, 2021]. The classification of news articles is another widely researched application scenario within text classification due to the prevalence of available datasets. Popular datasets 20 Newsgroups [Lang, 1995] or AG News [Zhang et al., 2015] are used as evaluation targets in many publications.

Dimensions in Text Classification

Different dimensions such as binary and multi-class classification provide a clear distinction for each classification application. However, when categorizing into multiple classes, it is also possible that the classifier assigns multiple labels to each text, which would be called multi-label classification [Mirończuk and Protasiewicz, 2018, *p.* 39ff.]. More dimensions such as hierarchical classification exist as well, but the focus in this thesis will be on multi-class classification.

Evaluation of Text Classification

For evaluating text classification systems, accuracy and error rate are common metrics in terms of quantitative analysis as well as precision, recall and f1-score once again [Yang, 1999, *p.* 74ff.]. For multi-label classification problems more advanced metrics such as micro-f1 and macro-f1, pre-

[41] Often found in the shared tasks held at SemEval, see `https://semeval.github.io/`.

cision at top-k or the Normalized Discounted Cumulative Gain (NDCG) are usually applied [Eisenstein, 2018, *p.*82f.]. In terms of quantitative evaluation metrics, the focus should not only be on one single metric, because one single score cannot fully characterize the performance of an approach. Instead, a combination of different metrics should be used [Yang, 1999, *p.* 81ff.]. Qualitative evaluation, as described above in Section 3.2.2 and also below for simplification (in Section 3.5) and summarization (in Section 3.6), is seldomly done for classification. A stronger focus is placed on other aspects of the classification pipeline, such as feature selection [Forman, 2003] or the applied preprocessing steps such as stemming or stop word elimination [HaCohen-Kerner et al., 2020].

Early Systems

Traditionally, classifiers were modeled through rule-based systems [Li and Liu, 2014] or, relatedly, based on rules in order to construct decision trees [Aggarwal and Zhai, 2012a, *p.* 176ff.]. Quick follow-ups included machine learning techniques to learn decision trees as classifiers – as was already laid out in Section 2.2.2 – through the methods of boosting and bagging in order to create either improved decision trees or use adapted data splits [Breiman, 1996; Freund et al., 1996]. Probabilistic and linear classifiers represent the continued evolution of classification approaches, starting with Naive Bayes classifiers. Binary classification tasks, such as spam detection, were applied early on using Naive Bayes. In this case, the classifier estimates the probability of a message belonging to one of two classes, i.e., regular mail or spam [Aggarwal and Zhai, 2012a, *p.* 181ff.]. Naive Bayes is easily adapted for multi-class classification problems through its probabilistic properties, e.g., for assigning the correct genre or newsgroup to a given text.[42]

One-Versus-All vs. One-Versus-Rest

Other linear classifiers such as SVMs were primarily intended for binary classification, which construct a hyperplane separating the two classes (also see above in Section 2.2.2). In order to also process multi-class problems, two strategies were introduced to handle this restriction, one-versus-all, also known as one-versus-rest in the literature, and one-versus-one [Eisenstein, 2018, *p.* 414]. One-versus-all focuses on each label, i.e., the model creates a binary classifier for each label, whereas one-versus-one deals with a pair of labels and a binary classifier is created for each of these pairs. Both variants produce confidence scores provided by all single classifiers, and the classifier with the highest confidence score is chosen, thus its predicted label is assigned. Still, this approach is prone to problems depending on the class distribution within the training data or in

[42] As in often used datasets for this use case, such as the 20 Newsgroups dataset or the AG News, see the beginning of this Section 3.3.

ambiguous regions when constructing the classifiers [Bishop and Nasrabadi, 2006, *p.* 183f. and *p.* 338f.].

Modern Systems

Deep learning systems have transformed the research field of text classification similarly to previously mentioned tasks. Due to the increased architectural size and complexity, many systems were proposed for the common problem of classifying texts, one being the previously mentioned RNN and other variants such as CNN and Graph Neural Network (GNN) [Li et al., 2022, *p.* 8]. Details for these systems differ, such as the focus on sequential information in RNN-based systems [Liu et al., 2016] or on the extraction of the most fitting features in CNN-based systems. However, all methods share the aggregation at the final layer in order to produce a prediction as classification result [Li et al., 2022, *p.* 11ff.]. GNN-based approaches differ due to its construction as a graph network, in which syntactic structure can be learned by the neural network through its connections between words, sentences and documents. This architecture allows the neural network to model the task as a graph node classification problem [Li et al., 2022, *p.* 16f.].

Transformer-Based Systems

While the different deep learning models were successful in capturing the attention of the classification community, transformer-based approaches were once more quickly able to surpass previous state-of-the-art systems [Li et al., 2022, *p.* 25ff.]. The previously mentioned advantages of contextual embeddings (see Section 2.3) and improvements in parallel computing served as the main advantages when adapting approaches based on BERT. Further optimized systems such as RoBERTa [Liu, 2019] and XLNet [Yang et al., 2019] build on these developments.

LLMs in Text Classification

When looking at architectures such as BERT and its offshoots, the adaptation towards classification as a downstream task happens in the final layers, where classifier layers are added on top of the output layers of the pre-trained language model [Jurafsky and Martin, 2024, *p.* 234ff.]. Similarly, LLMs have been studied for text classification in a variety of ways, with a focus on prompt-tuning [Han et al., 2022], in combination with knowledge bases [Hu et al., 2022] or within a survey looking at the advances through LLMs within classification scenarios in terms of cost, efficiency, performance and other indicators [Fields et al., 2024].

An annotation on a much smaller and specialized scale will be treated in the next section, referring to segments. Text segmentation approaches divide documents into smaller units, depending on some form of relation between them.

3.4 Text Segmentation

Definition

The NLP task text segmentation refers to the processing of text in so far, that cohesive units in some manner can be detected and differentiated from one another. The manner of "cohesive units" can differ. Semantic cohesion seems the most obvious and natural manner of segmenting text, with size being the other natural candidate. Text segmentation is sometimes also called document segmentation, referring to the same concept for this thesis. By contrast, discourse segmentation can also be observed as a contrasting research task, i.e., the identification of Elementary Discourse Units (EDUs), which comprise the minimal units in discourse analysis based on the rhetorical structure theory [Lukasik et al., 2020, *p.* 4707; Mann and Thompson, 1988].

Segmentation Based on Size

In terms of size as bounding parameter between segments, similarities between tokenization and segmentation become apparent. Tokenization, while usually conducted on a word- or n-gram-level, can also be applied on a sentence- or paragraph-level [Manning et al., 2008, *p.* 22]. Thus, the scope of segmentation and tokenization can be seen as similar, when focusing purely on size-constraints. However, the natural interpretation for tokenization refers to a smaller scale when compared to text segmentation, which typically deals with paragraphs or at least multiple sentences [Dadachev et al., 2014; Ponte and Croft, 1997; Sehikh et al., 2017]. Due to the smaller reach of this research field, SLRs are not common, but one available review from Pak and Teh [2017] instead finds a majority of the retrieved papers studying word-level segmentation [Pak and Teh, 2017, *p.* 174ff.]. The methodology of their paper explains this fact, however, since many of the word-level segmentation publications concern handwritten documents or data in non-English languages with semantically rich characters such as Chinese, necessitating more advanced word segmentation. Segmentation tasks for Germanic languages, as used in this thesis, are more focused on a topic-level, as the authors in Pak and Teh [2017] classified.

Semantic Segmentation

Topic-level segmentation deals with lexical cohesion. The semantic relationships between multiple sentences or even paragraphs and representing a single topic are the major criteria with which segments are separated from each other. The granularity or size of segments processed in a semantic or lexical manner ranges from only two to three sentences [Ponte and Croft, 1997] up to roughly a dozen sentences [Dadachev et al., 2014; Hearst, 1997]. Applications for this process include document clustering, the detection of topic boundaries, e.g., presented in the TextTiling

approach from Hearst [1997], or the processing of text within formats such as PDF [Pak and Teh, 2017, *p.* 169ff.]. The processing of text into segments can be approached using many different techniques, using statistical methods [Beeferman et al., 1999], SVMs [Xia et al., 2010], topic-modeling approaches [Riedl and Biemann, 2012] and in recent years applying neural networks [Gong et al., 2022] and LSTM-based systems [Ghinassi et al., 2023].

Chunking

Through the rise of LLMs, text segmentation has received renewed interest. Limited context windows, i.e., the restricted size of prompts concerning the input as well as output of LLMs, necessitate the splitting of longer documents that are intended to be processed in total by an LLM. Other strategies such as truncating also exist but suffer from problems since only abbreviated documents can be processed in this manner [Koh et al., 2022, *p.* 27]. Instead, text segmentation is implemented as a chunking mechanism – the segmentation procedure referred to in this context as *chunking* – and enables the sequential processing of large text documents. This approach is useful when documents exceed the maximum sequence length or context window size. Different chunking strategies have been presented, see Dong et al. [2023, *p.* 3], including chunking by using fixed size, with [Chalkidis et al., 2022] or without [Dai et al., 2019] overlapping segments. Other systems use flexible sizes that are adapted through reinforcement learning methods [Gong et al., 2020], the latter being one representative of approaches closely related to semantic segmentation strategies from above.

Segmentation by LLMs

Instead of using chunking as part of the preprocessing before actually using LLMs for other use cases, the task of text segmentation itself has been applied with modern techniques recently in a number of studies. BERT-based architectures have been applied in Lukasik et al. [2020] in order to extract both document segments and discourse segments, see above at the beginning of this Section 3.4 for a distinction between the two segment types. Another recent publication from Retkowski and Waibel [2024] on the structuring of video transcripts has applied segmentation using a more efficient smaller-scale transformer-based method. Even more recently, not only BERT-based models, but also BART and GPT-3.5 have been used for segmentation purposes in Alkhalil et al. [2025].

Evaluation

Text segmentation approaches are often applied as an intermediate step enabling more advanced tasks, such as clustering, text recognition in handwritten datasets, and more complex characters in non-western alphabets such as Chinese, see an overview on many tasks in Pak and Teh [2017, *p.* 169ff.]. Therefore, evaluation for this application scenario is

not straightforward, due to widely differing approaches within this NLP task. In addition, there is a lack of uniform datasets with which models are evaluated and compared against one another. Metrics such as precision and recall can be applied regarding correct selection of boundaries separating segments from each other, see Hearst [1997, *p.* 56f.] for details on the evaluation procedure, and Ponte and Croft [1997, *p.* 7f.] for a similar evaluation using precision, recall and f1-score on exact and partial matches. More recently, evaluations in a related manner have been applied in Lukasik et al. [2020, *p.* 4711]. Measures applying a moving window across the document and comparing reference and produced segments have also been used, which assign a penalty if the reference and produced segments do not match [Dadachev et al., 2014, *p.* 78]. Further details on an actual realization of a text segmentation problem will be discussed in Chapter 9, see also Jegan and Henrich [2025b].

The previous tasks all share one common feature, the assignment of a property to some text or parts of a text, be it a type of entity, relation, a class of text itself or segment boundaries. The next two tasks differ in their scope and type of output, which will be detailed in the following sections.

3.5 Text Simplification

Simplification Goals

The NLP task text simplification produces a new text based on some input text. The aim, besides simplifying, is also to usually keep the content from the input text, at least in terms of length, which is one of the main differences when compared to the next application scenario, text summarization [Shardlow, 2014a, *p.* 58f.]. The target of the simplification process differs, depending on the dataset, context and user group, and can specify the degree of simplification, i.e., the complexity of the produced text.

Simplification Strategies

Various simplification strategies exist, which differ in their approach towards generating the simplified text. In general, lexical and syntactic simplification approaches are distinct from each other, at least prior to neural approaches. These neural network-based systems model text simplification as a monolingual machine translation task, i.e., translating the original input text into a simplified translated output text [Al-Thanyyan and Azmi, 2021, *p.* 8ff.]. Since the simplification approaches generate a new text that is intended to be a self-contained version of the original text, further simplification procedures can be applied, if a certain degree of simplification has not been met. Thus, hybrid approaches, e.g., of first a lexical and then a syntactic simplification, are possible for this task.

Evaluation

Evaluation for simplification tasks is inherently difficult due to the sheer number of potentially generated texts. Thus, a simple comparison to a gold standard reference within a training and test dataset is automatically flawed. The generated simplified texts can match the reference simplification or be completely different. Since the simplification approaches differ in their goal as well as setup, different evaluation strategies are necessary. The length of sentences as well as lexical complexity, referring to the use of complex words,[43] both play a crucial role in simplification, but are separate problems. *Readability* as well as *understandability* are further distinctions, the former combining the grammatical complexity as well as length of sentences. The latter refers to the amount of information a user can capture from a text [Shardlow, 2014a, *p.* 59].

Qualitative Evaluation

Manual qualitative evaluation usually requires native speakers or domain experts, which grade the simplification according to several categories, in many cases across three aspects, i.e., simplicity, fluency and meaning preservation [Al-Thanyyan and Azmi, 2021, *p.* 3]. The first category refers to the degree of simplification of the generated text when compared to the original, while the second category is also known as grammaticality and represents the number of grammatical errors or correct grammar in general. The final category contrasts the content of the original text when compared to the produced text.

Quantitative Evaluation

Quantitative evaluation regarding simplification is – similarly to the task of defining complexity – multi-faceted. The number of syllables and sentences of a text have been used to compute the complexity of a text through widely used metrics such as Flesch-Reading-Ease (FRE) [Flesch, 1948] and Flesch-Kincaid-Grade-Level (FKGL) [Kincaid et al., 1975]. As will be detailed below later in this section, modern simplification approaches model the task as a translation task, which is why metrics from the machine translation domain have been applied. Bilingual Evaluation Understudy (BLEU) [Papineni et al., 2002] is one such metric, measuring overlapping n-grams between original and the translation. However, metrics intended for machine translation are not entirely capable of scoring an approach intended for simplification purposes. Thus, combinations of BLEU as well as FRE, here called FKBLEU with aspects from both FKGL and BLEU, have been used [Xu et al., 2016, *p.* 403]. A

[43] The definition of a "complex word" is itself rather complex and can be approached using word frequency, e.g., by measures such as the Kučera-Francis frequency, or other strategies [Shardlow, 2013; Shardlow, 2014b, *p.* 1584ff.]. However, problems persist regardless of the strategy such as the observation of shorter words not always being less complex [Maddela and Xu, 2018, *p.* 3749ff.].

further metric called System output Against References and against the Input sentence (SARI), focusing on comparing the produced simplification and the reference text, has also been applied [Xu et al., 2016, *p.* 404f.]. The latter metric SARI and all other translation-centered metrics suffer from the reliance on comparing the simplification with the reference text, thus eschewing all other potentially well-fitting simplifications, as mentioned above.

Syntactic Simplification

Early automatic simplification approaches focused on syntactic simplifications, by splitting longer sentences into simpler, more comprehensible shorter sentences. The identification of sentence candidates for splitting was initially done using a Finite State Grammar (FSG) or dependency-based models [Chandrasekar et al., 1996, *p.* 1042ff.]. The FSG approach selects chunks from sentences and enables sentence splitting through hand-crafted rules, with some drawbacks regarding long-range dependencies within a text. The problem was tackled with dependency-focused models, which used supertagging to extract more information than commonly found in PoS tags, in order to create dependency links to clearly structure related chunks of a sentence. These links were then used to logically order the chunks in the simplification. Many other syntactic simplification approaches exist, e.g., by extending sentence splitting strategies with the simplification of other syntactic entities such as relative clauses and passive constructions [Siddharthan, 2011]. Another line of work uses event extraction techniques (see above in Section 3.2.2) as a first step in the pipeline followed by simplification rules based on the extracted events [Glavaš and Štajner, 2013].

Lexical Simplification

Lexical simplification is a separate problem entirely, focusing on smaller units of text, mostly regarding the substitution of words with their less complex synonyms. In early systems, synonyms were identified by querying WordNet [Miller, 1995], and in this simple variant, the more frequent words and thus supposedly more familiar or simple words from the synonym list were chosen [Devlin and Tait, 1998, *p.* 164ff.]. Synonym ranking is an important aspect for such systems, when looking at pure lexical substitution as the simplification procedure and has been studied as part of a shared task, e.g., SemEval 2012 [Specia et al., 2012]. Extensions to the pure word-level substitution have been achieved through phrase-level substitution, either by other phrases or single words, or by improved processing of contextual information through vector similarity in order to improve word sense disambiguation, among others [Shardlow, 2014a, *p.* 61f.; Paetzold and Specia, 2016].

Simplification as Translation Task

Both syntactic and lexical simplification measures adhere close to the original text, which is in strong contrast to the remaining techniques. Due to the increasing capabilities of the machine translation research field, simplification was applied as a monolingual translation task. First experiments were done using a parallel dataset with original and simplified texts, the latter in two degrees of simplifications using a phrase-based statistical machine translation system [Specia, 2010]. The statistical machine translation tool Moses was widely used in many text simplification approaches [Koehn et al., 2007].

Hybrid Approaches

Noting the limitations of previous approaches, which include limitations on splitting and deletion operations, several hybrid techniques were created [Al-Thanyyan and Azmi, 2021, *p.* 20ff.]. One such system first applies a semantic model, which splits sentences as well as deletes phrases or words. Subsequently, a translation model is applied for both reordering and substitution, before finally using a language model trained on Simple Wikipedia[44] for fluency [Narayan and Gardent, 2014]. Combined lexical and syntactical simplifications can further be approached as a tree transduction problem [Paetzold and Specia, 2013].

Modern Systems

Since the rise of neural networks for many NLP applications was impacting the machine translation community as well, neural approaches were naturally also applied to simplification tasks. Word2vec-based embeddings as well LSTM-based and CNN-based systems were applied in various architectures and often as hybrid setups [Al-Thanyyan and Azmi, 2021, *p.* 24ff.; Aroyehun et al., 2018]. These models were applied on still relevant datasets for simplification such as WikiSmall [Zhu et al., 2010], WikiLarge [Zhang and Lapata, 2017] and Newsela [Xu et al., 2015]. While the research field of text simplification is much smaller than all other observed NLP tasks except maybe text segmentation, a few approaches using the transformer architecture have nevertheless been proposed. For one, a parametrization of simplification aspects such as length, paraphrasing, lexical as well as syntactic complexity has been proposed using a sequence-to-sequence model, in which these aspects can be controlled using a value from a fixed ratio [Martin et al., 2020]. For another, the discrete nature of simplification aspects was conceptualized as three rewards, fluency, salience and simplicity, for which separate optimization models were created that are then combined as one Self-Critical Sequence Training (SCST) system [Laban et al., 2021]. The latter model

[44] See `https://simple.wikipedia.org/wiki/Main_Page/`.

was applied for the German text simplification system from Fruth [2022], see Section 7.1.1.

LLMs in Text Simplification

The arrival of LLMs has seen some discussion for simplification approaches as well, however drawbacks were noted in the few approaches available. A LLaMA-based [Touvron et al., 2023] approach matched or surpassed prior simplification approaches based on BERT, while also noting restrictions due to prompt stability issues as well as limitations due to the number of computational resources needed [Baez and Saggion, 2023]. Further studies were done with a focus on coherent behavior of simplification models, mostly BERT-based models, but also on GPT-3.5 [Anschütz et al., 2024]. The authors noted limitations of all models when conducting artificial adversarial attacks on simplification data and discussed the possibility of a lack of simplified text within the training data for the large pretrained models as cause of these problems [Anschütz et al., 2024, *p.* 191ff.].

Simplification as an NLP task has been referenced by the more researched field of text summarization in some ways, e.g., as a part of the summarization pipeline through simplification steps such as lexical substitutions in the final steps of generating the summary [Al-Thanyyan and Azmi, 2021, *p.* 2f.]. However, a closer look at text summarization is needed due to the somewhat contrasting goals.

3.6 Text Summarization

Short Definition

The NLP task text summarization is, in terms of task description as well as output, at first glance similar to text simplification. An original text is analyzed, changed in some way and a new text with properties from the original text is given as the output. The selection of candidates that should be kept for the summary, in many cases sentences, is, however, also achieved through means that are also in use in other application scenarios mentioned above such as information extraction. Thus, the aim of summarization approaches is obviously different when compared to simplification, mainly in terms of producing a compressed variant of the original text.

Single- vs. Multi-Document

As with text classification, different dimensions have to be mentioned briefly. Single-document vs. multi-document summarization is one such dimension. This thesis will focus more on single-document summarization, thus a quick task description for the latter will suffice. Multi-document summarization, as the name suggests, takes several different documents, and produces one overarching document in-

tended to summarize all input documents. The difficulty increases in the multi-document setting due to problems such as the compression factor or redundant information included in separate documents [Goldstein et al., 2000]. These problems can be approached using traditional techniques such as random forest models [John and Wilscy, 2013]. The compression ratio can vary widely, which is natural due to different settings when summarizing only a few documents, or many documents, sometimes in the hundreds of texts. Redundancy can be a major factor when designing the summarization system, depending on the context as well as data, especially if the texts cover related topics. Single-document summarization will remain the focus here, but the research field of multi-document summarization is still immensely relevant due to ever-increasing amounts of data that require analysis.

Extractive vs. Abstractive

Traditional and more modern techniques are, as with the previous NLP tasks, another contrasting dimension, but text classification specifically encompasses an even simpler distinction. This distinction is between extractive and abstractive[45] summarization techniques, which will both be studied as the two main strategies of summarizing texts for this thesis. In short, extractive summarization techniques identify portions of a text that are relevant for the summary and extract those text passages without changing them. This process results in a summary that is completely composed of text that already appeared in the original text. Abstractive summarization, however, processes the text and, using a variety of different methods, composes some new text with the original text serving as the basis of the generated output. For abstractive summaries, new text is produced that does not necessarily have to appear exactly or even partly in the original text.

Early Systems

Extractive text summarization will produce a text that is essentially a selection of the most important parts of the original text, thus a subset of the input text. Approaches differ in their strategy on how to select those important passages. Initial approaches used statistical properties, i.e., the distribution of words within the text, to identify keywords and therefore also the selection of sentences for the summary. TF-IDF [Jones, 1972; Luhn, 1957] is one such rather simple measure in order to identify the importance of words relative to other words from the same text, which in turn allows the ranking of words and sentences. The identification of important words is usually one of the steps within a processing pipeline for summarization methods that comprise various statistical, topic-based

[45] Also known as generative summarization.

or graph-based approaches as well as using older machine learning techniques [Gambhir and Gupta, 2017, *p.* 4ff.].

Lead Bias

The news domain has been traditionally one of the most researched domains within text summarization and many of the more popular datasets have been created from within this domain such as CNN/DailyMail [Dernoncourt et al., 2018]. One consequence of this focus on the news domain has been the development of methods that are adapted for such texts, e.g., lead-n, which selects the first n sentences of a text due to the prevalent lead bias of news texts [Brandow et al., 1995; Jones, 2007, *p.* 9]. The lead bias has been noted and studied within the news domain, especially its impact on the performance even on modern models when randomly shuffling the sentences around [Kedzie et al., 2018, *p.* 1826f.]. Negative effects have also been noted on other domains, e.g., the legal domain in the form of European legal acts [Aumiller et al., 2022], and need to be considered there as well.

Machine Learning Summarization

Different architectural models have been proposed to vary the extraction process, such as TextRank. This algorithm is a graph-based ranking system, which extracts sentences for the summary based on a graph that is created for the entire text and a ranking of sentences is calculated by comparing common words between sentences [Mihalcea and Tarau, 2004]. Similarly, LexRank also creates a graph and compares sentences with each other, but with the help of cosine similarity instead [Erkan and Radev, 2004]. The rankings produced by both TextRank and LexRank are based on the information retrieval ranking algorithm PageRank, already noted above in Section 3.1 [Brin and Page, 1998; Page et al., 1999]. Due to the flexible ranking algorithms based on PageRank, TextRank and LexRank can be adapted to sentence similarity or other similarity tasks, such as information extraction or keyword extraction tasks [Wei and Ding, 2023; Yan et al., 2024]. More complex systems use machine learning approaches. They are modeling the summarization task either as a supervised learning classification task or implement deep learning approaches, which apply embedding-based approaches in order to compute a similarity ranking [El-Kassas et al., 2021, *p.* 6ff.]. The latter techniques, i.e., language models and computationally intensive systems, were, however, used in more expansive ways in the other category of text summarization approaches, described in the following.

Neural Networks for Summarization

Abstractive approaches, in contrast to extractive text summarization systems, generate a text that can differ from the input text. The generation of new text was achieved through the rise of neural networks (see Section 2.3) and systems such as RNN- or LSTM-based models that were also

applied on text summarization tasks [El-Kassas et al., 2021, *p.* 9ff.]. One widely referenced early method transformed the input text into a contextual vector representation using an attention-based encoder and generated the output based on a feed-forward neural network language model [Rush et al., 2015].

Hybrid Summarization Systems

Many different architectures and even hybrid approaches exist for abstractive text summarization, also incorporating extractive techniques within the processing pipeline [Wang et al., 2017]. In contrast, the use of neural networks is not restricted to the generative aspect of the abstractive summarization approaches but can also be applied to sentence selection or semantic text matching aspects in extractive summarization systems as well. Such hybrid systems can, e.g., be realized using a BERT-based architecture in order to extract sentences and then to match the candidate summary according to the similarity to the source document, but crucially not rewriting or generating text [Zhong et al., 2020].

Reference Summaries

One problem persists, regardless of extractive, abstractive or hybrid approaches, namely the evaluation of the generated summaries. This problem is analogous to text simplification (see above in Section 3.5), since in most datasets only one reference summary exists for each source document. Thus, quantitative evaluation methods, when only comparing the produced summaries to the reference summary, exhibit the same drawbacks as for simplification due to the reliance on one summary as the ground truth. The problem here, once more, is the fact that many possible summaries exist for an original text, and evaluations focusing on just one reference summary are thus restrictive in properly evaluating summarization methods. Extractive summarization methods are somewhat limited in producing summaries that can be seen as such new summaries that differ in a major way from the reference summary due to their purely extractive nature. This attribute of adhering to the original input text might however also be a feature for sensitive domains and use cases, see the discussion on lacking metrics in general in Section 6.2 and especially for summarization tasks in selected use cases in Section 7.2.2.

Quantitative Evaluation

The quantitative evaluation of summarization systems is commonly done using the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) metric. This score is applied either in the R-1 variant for unigram overlap between the original text and the produced summary, or with R-2 for bigram overlap or even R-L for the longest common subsequence [Lin, 2004b]. As mentioned above, the reliance on reference summaries is a limiting factor when purely looking at metrics such as ROUGE, or other metrics that are sometimes used in summarization

tasks like BLEU or Metric for Evaluation of Translation with Explicit Ordering (METEOR) [Banerjee and Lavie, 2005] that were originally created for machine translation tasks. Apart from this limiting factor, some further drawbacks with reference-based metrics on matching substrings are apparent.

ROUGE

Firstly, only exact matches between the reference and the produced summary are scored with two of the most popular ROUGE variants, R-1 and R-2 [Lin, 2004b, *p.* 74ff.] and thus syntactic or lexical variations lead to worse scores. In the original ROUGE paper, other variants of the metrics have been proposed apart from the R-L score with the longest common subsequence. ROUGE-W introduced a weighted variant of R-L emphasizing consecutive matches, and ROUGE-S added a skip-bigram mechanic based on the co-occurrence of bigrams [Lin, 2004b, *p.* 76ff.]. Secondly, ROUGE in all variants requires matching words in both reference and produced summary, thus punishing the use of synonyms or word substitutions instead of the original words. ROUGE therefore necessitates lexical overlap between both versions in order to produce a higher score [Akter et al., 2022]. For the last problem, methods using dependency parse trees have been proposed for comparing the summaries [Hovy et al., 2006]. Despite all noted restrictions, ROUGE is still the dominant quantitative metric for summarization to this day.

Qualitative Evaluation

The reliance on a single reference summary has been challenged as well, even by the author of the original ROUGE metric, through an analysis of an increased number of reference summaries, which would potentially lead to a higher correlation with human judgments [Lin, 2004a, *p.* 8f.]. In the same paper, ROUGE is successfully applied for evaluating generated summaries against multiple reference summaries [Lin, 2004a, *p.* 3ff.]. However, despite some datasets published in the Document Understanding Conference (DUC)[46] including multiple references, many large-scale datasets, as the above-mentioned CNN/DailyMail corpus, that are heavily used today, only include one reference summary and therefore prohibit informative evaluations. Thus, as has been noted widely, a qualitative evaluation is a logical next step when analyzing summarization systems, in addition to quantitative evaluation against a reference summary [El-Kassas et al., 2021, *p.* 19f.; Jones, 2007, *p.* 1453ff.]. The criteria for a manual evaluation done by experts are extensive, ranging from coverage,

[46] Available from 2001-2007, and now as part of the Text Analysis Conference (TAC), see `https://www-nlpir.nist.gov/projects/duc/index.html`.

cohesion, grammaticality to readability and many more.[47] The evaluation of summarization applications has been presented here in more detail than for the other NLP applications, but has been done so to offer a brief glimpse of the complexity of evaluating text processing systems. A more detailed view, regarding many different NLP tasks will be displayed below in Sections 6.2 and 7.2.

LLMs for Summarization

One of OpenAI's innovative advancements towards using transformer-based language models within a prompt-based interface has been Reinforcement Learning from Human Feedback (RLHF) on a summarization task [Stiennon et al., 2020]. While not directly incorporating a chat interface, the incremental development of this approach led directly to the creation of ChatGPT and prompt-based LLMs, as known and in use today.[48] In general, the summarization of texts is one of the most common and researched use cases for LLMs, as well as one of the most successful ones, reaching levels of human-written summaries in many cases [Zhang et al., 2024b]. While the context size was initially a limiting factor, newer models capable of handling larger context windows have allowed the summarization of larger single documents in total or at least for many use cases of small to medium length texts. These models reach token counts within the context windows in the thousands with models published in 2023 and further increases since then [Patil and Gudivada, 2024, *p.* 26].

Conditional Generation for Summarization

Briefly, the summarization task within LLMs, and for other abstractive or generative NLP tasks in general including simplification as well, can be characterized as a conditional generation task, see above in Section 2.4. More concretely for LLMs, the prompt includes the original text and a token signaling the action towards summarizing the preceding text, sometimes referred to as *tl;dr*[49] token. After this token, a word prediction task is started and the response produced by the LLM can be seen as the summary for the original text from the prompt [Jurafsky and Martin, 2024, *p.* 204ff.].

Problems with LLM-Generated Summaries

The more free-form LLM-generated summaries have shown that standard automatic metrics such as ROUGE are even more reliant on high-quality reference summaries, when comparing different reference sum-

[47] A larger discussion has been done in many surveys as well as essays on these topics, e.g., in Jones [1999].

[48] The introductory blogpost of ChatGPT from late 2022 references RLHF, see `https://openai.com/index/chatgpt/`.

[49] Referencing the *too long; didn't read* abbreviation, initially used colloquially in social media platforms and nowadays also in more mainstream use, see `https://www.merriam-webster.com/dictionary/tldr/`.

maries on the widely used XSUM dataset [Narayan et al., 2018; Zhang et al., 2024b, *p*. 48f.]. Other problems within LLMs such as hallucinations or biases within the training data are still being worked on [Ji et al., 2023, *p*. 15ff.; Jurafsky and Martin, 2024, *p*. 214f. and *p*. 219f.] and will be discussed in detail below in Chapter 6.

After the five main application scenarios from this thesis have been described in terms of background and an overview on models – both traditional and modern – and typical use cases as well as evaluation procedures has been given, it is necessary to take a closer look on the related work regarding traditional techniques. These techniques and factors in which they differ from modern approaches deserve further discussion, which will be given next in Chapter 4.

4 Related Work

Now that the NLP landscape in general as well as the foundations of the relevant NLP tasks for this thesis have been laid out in Chapter 2 and Chapter 3 respectively, a closer look at the related work will be presented in this chapter. While five application scenarios will be briefly detailed with a focus on their involvement or treatment of traditional models in Section 4.1, an analysis of further dimensions relating to the use and application of such models on language tasks will be given below in Section 4.2.

Definition of Related Work

Usually, the related work on NLP applications is focused on state-of-the-art techniques and depending on the background also on historical developments for the respective application. However, the focus of this work does not lie on one particular task that will be analyzed in detail, but rather on the use of different methods and traditional techniques for different application scenarios in particular. Therefore, a different approach was chosen for this thesis. Traditional NLP techniques and their use for tasks introduced in Chapter 3 will be discussed in Section 4.1. Furthermore, additional dimensions regarding the use of traditional techniques, especially in contrast to modern methods, will be presented in Section 4.2.

4.1 Related Work on Selected NLP Tasks

Traditional techniques as defined in Section 2.5 are in many cases not at the forefront of the state of the art, purely due to their age. However, when reviewing the state of the art for the application scenarios detailed in Chapter 3, several categories of ongoing use of traditional techniques have been detected [Jegan and Henrich, 2025a].

Categorization for Traditional Models

One such category is the use of traditional techniques as a benchmark, which are compared to the main model(s) of the paper in question. As just one example, relation extraction was implemented using a GNN as the main method and compared to several techniques that are considered traditional in this thesis, such as SVM, Naive Bayes and random forest models [Mazumder et al., 2023]. A second category is the use of tradi-

tional techniques as part of the pipeline, either in preprocessing stages or during the main computation of the system, e.g., using an SVM as preprocessing for a text classification system, which feeds into a BERT-based classifier [Wang et al., 2023b]. Finally, the third category comprises the application of traditional techniques as the main method of a system, which are still appearing in recent publications, e.g., combining extractive summarization approaches with BERT-based techniques to improve the ranking of sentences for extractive summarization approaches [Licari et al., 2023]. These three categories will be visualized below in Tables 4.2 to 4.5, including a classification regarding competitive performance or performance that is trailing behind in Tables 4.2 and 4.3. But first, to properly contextualize the techniques listed in Tables 4.2 to 4.5, the results of the SLR itself need to be presented.

Structured Literature Review

A short selection of literature will be presented for each application scenario that was retrieved as part of the SLR, which has been done for papers published in 2023 in the Association for Computing Machinery (ACM) Digital Library and that will show the continued relevance of traditional techniques in NLP [Jegan and Henrich, 2025a]. For details on the procedure of the SLR, see Jegan and Henrich [2025a, *p.* 4ff.]. In short, searches were conducted using string matching for publication titles, for papers published in 2023 in the ACM Digital Library. Also, some of the referenced publications use 2022, 2023 or 2024 as publication date in this section, despite the search parameters set to 2023, or for an updated query to 2024 later in this section. This discrepancy is based on different metadata indexed in the ACM Digital Library, in contrast to metadata retrieved from Google Scholar, which was the main literature search engine for this thesis. These small differences do not impact the learnings from this section. Overall, research question ❶ – referring to the continued research and use of traditional models – & ❷ – relating to the competitiveness of traditional techniques in contrast to modern techniques – will be studied through the results of this SLR, which are shown in Figure 4.1.

Information Extraction

Results for Information Extraction

Three publications were retrieved for this application scenario. One approach used SVMs and keyword extraction techniques as comparison methods for the main model, which was implemented as a multi-angle feature learning system in order to extract information from a web page [Liu, 2023]. The other two publications used traditional techniques as core methods of the respective paper. On the one hand, information is extracted from one of the more dominant domains in NLP research, the news

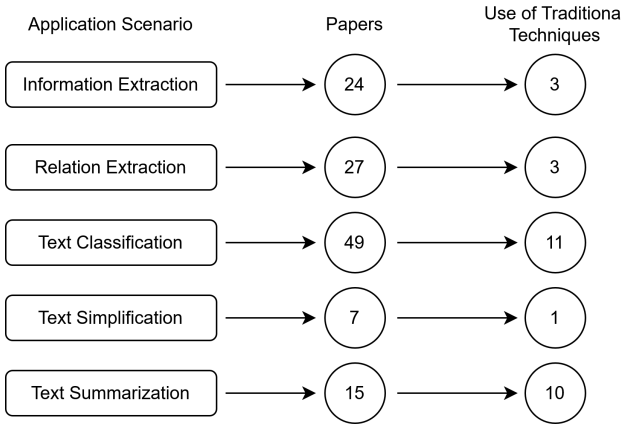


Figure 4.1: Results of the SLR regarding the use of traditional techniques in current NLP tasks. Based on Jegan and Henrich [2025a, *p.* 5].

domain, by applying SVMs, random forests and decision trees as main classifiers [Sultana et al., 2022]. On the other hand, rule-based extraction models are applied on company announcements in analyzing stock developments, which are also compared with machine learning and deep learning techniques [Xiang and Yangfei, 2023].

Relation Extraction

Results for Relation Extraction

Once again, three publications were retrieved for this application scenario. On a low-resource relation extraction task, a traditional technique using semi-supervised self-training is compared to the main method of this system, which is inspired by self-training methods, and others such as BERT-based approaches [Hu et al., 2023a]. Traditional models such as Naive Bayes and SVMs were compared to another newly introduced model in another paper, in this case based on a GNN, and while trailing behind in most benchmarks, the SVM model achieved the highest score regarding precision for the LectureBank[50] dataset, surpassing the GNN [Mazumder et al., 2023]. Finally, another core method was implemented using traditional methods by using SVM- and K-Nearest Neighbors (KNN)-based approaches within a hybrid machine learning model intended to extract complex Arabic relations [Osman et al., 2023].

[50] See `https://github.com/Yale-LILY/LectureBank/`.

Text Classification

Results for Text Classification

Eleven publications were retrieved for this application scenario. A selection of these eleven papers used traditional techniques as part of the processing pipeline. In one case, TF-IDF was applied within the pipeline [Shi et al., 2023], while both TF-IDF and point mutual information are used in another [Yuan et al., 2022]. SVMs are applied in the pipeline of a weakly supervised open text classification approach [Wang et al., 2023b]. Another paper, itself a survey on instance selection methods for text classification, applied TF-IDF as part of the pipeline, while also using SVMs as a comparison method, achieving the best results in two benchmarks, while trailing behind in most other metrics [Cunha et al., 2023].

Models as Comparison

Using traditional techniques only in a comparing context was done in several publications. Decision trees, BoW and logistic regression classifiers were applied in a study on interpretable text classification, only trailing slightly behind the state of the art in several benchmarks [Hong et al., 2023]. Similarly, traditional models such as SVMs, random forests and others were used as comparison methods towards text classification based on topic modeling, and once again showing competitive results on multiple benchmarks [Saeed et al., 2023]. However, two publications included traditional models purely for comparison's sake, displaying demonstrably weaker results for SVM and Naive Bayes classifiers when compared to the proposed models [Lu et al., 2022; Mehmood et al., 2023].

Models as Main Model(s)

The use of traditional models can be categorized as the main models or one of the main models of each of the remaining three publications. First, a linear n-gram classifier was applied as one of three main models in a study on automatic noise generation and reduction [Chen et al., 2023a]. The other two papers both apply decision trees as representatives of traditional approaches, once as the sole method [Lin, 2023] on an education task, and once in contrast to a modern BERT-based model [Zhang et al., 2023a]. In the first paper, a deeper understanding between the applied models, i.e., decision trees, and the ethical use of text classification systems in general as well as the social outcomes through programming abstractions were analyzed [Lin, 2023]. The second paper presents decision trees and a BERT-based approach in order to examine uncertainty within text classification systems [Zhang et al., 2023a].

Text Simplification

Results for Text Simplification

Only one publication was retrieved according to the search methodology presented in Jegan and Henrich [2025a, *p.* 4ff.], and only by extending the queries to include abstracts as a search parameter. The sole retrieved

paper is a survey on lexical complexity prediction and includes techniques such as SVMs, decision trees and random forest algorithms besides more modern models [North et al., 2023].

Text Summarization

Results for Text Summarization

Ten publications were retrieved for this application scenario. Placing the use of traditional techniques into the three categories presented above is more complicated for text summarization due to the more expansive definition of extractive summarization techniques as representatives of traditional techniques, see above in Section 2.5. Thus, a clear distinction between the three categories is not always possible and more overlaps are therefore present.

Models as Part of the Pipeline

Five of the retrieved papers use traditional techniques as part of the pipeline, and of those five, only one uses a purely abstractive summarization system – implemented using the PEGASUS model [Zhang et al., 2020a] – with traditional techniques only being part of the processing pipeline [Park and Park, 2023]. The other four papers all used extractive techniques as comparison baselines. The first publication presented a hybrid system with a combination of both abstractive and extractive techniques intended for rewriting purposes and a lead-3 summarization system as baseline, with this traditional technique trailing behind [Bao and Zhang, 2023]. Also trailing behind were the extractive systems in a system focused on Vietnamese text, in which text segmentation was used to improve the summarization performance [Lam et al., 2022]. The remaining two publications used traditional techniques as part of the pipeline, while also including extractive approaches as the main model. The first of which extended the PageRank algorithm with various similarity measures on a low-resource Asian language, namely on Konkani texts, and also applied lead-3 as comparison baseline [D'Silva and Sharma, 2023], too. The second paper dealt with Italian legal texts and also used traditional models for comparison, here LexRank, while also constructing a hybrid system consisting of an extractive summarization approach on a list of sentences pre-ranked by a BERT-based model [Licari et al., 2023]. The latter two papers used extractive techniques, and thus traditional ones, as part of their main model.

Models as Comparison and Main Model(s)

From the remaining five publications, two were using traditional techniques purely for comparison purposes and indeed showed results that trailed behind the main models of the papers. The first of the two dealt with extreme abstractive text summarization of scientific publications [Atri et al., 2023], while the second presented an abstractive sum-

marization model enhanced by reinforcement learning [Du et al., 2023]. Two further papers used extractive approaches as their main model, first presenting a TextRank-based extractive approach, which surpassed modern transformer-based approaches on Chinese text summarization [Luo et al., 2022]. Second, Arabic text summarization was analyzed by using a hybrid approach using statistical approaches in selecting candidates for the summary, which were chosen using a semantic comparison based on a word2vec model and that in the end surpassed other extractive approaches [Omar and Al-Shaar, 2023]. Finally, the last retrieved publication includes a literature review on social media text summarization approaches, which provides an overview on available studies on summarization approaches for this domain including both abstractive and extractive techniques to a similar extent [Papagiannopoulou and Angeli, 2022].

Overview of Techniques

After analyzing the retrieved papers, the categories regarding the use of traditional models that were introduced at the start of this section will be presented next. First, a non-exhaustive number of techniques and models from these publications is displayed in Table 4.1 as an overview. The table is not intended to be a complete collection of all approaches currently in use for the respective use case. Rather, a snapshot of the current NLP landscape is presented, informed by research on traditional models and by surveying competing methods, primarily through experiments (see Chapters 7 and 9) and the related work (see the current chapter).

Limitations of this Overview

Some compromises were necessary to succinctly visualize the overview from this Table 4.1 in this manner. While Tables 4.2 to 4.5 offer a classification regarding the role of each publication and its use case for the respective techniques, other information on the implementation, programming language and parameter selections were omitted. However, further references are included in other parts of this thesis, e.g., in the brief presentation of the landscape of NLP in Chapter 2, and as an overview in the SLR, presented in this section. Still, Table 4.1 displays an overview of both traditional and current models with their applicability on different use cases and can serve as a brief outline for techniques applied in this thesis. Noteworthy in Table 4.1 are the applicability of nearly every technique for the popular tasks such as classification and summarization. In contrast, niche tasks such as segmentation have been approached using much fewer techniques. Also, the generative nature of simplification in most cases prohibits many techniques from easily being used for this task, with the exceptions of rule-based or decision tree substitution tasks for simple lexical or syntactic simplifications. In general, all listed techniques could still be applied to any of the tasks. Additional effort would be

required if no preexisting publication or implementation is already available for the respective task-technique combination.

Applicability for Use Case					
Method	Inf. & Rel. Extr.	Class.	Segm.	Simpl.	Summ.
Traditional Approaches					
DT Learning		[49; 124]		[316]	[68]
DT	[412]	[167; 471]	[34]	[317]	[147]
LexRank	[458]				[106]
Log. Reg.	[278; 412]	[304]			
Naive Bayes	[278]	[374; 393]	[131]		[68]
Random Forest	[278]	[372; 393]			[198]
Rule-Based	[346; 450]	[242]	[95]	[210]	[147]
SVMs	[253; 486]	[80; 437]	[449]	[386]	[147]
TextRank	[441]				[282]
Modern Approaches					
BERT	[94]	[437; 471]	[261]	[125]	[29; 152]
CNN	[115]	[393]		[14]	[226]
LSTM	[115; 253]	[280; 393]	[130]	[307]	[97; 226]
RNN		[254]	[383]		[226]
LLMs	[434; 452]	[116]	[188; 236]	[13; 24]	[84; 477]

Table 4.1: Applicability of a selection of NLP methods and techniques for the use cases in this thesis. "Inf. & Rel. Extr." refers to information and relation extraction (see Section 3.2), "Class." refers to text classification (see Section 3.3), "Segm." refers to text segmentation (see Section 3.4), "Simpl." refers to text simplification (see Section 3.5) and "Summ." refers to text summarization (see Section 3.6). "DT" refers to Decision Tree and "Log. Reg." to Logistic Regression methods. Existing references denote the applicability of the respective technique (rows) for a use case (columns).

Role of Methods

The categorization of techniques into different roles is presented next. Regarding the Tables 4.2 to 4.5, some references appear in multiple tables, since the techniques can be part of the pipeline and also part of the comparison within the evaluation, see, e.g., Gupta et al. [2024], i.e., [147] in the Tables 4.1 to 4.5. Similarly, core techniques can also be compared to more modern methods and achieve performance that is competitive or trailing behind. Therefore, the same reference may appear multiple times in the tables. For example, Mazumder et al. [2023], i.e., [278], report results for information and relation extraction using Naive Bayes, logistic regression and random forest systems as comparison benchmarks. Only the latter

Competitive Performance					
Method	Inf. & Rel. Extr.	Class.	Segm.	Simpl.	Summ.
DT Learning					[68]
DT		[167]		[317]	
LexRank					[106]
Log. Reg.		[304]			
Naive Bayes					[68]
Random Forest	[278]	[372]			[198]
Rule-Based	[346; 450]			[210]	[147]
SVMs	[486]	[80]	[449]	[386]	
TextRank					[282]
BERT			[261]		[29]
CNN		[393]		[14]	
LSTM	[253]	[393]	[130]	[9]	
RNN		[254]	[383]		[226]
LLMs	[434; 452]	[116]	[188]	[13; 24]	[84; 477]

Table 4.2: Overview of NLP methods that achieve competitive performance. See the caption of Table 4.1 for hints with respect to the abbreviations.

system achieves competitive results for precision, see Table 4.2, while the other techniques trail behind, see Table 4.3.

Performance Classification

The classifications noted in Tables 4.2 to 4.5 are not based on a quantified evaluation but a qualitative analysis of the publications by the author of this thesis. Therefore, a certain fuzziness or room for interpretation regarding certain choices is certainly possible. Nevertheless, some observations can be glimpsed from the classifications. When directly comparing both Tables 4.2 and 4.3, more references can be observed in the top half of Table 4.3 showing performance that is trailing behind for publications presenting traditional techniques. In contrast, Table 4.2 displays more references in the lower half, thus displaying higher performance for the more modern techniques. Similarly, LLMs are never categorized as trailing behind and thus only show competitive performance. While a direct comparison is naturally not possible across different use cases and between drastically varying evaluation settings, a rough estimate can still be seen through this classification regarding the dominance of LLMs across a broad range of NLP tasks.

Trailing Behind Performance					
Method	Inf. & Rel. Extr.	Class.	Segm.	Simpl.	Summ.
DT Learning					
DT	[412]		[34]		
LexRank					
Log. Reg.	[278]				
Naive Bayes	[278]	[393]	[131]		
Random Forest		[393]			
Rule-Based			[95]		
SVMs	[253]	[80]			
TextRank	[441]				
BERT					[152]
CNN					
LSTM		[280]			
RNN					
LLMs					

Table 4.3: Overview of NLP methods that exhibit performance that is trailing behind. See the caption of Table 4.1 for hints with respect to the abbreviations.

Core vs. Part of the Pipeline

The other distinction concerns the role of the technique within the actual text processing system, i.e., purely as part of a pipeline in Table 4.5 or the technique being part of the core model or one of the core models in Table 4.4. A clear separation between the two classes cannot be drawn here. However, more publications were identified using the displayed methods as a core technique than just as part of a pipeline. This observation certainly reflects the selected list of publications displayed here, because the publications retrieved through the SLR from Jegan and Henrich [2025a] serves as a basis for this collection. In addition, publications concerning the supervised theses that will be referenced throughout this thesis, see above in Section 3.1 and mainly in Chapter 7, have been analyzed for Tables 4.1 to 4.5. Also, similar to the task-leading performance achieved by LLMs and their occurrence in Table 4.2, LLMs also serve as core techniques for all observed tasks, see Table 4.4, while they only appear as being part of a pipeline, see Table 4.5, in the text segmentation task, which will be discussed in Chapter 9.

	Core Technique				
Method	Inf. & Rel. Extr.	Class.	Segm.	Simpl.	Summ.
DT Learning		[49; 124]		[316]	[68]
DT		[471]			
LexRank	[458]				[106]
Log. Reg.	[412]	[304]			
Naive Bayes		[374]			[68]
Random Forest					[198]
Rule-Based	[346; 450]	[242]		[210]	[147]
SVMs	[486]		[449]		
TextRank					[282]
BERT		[471]	[261]		[29]
CNN	[115]			[14]	
LSTM	[115]		[130]		
RNN		[241]	[383]		
LLMs	[434; 452]	[116]	[188]	[13; 24]	[477]

Table 4.4: Overview of NLP methods that serve as a core technique. See the caption of Table 4.1 for hints with respect to the abbreviations.

Additional NLP Tasks

Additional NLP tasks were analyzed in a more limited manner, which is why they are not discussed in as much detail compared to the tasks presented in Table 4.1 and also why they do not appear in Tables 4.1 to 4.5. These other tasks include keyphrase prediction (see Section 7.2.1 and Lang [2025]), entity matching (see Section 7.4.2 and Hebeis [2025]), question answering (see Section 7.1.2 and Möglich [2025]) as well as text segmentation (see Chapter 9 and Section 3.4).

SLR for 2024

While the SLR was mainly focused on papers published in 2023, an outlook towards 2024 was done to include newer publications [Jegan and Henrich, 2025a]. The retrieved numbers stayed roughly the same regarding submitted papers in the five application scenarios.[51] The continued relevance of traditional papers was, however, consistent with the presented observations from 2023. Information extraction is still being researched with traditional approaches used for comparison purposes [Parfenova, 2024] or as part of a pipeline [Liberatore et al., 2024], while relation extraction applies such techniques as comparison benchmarks [Shi et al., 2024]. SVMs were continually used as part of the pipeline

[51] More complete numbers can be found in the SLR, see Jegan and Henrich [2025a] and the accompanying statistics on `https://doi.org/10.5281/zenodo.13683800`.

	Part of the Pipeline				
Method	Inf. & Rel. Extr.	Class.	Segm.	Simpl.	Summ.
DT Learning					
DT					[147]
LexRank					
Log. Reg.	[412]				
Naive Bayes					
Random Forest					
Rule-Based			[95]		[147]
SVMs		[80; 437]			[127; 147]
TextRank	[441]				
BERT	[94]	[437]	[261]	[125]	
CNN	[115]				[226]
LSTM	[115; 253]				[97; 226]
RNN		[241]			
LLMs			[236]		

Table 4.5: Overview of NLP methods that serve as part of a pipeline. See the caption of Table 4.1 for hints with respect to the abbreviations.

[Pang, 2024], or as comparison benchmarks alongside Naive Bayes and random forest approaches [Sikosana et al., 2024], both papers intended for text classification. Even text simplification was still present in 2024, with a paper adapting rule-based techniques for syntactic simplifications [Keerthan Kumar et al., 2024]. Lastly, extractive summarization approaches are still also in use, combined with comparative learning procedures [Zhou and Xu, 2023] or in a low-resource setting of Hindi documents using multiple linguistic features to enhance the summarization approach [Gupta et al., 2024].

More Perspectives

Research question ❶ can therefore be answered positively, since all application scenarios were still studied using traditional models and techniques, even in this small-scale SLR, conducted on the ACM Digital Library for 2023 – and as an addition also in a more limited manner for 2024 – and therefore ignoring preprints and newer publications. As for research question ❷, eleven publications were retrieved as part of the SLR with competitive performance from traditional models in contrast to other techniques for publications from 2023, see more details in Jegan and Henrich [2025a, *p.* 9], in addition to several other publications, see the top half of Table 4.2. After the continued use of traditional techniques

in recent literature has been briefly laid out, and thus an argument for the ongoing relevance for such techniques has been presented, further perspectives on the choice of techniques for NLP tasks will be shown in the next section.

4.2 High-level Comparison of NLP Methods

Lack of Uniform Goal

Some of the application areas detailed in this thesis can be grouped according to the scope of the output they produce, such as a transformed text based on some original input for text simplification and text summarization. However, they have not one unifying aim or goal across all applications, which is natural due to the varying output they produce. Due to this variance other dimensions should be identified, which all application scenarios share and that can shed some light on differences between traditional approaches when compared to modern approaches.

Quality & Performance

First and foremost, all other dimensions presented in this section will diminish in importance if a certain desired quality or performance is not met by the chosen approach. This desired quality is highly variable, depending on a multitude of factors. The quantification of quality is widely different depending on the NLP task applied for each use case, which in turn has to be evaluated within task-specific settings and appropriate metrics. A brief overview of different evaluation procedures has already been presented above for each of the NLP tasks relevant for this thesis, see Chapter 3, but problems and challenges applying the presented evaluation metrics are also common in today's NLP applications, see below in Section 6.2 and for concrete use cases in Section 7.2. Achieving the highest quality for a certain application is certainly desirable, but a cost benefit analysis within given constraints, e.g., budget, time, hardware resources, personnel among others[52] and especially relevant here, a quality threshold, seems a more realistic outcome.

Energy Consumption

Another such dimension is efficiency or energy consumption. The analysis of energy spent when training or even executing NLP models, or models in machine learning in general, is a research area, that is actively studied, e.g., in Al-Jarrah et al. [2015], Patterson et al. [2021] or Strubell et al. [2020]. No clear procedures or tactics have been cemented as standards or have been in use across different dimensions such as models or tasks. The emerging research field has been increasing in visibility not only due to the rise of neural networks within machine learning, but also

[52] See a comprehensive analysis of factors below in Section 5.2.

due to surging effects of the climate crisis in recent years. Thus, the research field has been approached by attributing the names *Green AI* [Schwartz et al., 2020] or *Green NLP* [Hershcovich et al., 2022].

Transparency of Model Training

Various strategies have been used for measuring efficiency in NLP research. One common and relatively basic value comprises hardware specifications that have been used during training, however even popular and widely cited papers omit such information, e.g., Radford et al. [2018] or Liu [2019]. In other popular publications at least the specific hardware is mentioned, e.g., which graphics cards are used, however not the number of GPUs or the time spent when training the models [Brown et al., 2020]. A more transparent step includes both the hardware that was used and the training time, e.g., in number of graphics cards and hours that were spent training [Devlin, 2018, *p.* 13; Vaswani et al., 2017, *p.* 6002]. Starting with these details, some studies have been done, which compare different language models and their costs for pre-training, fine-tuning and inference in terms of hardware requirements, training time and estimated costs in terms of renting the corresponding GPUs [Zhou et al., 2021, *p.* 142].

Limitations in Measuring Energy

The previous study in Zhou et al. [2021] was, however, itself critically analyzed in a different work, which describes the restrictions when using time and financial costs as the only features [Cao et al., 2020, *p.* 147].[53] Other measurements that are more detailed count the number of Floating-Point Operations Per Second (FLOPS), which are necessary to produce an output, as the major parameter in order to gauge the computational expenses [Schwartz et al., 2020, *p.* 60]. This measure, however, was also criticized as unreliable and uncorrelated for efficiency predictions [Cao et al., 2020, *p.* 147; Henderson et al., 2020, *p.* 2].

Hybrid Energy Tracking

A combination of software- and hardware-based measurements was thus proposed, combining software-based energy measurements[54] and a power meter, which records the passthrough as well as voltage [Cao et al., 2020, *p.* 143ff.]. However, while the authors note the differences when comparing software- and hardware-based monitoring, they also recognize limitations of clearly allocating the impact between Central Processing Unit (CPU), GPU and memory on the measurements [Cao et al., 2020, *p.* 147]. This brief excerpt should serve as a small observation that the accurate measurement of energy consumption required for NLP

[53] The critique in Cao et al. [2020] was first published on arXiv in October of 2020, after the initial preprint of Zhou et al. [2021], see `https://arxiv.org/abs/2002.05829`, was uploaded to arXiv in February of 2020. The preprint was later accepted to the EACL 2021, hence the year 2021 in this citation for Zhou et al. [2021].

[54] Using the framework *experiment-impact-tracker* from Henderson et al. [2020].

models or machine learning systems in general is still actively researched and not fully formalized. A brief reference shall be included here to the experiments conducted as part of this thesis for different NLP tasks, which will be presented below in Section 8.1 for both runtime and energy consumption.

Reproducibility

A further dimension is reproducibility. First, the distinction towards repeatability, meaning that "[...] a researcher can reliably repeat her own computation", and towards replicability, i.e., that an "[...] independent group can obtain the same result using artifacts which they develop completely independently", needs to be observed.[55] Reproducibility, itself defined as the ability so that "[...] an independent group can obtain the same result using the author's own artifact"[56], as a middle ground between the two opposed definitions of repeatability and replicability will be discussed here. Still, an even finer distinction into dimensions of reproducibility, i.e., into conclusion, finding and value, has been proposed by Cohen et al. [2018]. While this discussion has merit, the higher-level interpretation of reproducibility shall suffice for this thesis.

Reproducibility for LLMs

Traditional methods, in many variants and use cases, have an advantage when compared to modern approaches in this area. Rule-based approaches, decision trees, as well as extractive approaches, can all be easily reproduced. In contrast, modern language models, due to their generative and abstractive nature, can be at a disadvantage here. The study of reproducibility for LLMs is still in its early stages. One such study noted that despite the availability of parameters such as temperature, which are intended to restrict diverse output generation and to improve reproducible results, this is not always the case [Staudinger et al., 2024, *p.* 187]. Further studies focus on different sampling strategies and their influence on the generated output [Zhou et al., 2024b]. Thus, reproducibility is an ongoing challenge for modern LLMs, while many traditional NLP models, like most decision tree classifiers for instance, are deterministic and can thus be reproduced. Exceptions in traditional models include implementations relying on a deliberate random seed, such as for random forest classifiers [Ho, 1995].

Dependence on API Access

Closely related to reproducibility is the access to NLP methods. Except for locally installed or self-hosted models, most modern LLMs are hosted services that are provided via API calls or other user interfaces from large

[55] Both quotations are taken from the ACM terminology described in more detail in: `https://www.acm.org/publications/policies/artifact-review-and-badging-current/`.

[56] See the same ACM website from Footnote 55.

LLM providers. Not only financial aspects are at play here, but also the uncertainty of updates by the LLM provider or host. Due to the black-box nature of LLMs, even more so since providers like OpenAI oftentimes do not include detailed documentation when presenting new models, a commitment to one service is a potentially risky strategy. Studies have been carried out, which display not only changes between major version updates, e.g., from GPT-3.5 to GPT-4, but also from within one model when studying the responses of one such model, e.g., with responses by GPT-3.5 that differ in major ways within a matter of months [Chen et al., 2023b].[57] Increasing the severity of this problem is the lack of transparency, which OpenAI justifies as such in the technical report for the development of GPT-4:

> "Given both the competitive landscape and the safety implications of large-scale models like GPT-4, this report contains no further details about the architecture (including model size), hardware, training compute, dataset construction, training method, or similar." [Achiam et al., 2023, *p.* 2]

Such problems simply do not exist with traditional models or other models that are installed on local machines or hosted/provided by oneself or in-house departments.

Explainability

A further dimension is explainability. Due to the black-box nature of deep learning techniques, despite efforts done in this research area, interpretability of LLMs is another clear distinction when contrasting modern methods with traditional models [Hadi et al., 2023, *p.* 21 and *p.* 24]. Traditional models, be it rule-based approaches with clear statements regarding which rules have been active in order to create a decision, or linear classifiers due to their white-box model structure, provide in many cases interpretable results, whereas eXplainable Artificial Intelligence (XAI) is still an ongoing research area [Linardatos et al., 2020, *p2.*]. However, recent achievements in incorporating reasoning within LLMs have maybe offered an approach towards some way of interpreting the results generated by language models [Huang and Chang, 2023], despite not solving issues such as biases or hallucinations [Huang et al., 2025].

Risks through Using LLMs

One final aspect is the risks associated with using and deploying modern language models. While the use of open-source software in general is dependent on software repositories containing safe and responsible

[57] The authors from Chen et al. [2023b, *p.* 4ff.] call this phenomenon *LLM drift*, which they study on a variety of applications, and which has been noted in more small-scale studies as well, see Aiyappa et al. [2023] or Shakarian et al. [2023].

code without any malicious intent,[58] the scale of software necessary to run modern models has risen immensely and has made the supervision of necessary code, packages and frameworks increasingly harder. Studies focusing on Hugging Face[59], one of the most popular platforms for distributing and re-using language models, have shown some risks when using models hosted there, such as adversarial attacks, privacy problems for the model, biases included in the models as well as the associated data or vulnerabilities within the models [Jiang et al., 2023, *p.* 2469ff.; Kathikar et al., 2023, *p.* 5f.]. Again, malicious code can also be included in libraries available for traditional models. However, the long-ranging availability of such models and comprehensive documentation of traditional methods, as well as their reduced complexity when compared to modern methods, limit security risks. Furthermore, due to the fact that many traditional models need to be adapted for the respective use case, by writing rules for an extraction task or by training a classifier on the available training data, the responsibility for accurately applying the operations lies with the user and not with the models, which the user potentially does not have sufficient insight into.

After presenting the related work on traditional methods through the SLR from Jegan and Henrich [2025a] in Section 4.1, a high-level view on other dimensions has been given in the current section. By looking at different domains in the next Chapter 5, a continued overview for other perspectives on NLP tasks will be given through the lens of engineering and design theory.

[58] Although problematic cases for such repositories have also been observed, see, e.g., `https://arstechnica.com/information-technology/2023/01/more-malicious-packages-posted-to-online-repository-this-time-its-pypi/`.

[59] See `https://huggingface.co/`.

5 Engineering of NLP Methods

The previous chapters were focused on an NLP centric perspective, which is natural due to the research field that this thesis is a part of. An overview of techniques from the past decades in Chapter 2 was thus provided, in addition to the foundations of the application scenarios specified for this work in Chapter 3. Lastly, the related work on traditional techniques and aspects of high-level features of NLP methods was shown in Chapter 4.

Leaderboards for NLP Tasks

The selection of appropriate methods for a given text processing task is also naturally based on criteria presented in the chapters above, mostly by focusing on the available metrics and evaluation criteria presented in Chapter 3. Thus, one simple approach would be to look at leaderboards[60] for the relevant NLP task and choose the model with the highest score. While this action would typically suffice, some problems might occur. First, the model listed in the first place may only be executable on high-powered hardware that is not available for every use case or target scenario, see above in Section 4.2. Following that, the model might be proprietary, thus an adaptation or fine-tuning on the target scenario might not be possible or, in other cases, might only be available via API calls and therefore locked behind licensing fees.

Other Perspectives

Finally, analyzing perspectives from other disciplines might reveal chains of thought and dimensions that are not readily apparent for the usual workflow of tackling NLP problems. This chapter will provide an analysis of such other disciplines, i.e., other domains, and offer glimpses at different perspectives. These perspectives include engineering design or theory of design, which can be seen as the same discipline, see below in Section 5.1, and product development, and will therefore directly analyze and answer research question 4 – referring to the systematic approach that can potentially benefit the selection of methods for NLP tasks. Though there are naturally major differences when conceptualizing a physical product and an NLP method, some analogies will be drawn between those different domains since there are established workflows

[60] E.g., compare `https://huggingface.co/spaces/mteb/leaderboard/` for an overview on scores across hundreds of models for the Massive Text Embedding Benchmark (MTEB) [Muennighoff et al., 2023].

and procedures, e.g., in product development, that can offer benefits even for text processing tasks.

In this chapter, a focus will be placed first on product development in Section 5.1, before presenting the observations from these domains in terms of selection criteria for choosing NLP methods in Section 5.2 and finally detailing other factors in Section 5.3.

5.1 Analogy to Product Development

Software Engineering

A domain with longstanding roots that can serve as inspiration for non-NLP perspectives is engineering design. This domain, however, needs to be properly contextualized. While software engineering as a discipline contains the same term, *engineering*, the disciplines are somewhat different, especially when concerned with NLP problems. Software engineering, as far as computer science is concerned, is defined as follows:

> "Software engineering is an engineering discipline that is concerned with all aspects of software production from initial conception to operation and maintenance." [Sommerville, 2015, *p.* 20]

The focus is therefore on the production of software first and foremost. While in both agile and even more so in traditional software development, workflows and systems are in place to offer a structured and purposeful plan, an analysis of such criteria in an NLP focused study has not been done thus far, to the best of our knowledge.

Product Development

Engineering design is rooted in other disciplines, mainly in domains such as product development [Pahl et al., 2007]. Terminologically, engineering design, in the context as it will be used in this thesis, is also known as theory of design or construction theory.[61] In contrast to software engineering, which is characterized by iterative procedures and a clear focus on the implementation, even more so in agile project development, engineering design can offer broader perspectives and other insights.

PEP

One such insight can be gained from looking at a specific process type within the product development domain, which will be abbreviated with the more formalized and established German term Produktentstehungsprozess (PEP)[62][Feldhusen et al., 2013, *p.* 11]. PEP as a concept is expans-

[61] The German word "Konstruktionslehre" is more fitting here, due to the somewhat vague and imprecise English terms [Feldhusen et al., 2013].

[62] English: Product Development Process.

ive, which is why the focus will be placed on a few aspects from which analogies to the construction and application of NLP methods will be drawn. One of the first PEP-inspired aspects is the development based on two properties as a foundation, i.e., a model and the data. Models in this context should not be confused with, e.g., specific machine learning models for NLP tasks, but rather refer to generalized patterns, from which in combination with the specific data a working method or approach can be generated. A visualization of a simple workflow is displayed in Figure 5.1.

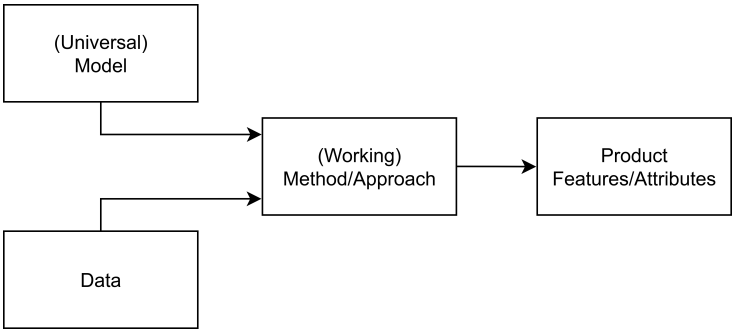


Figure 5.1: Processing workflow during PEP construction. Based on Feldhusen et al. [2013, *p.* 12].

Universal Model

Using such a universal model, provided with some data from the specific problem, a working method can be created, and some features or attributes of the resulting product can be glimpsed. In this way, similar to prototyping, a first measure of success can be visualized or theorized, which is used to further define future approaches and updates for subsequent models or necessary adaptations for the data. This, meaning processes as displayed in Figure 5.1, represents one sample workflow from the product creation domain. The specific workflow from Figure 5.1 is not necessarily fitting in full for NLP problems, since in many cases the data is the basis and an initial problem is also selecting the correct model. In Figure 5.1, a concrete implementation should be chosen and not the universal model from above, as will be realized later in Figure 9.1. Nevertheless, it is rather worthwhile to view models as a universal property instead of a defining characteristic from the beginning of approaching a new NLP problem.

Layered Process

The basis of the workflow from Figure 5.1 is the universal model as well as the data, which as mentioned differs from usual software engineering workflows. A different perspective, which aligns somewhat more

closely with software engineering, is a visualization with layers representing model, method and data, see Figure 5.2. The two incoming arrows towards the method layer are noteworthy. The development and construction process, as part of the PEP, involves an iterative procedure to create the model for realizing the final product. This process in the PEP is defined as design or construction steps [Feldhusen et al., 2013, *p.* 13]. Within one such construction step, product data, e.g., material data for a specific product, and a product method, which is once again a universal model when compared to the more concrete NLP term, are both used to create and improve upon prior versions of the product. Whereas the process from Figure 5.2 is used in the product development domain as a more abstract part of the PEP, more direct analogies towards NLP systems can be drawn. The focus on methods in the center layer can be interpreted in a straightforward manner in terms of prototyping or iterative software development.

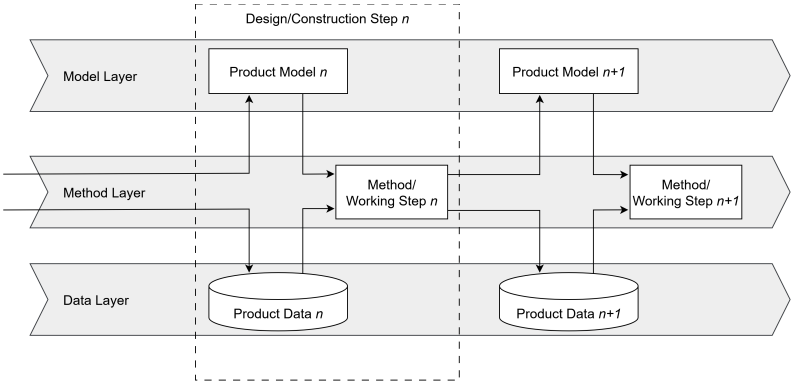


Figure 5.2: Layered development and construction process. Based on Feldhusen et al. [2013, *p.* 13].

Adapting the Layer Model to NLP Tasks

When interpreting the layers from Figure 5.2 for, e.g., a text summarization problem, the data would be the dataset in question accompanied by other surrounding parameters, which comprise the know-how of the implementing staff or the available computational resources, see below as part of the selection criteria for NLP methods in Section 5.2 and Table 5.1. The method layers can be interpreted as concrete implementations for the summarization problem, using, e.g., a LexRank [Erkan and Radev, 2004] summarization system. Finally, the model layer would be a more universal view on summarization systems such as the distinction between

abstractive and extractive models. The choice of broader models in this layer – and again differing from the term *model* like the model LexRank – results in the option of evaluating the middle layer, which encapsulates the concrete method, here LexRank.

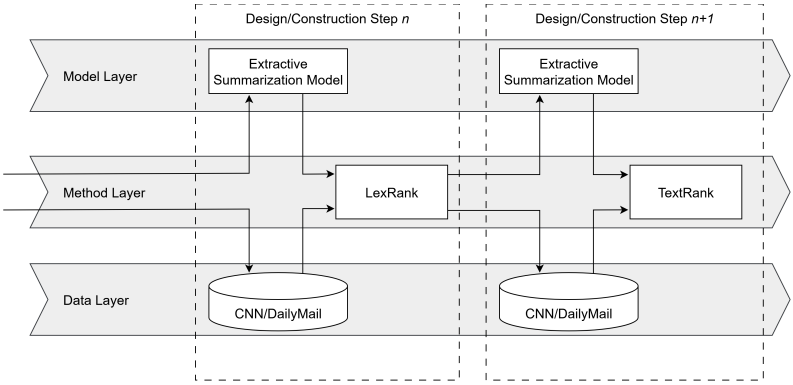


Figure 5.3: Layered development and construction process for a text summarization problem. Based on Feldhusen et al. [2013, *p.* 13].

Restrictions for Layer Model

Thus, the definition of the type of summarization approach in the top layer, be it extractive, abstractive or a different manner of category, enables a modular system in order to choose different methods and iteratively improve the system, see Figure 5.3 with two extractive summarization techniques chosen as concrete methods that will be applied and evaluated – LexRank [Erkan and Radev, 2004] and TextRank [Mihalcea and Tarau, 2004], both tested for a news summarization scenario on the popular CNN/DailyMail dataset [Dernoncourt et al., 2018]. The model layer will therefore serve as the selection criteria for determining the type of approach that will be selected and as a filter for available approaches, while the data layer provides the necessary framework, i.e., the dataset but also potentially other resource restrictions. The method layer represents available implementations, which are evaluated against each other and tested iteratively. The sequential nature of this representation is usually not realistic or efficient in practice, which is also true in the product development domain [Feldhusen et al., 2013, *p.* 18f.]. Still, the viability for planning and visualizing NLP methods with both the layer model from both Figure 5.2 and Figure 5.3 as well as the previously described universal model from Figure 5.1 will be applied to a concrete NLP use case to analyze their applicability, see Section 9.2, especially Figure 9.1 and Figure 9.2.

Sequential & Concurrent Development

The development process, starting with the requirements for a specific product, and resulting in the final product as output of this process, can be represented as a sequential workflow. Initially, a general draft of the system is produced, which is converted into a more domain-specific concrete draft of the same system. This draft leads into the integration of the system to such an extent that a product can be realized. Concurrently with these three steps, a model is created and analyzed constantly. The sequential nature is rooted in long-standing product development standards. In practice, as mentioned, measures such as property safeguarding are reflected during the latter stages, i.e., the integration phases, back towards the system draft phase, in order to continuously refine the product in its development process and resulting in iteration cycles [Feldhusen et al., 2013, *p.* 19].

Agile vs. Waterfall Models

Both agile development processes[63] and other, older project planning strategies such as the waterfall model [Royce, 1987] are still in use today in software development [Saravanos and Curinga, 2023, *p.* 4ff.]. The sequential nature of project planning, mentioned above as a somewhat limiting factor, results in a contradiction for software development projects when compared to the popular type of agile development, which has been increasing in popularity since the 2000s [Hoda et al., 2018].

Analogy to Traditional NLP Models

In a somewhat imaginative analogy, the waterfall model with its structure, step-by-step design and clear separation between stages can be compared to traditional NLP models such as decision trees or rule-based approaches, since they, too, are clearly structured with a high degree of explainability as well as interpretability and an iterative process. The step-by-step design of such traditional approaches, e.g., implemented by a simple sequence of if-else cases, is also similar to the stages within a waterfall model, that sequentially lead into the next phase. While an analogy between modern deep learning-based NLP approaches and agile development models is not apparent, the similarity between older approaches, both in project planning in general and for NLP techniques, is noteworthy. A structured and purposeful approach towards a given problem, either in project planning in general or within an NLP task, can therefore be aligned to the more straightforward processing when applying traditional techniques, instead of the more abstract use of larger neural network architectures.

Temporal Influences

When returning to product development, long-term planning stages such as material procurement have been mentioned above, which need to

[63] As popularized with the agile manifesto, see `http://agilemanifesto.org/`.

be properly situated within the development plan, in order to adhere to a timetable. Time as a factor is in industry terms usually structured according to concepts such as time-to-market for the whole product and milestones or intermediate results for smaller steps [Pahl et al., 2007, *p.* 134ff.]. Of course, time is similarly a factor for NLP or software engineering projects in general. Long-term material procurement processes are, however, usually not as relevant in comparison to product development, at most in terms of hiring new staff for the implementation or regarding hardware purchases. The latter, meaning specific hardware resources such as powerful GPUs, are a factor that has been altered due to the success of LLMs. Therefore, a small excursion will be dedicated to the recent hardware developments in the next two paragraphs.

Hardware Resource Restrictions

Several factors contributed to the difficulty in procuring GPUs for training and executing LLMs. Different use cases for GPUs such as mining operations in the cryptocurrency realm or their previous main usage as a gaming resource have contributed to an increase in cost for GPU hardware even before the rise of LLMs [Wilson, 2022]. The COVID-19 pandemic additionally impeded hardware production and supply chain processes, exacerbating the problem [Tang, 2023]. A contributing factor to GPU shortages is the strong dominance of the GPU manufacturer Nvidia in terms of providing GPUs capable of efficiently processing LLM workflows. Other GPU providers such as AMD's Radeon line or since 2022 also Intel's Arc line of chips both are far less performant when compared to Nvidia's GPUs [Chen, 2024; Smith, 2024], at least for AI-related workflows. This near monopoly worsened the situation due to scalping and more hardware shortages in general.

Demand for GPUs

The rise of LLMs exacerbated the problem due to increased demand for GPUs, on the one hand from dedicated LLM-focused businesses such as OpenAI[64], Mistral[65] or Anthropic[66], and on the other hand corporations, that applied LLMs to their core business, such as Google[67] in their search infrastructure or Microsoft[68] in their software stack. One potential approach to handling hardware shortages is using cloud options, such as renting hardware capacity from cloud providers, however, even in that case, the shortage of GPUs has been noticed [Strati et al., 2024].

[64] See `https://openai.com/`.
[65] See `https://mistral.ai/`.
[66] See `https://www.anthropic.com/`.
[67] See `https://blog.google/products/search/generative-ai-search/`.
[68] See `https://techcommunity.microsoft.com/blog/microsoftmechanicsblog/how-microsoft-365-copilot-works/3822755/`.

Potential for More Analogies

Returning to the initial argument and reference to the PEP, these processes capture long-established ideas and as shown over the course of this section, some of these concepts can be applied to NLP methods as well, such as the construction phase or the layered approach towards project planning. Significantly more process workflows or planning concepts are commonplace within the theory of design domain and are certainly a worthwhile comparison for future work.

Now that some concrete factors have been discussed, e.g., hardware requirements, a broader perspective will be presented by analyzing the lifecycle of a product in analogy to NLP tasks and the goals of such a project.

Product Lifecycle

One aspect that should only briefly be mentioned is the concept of separating three distinct phases in the lifecycle of a product, i.e., Beginning of Life (BOL), Middle of Life (MOL) and End of Life (EOL) [Terzi et al., 2010, *p*. 364ff.]. BOL refers to the phases mentioned above during product development, while the latter two phases have only been referenced in passing thus far. MOL comprises the time frame of actual use or the service phase of a product and also includes distribution as well as support, when taking the perspective of the product provider. EOL, however, is probably most commonly used in terms of software (including hardware supported by accompanying software) as their last published updates, after which the software has reached its end of life, thus no further updates are provided. In terms of product development, EOL refers to recycling and reusing the products by the company that produced them initially for disassembly or disposal, thus completing the lifecycle of the product. A direct analogy to NLP methods is less obvious here, except the EOL interpretation mentioned above in terms of software support, which is why this short discussion shall suffice here.

NLP Lifecycle

When summarizing the PEP, different phases are apparent even for NLP methods and also when planning a software project with either agile or traditional project planning. As in different planning paradigms mentioned above, some or all of the following steps are usually conceptualized within a software project, i.e., planning, development, conceptual development as well as construction, design, documentation and production [Feldhusen et al., 2013, *p*. 22]. The concurrent development of prototypes, in case of a tangible product both virtual and physical, is typically conducted in case of software as a virtual prototype, however in different shapes such as low vs. high fidelity prototypes [Rudd et al., 1996]. The PEP concludes with the production of a product, and other steps such as the continued use and even *deproduction* of a product fall outside the scope of

the PEP. In contrast, the maintenance and support of a software product can result in immense budgetary and time efforts during continued support, which represents one large distinction when compared to a physical product. Further iterations of a product can certainly be produced as well, however, in terms of design theory and concepts such as the PEP, this action would result in a new process and workflow for a new product.

Result Objectives

A final phase of a product development process can also focus on another perspective, i.e., an analysis to what degree the initial objective was completed successfully. This analysis can materialize in various ways. One such interpretation is strictly motivated from a product development perspective and based on three criteria, adapted from Feldhusen et al. [2013, *p.* 26]:

- Fulfill the customer's wishes/benefits (customer's requirements).
- Accomplish a reasonable time-to-market.
- Conform to prices in line with the market.

Fulfillment of Requirements

Adapting this criteria list to an NLP method is possible with some degree of abstraction. The first item, fulfilling the customer's requirements, is easily linked, if requirement engineering was part of the project lifecycle, usually at the beginning of a software project. In case of traditional project planning, such as the waterfall model, the gathering of requirements is an important part of the initial phase within a software project, which mostly remains fixed until the project has concluded. Agile software development, on the other hand, welcomes flexible requirements, which can change over time even in the later stages of development [Paetsch et al., 2003]. A separation in distinct phases with a static list of requirements, which is set at the beginning of a project, similar to product development is not provided in agile software development, in contrast to traditional project planning processes.

NLP Task Objectives

For NLP methods, the requirements can similarly change across the lifetime of a project, ranging from updates, additions or adaptations regarding the datasets. Further objectives can include the addition of feature requests or organizational circumstances that may have changed, such as staff rotations or others. Thus, a fixed demand analysis is often desirable but not completely realistic across the whole project lifecycle.

Continued Maintenance

The second aspect from the list above regarding the desired time-to-market can be interpreted as the time of development for the NLP method. However, it must be considered that the production of a physical product is hardly comparable to software, as mentioned earlier. Software needs

to be maintained and updated, both in terms of security and feature updates. Time-to-market can be deemed accomplishable if this criterion is interpreted as the successful delivery of a single software version. The degree of sophistication of the software, even in broad terms such as alpha, beta, release candidate or full release, is not as fixed as for a physical product.

Budget

Finally, the third criterion of the list above concerns budgeting and financial circumstances. Once again, setting a fixed price in advance that has to be met at the end of a PEP, is usually not possible for a software realization. While a certain version of a software system can be billed according to some budget, further costs are a factor that needs to be considered. Maintenance and software updates are not free and require manpower as well as financial reserves. In terms of NLP methods based on LLMs, hardware costs are another financial factor, which can rise in terms of cost depending on user counts. This increase in costs can materialize by necessary hardware that has to be purchased, if prior hardware capacities are exceeded, or certain budgets need to be expanded if the prior usage quotas for cloud access to LLM models are surpassed. In the end, a final budget is thus not easily fixable for software in present times.

Ongoing Use & Lessons Learned

In NLP methods many use cases are intended for continued use, such as in search systems or for processing data that is arriving constantly, or at least in recurring intervals. Thus, a finished result without direct follow-ups in terms of updates or further developed versions is seldom the case for software, in contrast to a physical product, which is produced based on a final specification.[69] Therefore, a different perspective on the continued use of software and an updated workflow is necessary, which can, once again, take different forms. Lessons learned, as a means of recapitulating a project and focusing on the most important and sometimes unexpected insights, is one such form. The structured compilation of these lessons is one way of ensuring the transfer of past experiences into a new project.[70]

Retrospective

A similar concept, and formalized as part of agile software development, is the retrospective. Different definitions regarding retrospectives within agile development are in use. On the one hand retrospectives can occur multiple times at the end of a regular interval, e.g., after a sprint or scrum [Derby et al., 2006]. On the other hand, a final retrospective at the end

[69] In this interpretation, further iterations upon a product, which are produced subsequently, and other forms of further manufacturing runs are deliberately disregarded for conciseness' sake but naturally need to be discussed in a more complete analysis of product development.

[70] Lessons learned, as structured information, are saved in a variety of libraries and collections, e.g., the NASA Lessons Learned System, see `https://llis.nasa.gov/`.

of a project can suffice [Kerth, 2013]. The former interpretation of retrospectives, at fixed intervals or at a fixed point, is one key aspect of agile software development, whereas the latter interpretation lends itself more to traditional project development [Derby et al., 2006, *p.* 141; Kerth, 2013].

Self-Reflection

Both concepts, lessons learned and retrospective, can be rooted in psychological processes such as self-reflection. Learning as one key goal is central in self-reflection, as is taking on outside perspectives in order to review and properly contextualize past experiences [Ispaylar, 2015]. The goal of this process is to gain a deeper understanding about oneself, both in private and professional contexts, and to improve future decision making and behavior. This concept is popular in domains apart from software engineering, both as defined above in psychology and in teacher training and didactics [Rahm and Lunkenbein, 2014]. The lessons from a more general interpretation of retrospectives and lessons learned, as part of self-reflection, are at least a worthwhile thought experiment during purposeful problem solving, not exclusively for NLP methods but naturally also applicable there.

The structured approach for realizing a product, be it material or software, involves an early phase of requirements engineering. While this phase has diminished in importance in agile software development, lessons from the requirements engineering process including elicitation, analysis, documentation, validation and management, see Paetsch et al. [2003, *p.* 308ff.], are certainly a worthwhile point of reference for this chapter. However, strictly defined processes akin to requirements engineering from a software engineering perspective, see Sommerville [2015, *p.* 101ff.], are not desired in this thesis. Instead, a broader interpretation of lessons from other domains, as has been shown in this section, combined with observations from NLP tasks and related domains will be analyzed and serve together as a methodological guide in approaching NLP problems. A collection of these lessons and observations will be compiled and organized in the next section.

5.2 Selection Criteria for Choosing NLP Methods

Methodology for NLP Methods

After the overview regarding product development has been presented, with a focus on the PEP and contrasting as well as similar characteristics of NLP methods in the previous Section 5.1, a separate point of view will be analyzed in this section. A different methodology is developed with lessons from product development in mind, which is intended to present

shortcomings in common workflows of realizing NLP methods and to show an alternative way of approaching text processing problems.

Impact of Iterative Workflows

The first severe difference that will be discussed is the dominance of iterative workflows, actions and models, most prominent in agile software development. When addressing the fast nature of iterative development cycles defined by daily meetings between developers and business representatives, short iterations organized in sprints and larger organizational frameworks such as Scrum[71], a famous and probably extreme version of this mantra was introduced by Mark Zuckerberg during the earlier days of Facebook: "Move fast and break things".[72] While the quote was rescinded or at least softened by Zuckerberg himself,[73] it was also characterized as representing the negative effects of social media and online discourse, in terms of fake news, online extremism and its effect on society in general [Taplin, 2017]. The discussion on societal effects shall remain out of scope for this thesis, but effects of agile development methods on critical aspects such as security, failure tolerance and ethical consideration shall be nonetheless discussed here.

Systematic Approach

Once again, a point of view from product development in proposing a different workflow with special considerations on quality and safety can counteract potentially negative results of the iterative software development. The application of a systematic approach during the whole production process, including the design as well as the production phase, can be a critical factor in preventing failures or flawed product quality in general [Pahl et al., 2007, *p.* 517ff.]. A similar chain of thought, or at least lessons from such product development concepts, can serve as new selection criteria when conceptualizing and realizing an NLP method.

Standards and Norms

Design for quality as a cornerstone of product development is rooted in the rationale that most product defects can be directly linked to insufficiencies in planning stages or poor product development [Pahl et al., 2007, *p.* 517]. To combat this problem, various practices were created and structurally formalized. International standards such as DIN ISO norms[74] and

[71] See, e.g., the official Scrum guide by its inventors Ken Schwaber and Jeff Sutherland at `https://scrumguides.org/`.

[72] See, e.g., Wired discussing the letter from Zuckerberg to investors, which included the quote: `https://www.wired.com/2012/02/zuck-letter/`.

[73] See, e.g., `https://www.businessinsider.com/mark-zuckerberg-on-facebooks-new-motto-2014-5/`.

[74] See, e.g., the DIN ISO 9000 (`https://www.iso.org/standard/45481.html`), which defines fundamentals and vocabulary for quality management systems, and the DIN ISO 9001 (`https://www.iso.org/publication/PUB100373.html`), which defines requirements for organizations in order to conform to a quality management system.

interdisciplinary project structures were created, the latter with representatives from design, product planning, assembly, distribution, supplier and customer teams all participating in the product creation and tracking processes, combined under the umbrella term *simultaneous* or *concurrent engineering* [Pahl et al., 2007, *p.* 138ff.].

Interdisciplinary Work

While the direct contact between stakeholders and developers is at the forefront of agile software development, the close interlocking between even more interdisciplinary teams outlined in simultaneous engineering would ensure even broader perspectives and higher quality in the long run. The trade-off would be the designation of an interdisciplinary team, preferentially for the duration of the whole project, with a correspondingly increased budget and staff.

Methodical Design

The personnel certainly play a role here, especially in providing out-of-domain insights, however, design principles from the process formalized as *methodical design*[75] are equally helpful and listed in Pahl et al. [2007, *p.* 227ff. and *p.* 517ff.]. Not all design principles are directly translatable to software. For example, the distinction between direct and indirect safety principles represents two types of measures. On the one hand, the prevention of danger through the choice of components, fail-safe measures and redundancies directly contributes to safety, while on the other hand additionally implementing protective systems or devices, which activate after the former measures fail, aim for the same goal [Pahl et al., 2007, *p.* 249ff.]. In terms of software, safeguards or assertions can play a role here, which can guard the user's or the systems' output from producing ill-advised results. Also, the redundancies or fail-safes can be transferred with a hierarchical or hybrid design, meaning the availability of a second model, which can be applied if the results from the primary model were recognized as error-prone or ill-formatted.

Safety Concerns for Software

Some further quality measures are clearly focused on haptic products and thus do not correlate with software, e.g., design against corrosion and to minimize wear and others, which are intended to cover environmental influences on the materials [Pahl et al., 2007, *p.* 321ff.]. When other quality measures however are conceptualized without the direct focus on materials, more analogies towards NLP methods are apparent once again. The principle of stability is intended to handle disturbances that may arise during the use of the product and how to cancel those disturbances or at least to mitigate them [Pahl et al., 2007, *p.* 301ff.]. Product development,

[75] Once more, the German translation "Methodisches Entwerfen" is perhaps more fitting here, see `https://www.vdi.de/richtlinien/details/vdi-2223-methodisches-entwerfen-technischer-produkte/`.

depending on the business area in question, might have to handle pressure, temperature or other factors, while software needs to handle fewer tangible factors and resources.[76]

Service Availability

Starting with hardware, the accessibility of the desired service is critical, for example by guaranteeing the availability of servers through redundancy. Other hardware-adjacent factors revolve around accruing data and requests. Measures such as usage quotas or a maximum input length need to be considered to guarantee the uptime of the server and to prevent an overload. The risk of overload through multiple users, not necessarily by malicious actors through DDoS attacks but potentially by an unexpected increase of organic users, is a further factor that needs to be considered here, and which can be mitigated through usage limits or by limited signups. These measures result, however, in a trade-off by decreasing the availability of the service.

Risks for Software

The concrete usage of software entails further risks. Sanitizing the input provided by the user, either by input validation or by using other methods such as prepared statements, are measures in order to prevent bad actors from accessing or modifying data. Malicious intent is not the only reason why incoming data should be properly analyzed. User-provided data needs to be validated, in terms of schema, length or other metrics, depending on the use case, to be properly processed by the provided software. The use of LLMs, while potentially providing more flexibility through chat-based interfaces for the user, would result in other, probably also unexpected consequences. Undesired data as input would, without any further preprocessing prior to querying the LLM, be processed as regular input and thus also generate output that is potentially unfit for the defined use case.

Reproducibility & Explainability

The *principle of stability*, besides handling outside disturbances, is also intended to ensure stable behavior and characteristics [Pahl et al., 2007, *p.* 301]. This key aspect can be compared to both reproducibility and explainability for NLP methods. Reproducibility is here defined as the ability to produce the same results with the same input when applying the same queries, which can be achieved using various models, e.g., by using traditional models as presented in Section 4.2 and which will be discussed below in Sections 6.4.1, 7.4.1 and 8.2. Explainability enhances stability of

[76] In a broader discussion on software engineering, textbooks such as Sommerville [2015, *p.* 306ff.] have discussed safety aspects for software development before, in terms of reliability, safety, security and resilience engineering. Instead, a more focused discussion on NLP methods will be presented here, which will naturally also include lessons from the software engineering domain.

the system due to the transparency enabled through models that can offer rationales for their produced output, such as mentioned above as well in Section 4.2 and further down in Sections 6.4.2, 7.4.2 and 8.4.

Models & Evaluation Criteria

When starting the design process, two often interlinked decisions are of critical importance, at least for NLP methods: The choice of model and choice of evaluation criteria. A rather simplistic line of action could be choosing the "main" quantitative evaluation metric for the respective use case, referencing an up-to-date leaderboard for the same use case and choosing the top-ranked model, as mentioned at the start of Chapter 5. This approach has several shortcomings, the first one revolving around the limitations or problems of the metrics themselves. As will be shown below in Section 6.2 and Section 7.2, purely relying on quantitative metrics is in most cases not advisable, at least not without using other metrics or an accompanying qualitative evaluation. Also, the use case displayed in common leaderboards is naturally represented by models that have been evaluated for standard datasets, e.g., mentioned above in Chapter 3 for the major research areas analyzed in this thesis. Thus, the focus is often on optimizing models for those datasets and their performance might not be generalizable for the intended use case. The parameters, available data and other circumstances for a specific NLP method need to be properly compiled, e.g., by providing a requirement analysis and focus on quality measures, as presented in this chapter. This analysis can then serve as a basis for the selection of models and metrics, in turn leading towards the realization of the NLP method.

Comprehensive Perspective for NLP Tasks

As mentioned at the beginning of this chapter, the iterative processes of agile software development usually do not focus on concepts such as the principle of stability from the outset whereas the focus is on working software in quick iterations. By incorporating lessons and concepts from the PEP and related domains such as methodical design, criteria for the creation of NLP methods can be collected that enable a perhaps more comprehensive perspective with a stronger focus on quality for software development.

List of Criteria

In order to collect the criteria presented from the previous Section 5.1 and current Section 5.2, a list of criteria[77] is proposed in Table 5.1. The list of criteria is not all-encompassing, but provides a selected, sensible and useful overview that is helpful when conceptualizing a method for a text

[77] This visualization was inspired by the discussion of product costs during development of a product in Feldhusen et al. [2013, *p.* 143ff.] and by the utility analysis as part of the PEP in Feldhusen et al. [2013, *p.* 390ff.], in particular the matrix in Feldhusen et al. [2013, *p.* 392].

processing problem, from the perspective of this thesis and, particularly, the learnings from this chapter. Also, the characteristics mentioned in columns two to five are flexible and should be adapted to each use case. The order of parameters is also adjustable. The chosen order was selected with a chronological perspective in mind, i.e., parameters that should be defined at the outset when working on an NLP method are listed up top. A brief primary explanation for each parameter from Table 5.1 is given in the following, which is just one non-exclusive interpretation of each parameter.

Parameter	Characteristics for NLP Method			
Budget	-	Small	Medium	Large
Time Frame	-	Short-term	Mid-term	Long-term
Headcount	Single	Small	Medium	Large
NLP Skills	N/A	Basic	Intermediate	Proficient
Developer Skills	N/A	Basic	Intermediate	Proficient
Other Skills	N/A	Basic	Intermediate	Proficient
Data Availability	N/A	Little	Medium	Much
Data Size	Single Token	Sentence	Paragraphs	Large
Data Quality	Unknown	Low	Medium	High
Data Structuredness	None	Low	Medium	High
Model Count	Single	Two	Multiple	-
Metrics	Quantitative	Qualitative	Quant. & Qualit.	-
Hardware Resources	-	Low	Medium	High
Customer Size	None	<100	100-10,000	>10,000
Availability	None	Selected Times	Fixed Intervals	Constant
Portability	None (Specific)	Low	Medium	High
Adaptability	None (Specific)	Low	Medium	High
Explainability	None	Low	Medium	High (Required)
Reproducibility	None	Low	Medium	High (Required)
Risk Factors	-	Low	Medium	High
Documentation	-	Low	Medium	High

Dimension Overview

Table 5.1: Dimensions relevant for planning and structuring NLP methods.

Budget refers to financial means designated for the project. The characteristics mentioned here – small, medium and large – are deliberately vague since this parameter will vary widely depending on the project. A budget of "none" is conceivable here as well, e.g., for leisure projects by enthusiasts, private individuals or interest groups.

Budget

Time Frame **Time Frame** refers to different time-related metrics, e.g., the intended duration of the development process or the usage of the produced service or method. The chosen interpretation here relates to the latter variant, meaning, e.g., short-term projects referring to the processing of a dataset without follow-up work or a perpetual service for customers. Long-term projects, however, require the software potentially for an undetermined time, thus for a longer period.

Headcount **Headcount** refers to the team size that is available for generating the software. Single refers to only a single developer, while small, medium or large teams are conceivable here. This parameter is naturally heavily linked to other parameters such as budget and available personnel.

NLP Skills **NLP Skills** refers to the degree of skill in applying NLP techniques but can also include background knowledge of linguistics or corpus linguistics. These skills can be differentiated from the following parameters by focusing here on NLP skills for the application of code on textual problems.

Developer Skills **Developer Skills** refers to the degree of skill in incorporating the produced software into a larger setting, e.g., in a web service or an existing infrastructure, which in recent years has been commonly covered by DevOps through encompassing deployment, operations management and automation procedures.

Other Skills **Other Skills** refers to all other proficiencies necessary for the specific project, including but not limited to language, team compatibility, team hierarchy, communication and others. This parameter is used as an umbrella term, representing all other factors relevant for the characteristics of team members.

Data Availability **Data Availability** refers to the amount of data that is available beforehand, either as simple examples provided by the client or as training data, which can severely impact the choice of models that are suitable for the method. Regarding LLMs, the context window can play a large role here, therefore limiting the amount of data that is processed at once, see, e.g., the discussion of text segmentation applications in Section 3.4.

Data Size **Data Size** refers to the amount of data processed for each query or processing step. Once again, this parameter is highly dependent on the specific task area, ranging from single tokens, e.g., for search queries, to texts with hundreds of paragraphs in case of text summarization. This parameter not only impacts the choice of model but also the hardware resources factor, due to requirements in terms of main memory and storage.

Data Quality refers to different quality aspects regarding the data. When referencing available data beforehand, consistent formatting as well as adhering to a pre-defined schema can play a role here. Moreover, data quality can also refer to the incoming data that is provided as input for the produced software, which can vary widely and needs to conform to certain standards, depending on the method used, in order to be processed. The principle of stability mentioned above in this section is a relevant concept regarding this parameter. Data Quality

Data Structuredness refers to the presence of a schema, data definition or other structured text, to which the data, that needs to be processed, or incoming data from users needs to conform. Chat-based interfaces usually do not require such a structure, which is why "none" is one possible option here, whereas various degrees of structuredness are possible as well. Such data can range from only processing incoming data that was verified by pre-defined schema file, e.g., an XML schema file,[78] to the adherence of a character limit. While appearing related to the previous parameter of data quality, data structuredness refers purely to the existence of a predefined structure or lack thereof. Even the quality of structured data can differ widely, which would be included in the former parameter data quality. Data Structuredness

Model Count refers to the number of models in use for the specific software. This amount has an impact on the development of singular or multiple methods and furthermore on how to select which model to use in case of multiple models, which can be done with or without user input. Staggered and hybrid methods, such as boosting classifiers or sequential summarization systems, see Sections 2.2.2 and 8.5 respectively, are imaginable as well. Model Count

Metrics refers to the different evaluation criteria that are intended for the software. This parameter should certainly include the evaluation metrics, and should, if possible, be chosen before a method is implemented. This decision therefore impacts the selection of a certain model, not limited to purely quantitative metrics, but also referring to qualitative evaluation procedures, presented in more detail in Sections 6.2 and 7.2. Furthermore, the parameter can also incorporate ongoing evaluation processes after the software has been created and has been in use. Metrics

Hardware Resources refers to the requirements that need to be fulfilled in order to implement the software. These requirements can include a certain amount of processing power, memory or GPU capabilities, which Hardware Resources

[78] See, e.g., the W3C documentation at `https://www.w3.org/TR/xmlschema-0/`.

are naturally closely influenced by other parameters such as customer size or model selection. The integration into existing infrastructure has to be considered here as well, which relates to parameters such as developer skills. A further dimension in this regard can be the distinction between providing the hardware in-house compared to renting hardware from cloud-based providers.

Customer Size

Customer Size refers to the prospective user base. The range starts here at none since not all NLP methods are planned for customer use, but are instead, e.g., intended for the internal processing of a specific dataset or as a means of executing a certain method at fixed intervals internally, without providing a user interface for customers.

Availability

Availability refers to the setup of the prospective service for customers, i.e., when the software is available for accepting queries or incoming data. Similarly to the previous parameter, in case of purely internal use, "none" is conceivable here as well. Otherwise, different operational scenarios are possible, ranging from constant availability to more selected units of time, fixed intervals or only a pre-selected amount of time frames.

Portability

Portability refers to the importance of providing software that can be easily transferred to other servers, locations or providers. This parameter thus includes the distinction between providing the implementation and execution in-house on internal hardware compared to using service providers or cloud-based services. The characteristic "none (specific)" can involve specific circumstances with critical or sensitive data, which needs to remain on internal servers. Other interpretations are conceivable here as well, e.g., the use of development environments such as containerization in order to easily install the produced software on another system or server. This parameter, once again, is closely related to development skills and hardware resources.

Adaptability

Adaptability refers to the flexibility of the produced software regarding changes with the data or processing. Changes to the data structure can necessitate updates regarding the processing pipeline, whereas change requests from customers or other participating parties can result in adaptations for the existing software. Depending on the circumstances of the specific method such updates can or should be considered beforehand, especially in quickly changing scenarios or contexts. Extensibility is relevant for this parameter as well, regarding the addition of other datasets and more or updated models while also being able to further process previous datasets and to use prior models.

Explainability

Explainability refers to the degree to which the produced output of the models has to be explained, e.g., by tracing cases for decision trees or by

referring to confidence values or indices in classification systems. Similar to other parameters such as portability and risk factors, the presence of sensitive data is a critical aspect when providing an analysis of this parameter for the specific use case. For example, applications on texts from the medical or judicial domain are highly critical and should not be exposed to hallucinations or other problems such as biases within LLMs.

Reproducibility

Reproducibility refers to the capability of a method to produce the exact same output given the same input. As with the previous parameter and other parameters dealing with sensitive data, confidence in reproducing the same result is of high importance in many use cases, e.g., for classifying problematic texts, be it hate speech detection or spam identification. Repeatability and replicability as similar concepts play a role here as well, as has been mentioned above in Section 4.2, and which pose a problem for generative modern methods.

Risk Factors

Risk Factors refers to the wide variety of risks that can potentially be created when using text-based methods. The generation of hallucinations or simply false statements, produced by LLMs or other abstractive and generative language tools, can result in tangible problems, fees or other negative outcomes for the service provider.[79] Thus, an assessment of such risks is necessary when dealing with sensitive data, especially if the generated texts need to be factually correct. In that case, the selection of models should be limited or the application of other guardrails is recommended.

Documentation

Documentation refers to the degree of documentation that is designated for the provided NLP method. Depending on the project planning system in use, more or less documentation is already scheduled through the project plan, e.g., more in case of traditional waterfall project planning and considerably less for agile software development. Nevertheless, some degree of documentation is required and strongly recommended in terms of comprehensibility and in case of absences or loss of personnel.

Justification for Criteria

The list presented above is a non-exhaustive interpretation gained from analyzing NLP problems by the author of this thesis, as well as by incorporating literature analysis from both NLP research and other domains such as engineering design, software engineering and others. While obvious factors such as budget, time frame or customers size, at least from a product development perspective, were still included since they natur-

[79] One of the more prominent reports in recent years concerns a chatbot provided by the Canadian airline Air Canada, which falsely awarded a customer a reduced fare. Later, a court ruled in favor of the customer, see `https://www.forbes.com/sites/marisagarcia/2024/02/19/what-air-canada-lost-in-remarkable-lying-ai-chatbot-case/`.

ally remain relevant for NLP problems as well, a stronger focus was given to the dimensions specifically relevant for projects developing text processing methods. Thus, just as an example, the data dimensions were split into four parameters accordingly, a dimension each for data availability, size, quality and structuredness, which are all relevant for NLP tasks. Due to the focus on a select number of dimensions, this list is far from complete. Further dimensions will be discussed in the next Section 5.3 and are presented in the literature, e.g., by Feldhusen et al. [2013, *p.* 136], who collect potential topics for requirement engineering including other dimensions such as advertising, warranty period and more, thus clearly intended for product development.

Uniform Framework

A concrete utilization of Table 5.1 was conducted for the NLP task of text segmentation below in Table 9.1 in Section 9.2, which applies the presented dimensions from this chapter onto a specific use case and therefore presents the theoretical draft from this section more practically. When creating Table 5.1, certain criteria were noticed, that have been absent in traditional NLP problem solving or at least have been neglected in papers, tutorials and other documentation, such as NLP skills or developer skills, which can nonetheless be an important aspect for adhering to a project plan and to the successful realization of the proposed software. In focusing also on such factors and a broad encompassing view on other dimensions, as displayed in Table 5.1, the intended goal of this table is to provide a more uniform framework or broad perspective for analyzing use cases, not only to achieve better NLP methods, but also to include a cost benefit analysis with engineering criteria in mind from the start. A closer look at other *softer* factors, which were not easily represented in clearly defined criteria in Table 5.1, will be displayed in the next section.

5.3 Soft Skills and Other Factors

Softer Factors for NLP Tasks

While Table 5.1 provides a broad overview across different characteristics fitting for a potential NLP method, a closer look at a few selected criteria shall be included in this section. Other requirements, such as time frame, hardware necessities or budget, see above in Table 5.1, are candidates for prominently featured characteristics that seem obvious for new NLP and other programming projects. In this section, *softer* factors will be interpreted in terms of their importance and fit.

Interdisciplinarity

The first factor discussed here is *interdisciplinarity*. Interdisciplinary work is intended to bridge departments and teams early in the develop-

ment process in order to set the groundwork for simultaneous or concurrent engineering, which in turn leads to faster development, cost reduction and quality improvement [Feldhusen et al., 2013, *p.* 31ff.]. As part of interdisciplinary work, and as recommendations or even requirements for achieving the benefits mentioned above, new aspects are created for the teams concerned with such tasks, as listed by Feldhusen et al. [2013, *p.* 33], such as shared language and communication procedures or close information exchange, which will be described in the following.

Communication & Project Plan

The setup of a common language and definitions of key terms is necessary in order to enable clear communication and prevent ambiguities or misunderstandings based on different backgrounds between teams from various disciplines or domains. Closely related to these common understandings is an early transfer between departments involved with the development as well as a common basis in terms of communication, organizational and other tools and software, so that the information exchange and transfer is ensured. The project plan, irrespective of the type of planning such as traditional project management or agile project management, depends on the timeline and milestones, for which the different teams are integrated. Thus, the design process is structured more methodically and streamlined somewhat in order to conform to the interdisciplinary focus [Feldhusen et al., 2013, *p.* 32ff.]. The parallelization of processes is one consequence of these steps as well as one further condition to enable interdisciplinary work.

Linguistic Domain Experts

While these aspects are recommendations for generic projects irrespective of their domain and would be helpful for projects in different sizes, organizational structures and other dimensions, the applicability of such interdisciplinary chains of thought in terms of NLP methods will be briefly discussed. The first and probably most clear-cut connection is the link between the development by software engineers or developers in general and linguistic and/or domain experts of the data or problem in question. The potential of linguistics for NLP methods, as has been laid out in Section 2.1, is broad, but the details in terms of preprocessing decisions[80] depend on a closer analysis of the concrete use case [Krouska et al., 2016; Maslej-Krešňáková et al., 2020].

Necessary Adaptations

The rather straightforward procedure of choosing the top-scoring model from a leaderboard for an NLP task is naturally a starting point as mentioned above at the start of Chapter 5 and in the previous Section 5.2, and in practice is certainly used, but it necessitates adaptations for the

[80] Choices are, e.g., which stemmer to employ or if stop word elimination is necessary and if so, deciding which list of stop words is the most suitable.

actual problem to be solved. Such adaptations might be small in nature, such as transforming the input data into the required data format, or more widespread. While simply using a model initialized for, e.g., English[81], can work, if the data is also in the same language and of the same format or domain. However, slight changes in the data can have a massive influence on the applicability of the chosen model on the selected use case, see also above in Table 5.1 regarding the factors adaptability and portability. When transforming an approach created for English, e.g., into German, adaptations are necessary in various parts of the pipeline. As a concrete example, the choice of which pre-trained model to use as a basis for the selected downstream task, which preprocessing steps or scoring functions to employ, are all relevant decisions that need to be considered, see, e.g., the German text simplification system presented in Section 7.3.2 and Fruth [2022]. In this case, the simple fact of being a native-tongue speaker has enabled the adaptations for German, so no explicit domain, linguistic or language expert was necessary to implement and evaluate a first version.

Domain-Specific Data

In another case, a text classification use case was applied on a dataset provided by an insurance company, see also Section 7.3.2 and Raab [2023]. The creation of the dataset used for training the models was only possible after manual cleaning and preprocessing the data, after which extensive consultation with the insurance experts took place, who furthermore also labeled the datasets themselves. While the capability of data processing tools and frameworks has only risen in recent years, a distinction between data expert and domain expert is still noteworthy. As has already been mentioned by Viaene [2013], data scientists are not able to fully analyze and comprehend their data without insights provided by domain experts. Only an interdisciplinary perspective as well as "multiskilled team" [Viaene, 2013, *p.* 12] can fully realize the potential of data science as a discipline.

Personnel Skills

A second dimension, which will be also briefly discussed here, is the requirement of a particular set of skills for the engineers tasked with the implementation. As presented above in Table 5.1, different types of skills have been included there already, namely NLP, developer and other skills. While these types were already described in the previous Section 5.2, the catch-all category of *other skills* can include the softer factors mentioned above. The interpretation of the previously mentioned comprehensive list of potential topics for requirements of a product, see also Feldhusen

[81] The focus on the English language, as has been the case for many years in NLP research, will be the focal point of later discussions in Section 6.3 and Section 7.3.

et al. [2013, *p.* 136], mentions consulting and training as dimensions included in the organizational requirement category specifically relevant for the workforce. These aspects are relevant for these softer skills as well.

Specific Skills

While the degree of NLP and developer skills can be presented in concrete terms, e.g., the proficiency in a certain programming language or the knowledge of a version control software in terms of deploying software, a less-defined factor involves the need for further training in order to produce the desired software. When applying a method from online sources, e.g., Hugging Face or the provided code for a paper on GitHub, the software requirements are usually presented, meaning which software and packages need to be installed to use the model. However, it is often not readily apparent which other skills the user needs to possess in order to adapt the method or to understand in what way the method needs to be adapted for the use case in question. Examples in this realm could be the special properties of certain languages, the impact of using different preprocessing steps or the understanding of a certain technique that is already part of the pipeline such as vector space transformations. Circumstances in the organization or department, wherein the software is being developed, can further create the need for training a certain skill, e.g., for project management tools, DevOps frameworks, communication software or other technical requirements. The adaptability within an interdisciplinary setting can also play a role in terms of soft skills, requiring the capability of working in multiskilled teams and the integration of feedback from domain experts.

Recommendation for Project Initialization

The rather vague description of the second dimension regarding skills is on purpose due to the wide spectrum of potential problems. The nature of training and education that are necessary for the problem in question is equally flexible and variable, which is why an analysis at the start of a project is recommended if such *softer* actions in this realm are necessary.

Other Dimensions

Further dimensions are certainly imaginable and have already been discussed in other work. Risk management, see Table 5.1, change management[82] as well as further conditions such as standardization or patents and their respective role as well as the impact on the produced software, discussed, e.g., in Feldhusen et al. [2013, *p.* 199ff.], can all matter, depending on the project. Also, a brief reference to latter parts of a project should be included here as well, i.e., lessons learned or a retrospective, see Section 5.1, which can be beneficial factors for follow-up projects and, when

[82] Change management as a dimension will not be further discussed in this work but is certainly also relevant in terms of DevOps or version control systems and also from an engineering perspective, see Feldhusen et al. [2013, *p.* 110ff.].

designated at the start of a project regarding its final stages, are already a fixed part of the project timeline.

Review of this Chapter

Contextualizing the learnings from this chapter towards the larger research questions from this thesis (see Section 1.1), connections can be drawn from various stages of the PEP, such as layered processes differentiating models, methods and data layers, as seen in Figure 5.3. Further analysis of critical aspects of product development such as resource procurement can be linked towards hardware requirements, which are especially relevant for modern models with increased computational demands like the requirement of GPUs for rapid processing. In a more focused manner, a table was used in order to present a wide array of perspectives for potential NLP methods in Section 5.2, see Table 5.1. The interdisciplinary point of view necessary for not only product development but also the development of NLP methods is further presented in the last section of this chapter, as well as comments regarding the choice of technique for a prospective NLP method accompanied by two concrete examples of such choices as text simplification and text classification and the importance of soft skills for developers.

When surveying this chapter, a direct link towards traditional methods, as already defined in a rather generic way early on in this thesis in Section 2.5, depends on the actual use case of the NLP problem that needs to be solved. Certain dimensions and factors from the discussion in Section 5.2 exhibit clear advantages of traditional techniques such as the principle of stability or dimensions from Table 5.1, e.g., explainability and reproducibility. Apart from these links towards traditional techniques, other factors presented in this chapter should be generally observed in NLP methods, regardless of the chosen approach. Layered perspectives during early planning stages, know-how of the hired employees, potential hardware requirements and interdisciplinary work can all play a role during the development of a desired text processing method, and a higher priority should be given to such factors. Therefore, research question ❹ has been addressed comprehensively throughout this chapter for engineering and design theory domains, and their perspectives should be considered for realizing NLP methods. After presenting a high-level view on development, planning and other factors, precise problems common in current NLP will be detailed in the next chapter in order to examine the potential for traditional methods and to raise awareness for such problems.

6 Challenges in Natural Language Processing

After the analysis of other domains outside of NLP and their potential impact on NLP projects was presented in the previous chapter, a focus will be placed once again on actual NLP methods in this chapter, especially on LLM-based approaches. Common problems will be presented that can be observed by both analyzing the literature and by describing actual applications on concrete NLP tasks in the following Chapter 7. The problems collected here can be classified in various ways, e.g., impacting the output produced by the NLP methods or difficulties at the outset of approaches when porting methods to other domains as well as languages or also various evaluation aspects.

Hallucinations

First, regarding the initial class mentioned just above, Section 6.1 will present the problem of *hallucinations*. They have been a phenomenon for generative or abstractive text processing tasks in the past several years as well as in other domains such as computer vision since the early 2000s and gained particular notoriety during the rise of LLMs since 2023 [Maleki et al., 2024]. This problem type is generally restricted to generative models, whereas many traditional models, e.g., extractive text summarization methods, do not fall prey to this issue.

Lacking Metrics

Afterwards, a problem unfortunately inherent for all NLP tasks will be discussed in Section 6.2: The restrictions of quantitative metrics for evaluating NLP methods. As just one example, without going into too much detail before presenting them in the actual section, the application scenario of text summarization is per default evaluated by using the metric ROUGE [Lin, 2004b]. While the metric is universally used within this research field, various problems have come to the forefront including the reliance and fit towards the reference summaries provided within the datasets, which attributes potentially valid summaries worse scores due to their disparities with the gold summaries [Aumiller et al., 2023]. Metrics for other tasks exhibit similar but also different limitations and the reliance on such evaluation metrics for many NLP applications is a noteworthy discussion point.

Focus on English

Another dimension of problems will be approached in Section 6.3, i.e., the focus of methods on the English language. Groundbreaking advances in text processing research naturally have in the past been discovered by research groups from Anglo-American countries. Additionally, due to English being the largest global language in terms of native as well as second-language speakers,[83] the biggest available datasets are usually in English. Examples for ground-breaking discoveries from anglophone countries include the advances through statistical methods within NLP in a variety of tasks [Manning and Schutze, 1999], the rise of word2vec [Mikolov, 2013] and later the transformer architecture [Vaswani et al., 2017], which in turn led to the breakthrough of LLMs, not necessarily restricted to ChatGPT, but initialized by OpenAI [Wu et al., 2023]. While the latter discoveries enabled multilingual and cross-lingual operations, a large part of the underlying data is still in English.[84] Drawbacks such as worse performance and required adaptations for languages other than English are apparent and will be discussed.

Other Challenges

Further challenges will be presented more briefly in Section 6.4 such as reproducibility in Section 6.4.1, explainability in Section 6.4.2 and efficiency in Section 6.4.3. All these challenges are part of a larger discussion later on in this thesis, first by showcasing selected use cases in Chapter 7 and afterwards when discussing potential benefits of using traditional approaches, which will be displayed in more detail in the subsequent Chapter 8. Research question 3 – referring to potential benefits of using traditional models – should be considered throughout this chapter, when challenges for modern LLMs as well as all other NLP techniques are noted, which can provide a basis for potential benefits of traditional techniques. These benefits are potentially viable in two distinct manners, on the one hand through the absence of limitations when applying traditional techniques or on the other hand through less severe restrictions by using traditional techniques. The first challenge presented here will be the occurrence of hallucinations in abstractive and generative NLP approaches.

[83] See https://www.ethnologue.com/insights/ethnologue200/.

[84] See, e.g., https://commoncrawl.org/blog/expanding-the-language-and-cultural-coverage-of-common-crawl/.

6.1 Hallucinations

Rise of Hallucinations

Hallucinations have been a major problem for NLP tools for some time, but the impact of hallucinations has begun to appear in a widespread manner due to the rise of LLM-based approaches and especially by using chat-based interfaces such as ChatGPT [Huang et al., 2025, *p.* 5ff.]. Prior to the analysis of output generated by chat-based LLMs as known today, previous abstractive or generative systems have already exhibited hallucinations, e.g., in abstractive text summarization tasks [Huang et al., 2021; Maynez et al., 2020].

Origin

Generative text processing tasks were, however, not the origin of the phenomenon. The earliest source of hallucinations, apart from other definitions within medical contexts, can be traced back to a discussion on virtual inputs in neural network completion that produce "phantom experience and hallucination" [Thaler, 1995, *p.* 63]. These first discussions appeared in theoretical papers on artificial neural networks without positive or negative assessments in a practical sense. In contrast to the predominantly negative judgement towards hallucinations in NLP applications known today, the first usage of hallucinations within the research field of computer vision in the early 2000s were interpreted as a desired occurrence due to their role in computer vision processing such as resolution enhancement [Baker and Kanade, 2000; Ji et al., 2023, *p.* 2].

Intrinsic & Extrinsic Hallucinations

As noted in Maleki et al. [2024, *p.* 133], the phenomenon now classified as hallucination appeared several years later for the classifications of errors in text processing, applied for the tasks of machine translation [Koehn and Knowles, 2017] and data-to-document generation [Wiseman et al., 2017]. The initial identification within NLP, as mentioned above, was discussed in task areas such as text summarization, in which the term *hallucination* was stated during the discussion of factual inconsistency errors as both intrinsic and extrinsic hallucinations [Huang et al., 2021, *p.* 2]. Intrinsic hallucinations refer to the manipulation or adjustment of some text from the original source, while extrinsic hallucinations involve the generation of some facts or information that did not appear in the original text [Maynez et al., 2020, *p.* 1907].

Factual Hallucinations

A third categorization of hallucinations can be recognized here as well, meaning factual hallucinations, which can be both intrinsic and extrinsic, but differing from those two types in that they are factually correct, but do not appear in the source [Maynez et al., 2020, *p.* 1908]. The former two hallucination types (extrinsic and intrinsic) are thus generally seen as undesirable, since only the third type – factual hallucinations – is considered

to be factually correct. Notably, intrinsic hallucinations are by definition erroneous, since they contradict the information from the source text, while extrinsic hallucinations can be either factually correct or incorrect, but the identification depends on knowledge that goes beyond the source text and is thus hard to assess [Ji et al., 2023, *p.* 3].

Factuality vs. Faithfullness

One further classification should be mentioned here as well, which concerns modern LLMs and prompt-based interfaces in particular. Huang et al. [2025, *p.* 5ff.] define factuality and faithfulness as categories for hallucinations. The former comprises the already mentioned categories of intrinsic hallucinations – here factual contractions – and extrinsic – here factual fabrications – whereas the latter refers to hallucinations due to instruction, context or logical inconsistency [Huang et al., 2025, *p.* 6f.]. Without going into too much detail, these three latter categories revolve around the LLM not adhering to the desired output structure given in the prompt, falsely processing the provided context from the prompt or not following the command given a certain prompt [Huang et al., 2025, *p.* 7]. These distinctions regarding different hallucination types have been included here since they were noted during several experiments, which will be presented in Section 7.1.2.

Limitation of the Term Hallucinations

One last terminological consideration shall be included here since the term *hallucination*, in the medical or psychological sense, is potentially misleading when attributing human capabilities onto LLMs. The psychological definition of a hallucination involving an unreal perception that feels real shares similarities with the interpretation of an NLP error as unfaithful or factually incorrect generated text [Ji et al., 2023, *p.* 3]. However, the generation itself within the language model does not necessarily mean similar phenomena akin to human hallucinations since the generated texts themselves depend on the training data and probabilities, with which the texts are produced.

Interpretation for this Thesis

The existence of hallucinations therefore does not provide an argument that LLMs *think* in a similar way to humans, at least in this case. Hallucinations as a type of error therefore are not transferable to a human psychological process, as is noted by Maleki et al. [2024]:

> "1) The 'hallucination' metaphor in AI from this perspective is a misnomer, as AI lacks sensory perceptions, and errors arise from data and prompts rather than the absence of stimuli, and 2) this metaphor is highly stigmatizing, as it associates negative issues in AI with a specific issue in mental illness [...]." [Maleki et al., 2024, *p.* 133]

For this thesis, the definition of hallucinations as factuality or faithfulness errors are used, as stated above, but this terminological observation still remains noteworthy and future discussion regarding other terms, such as the alternative term *confabulation* should be considered [Sui et al., 2024].

Examples for Hallucinations

A concrete application scenario in the next chapter will be presented in Section 7.1.1, which refers to a text simplification use case, for which hallucinations were observed, and which will illustrate the problem. The experiments were conducted in 2022, before the release of ChatGPT. Prompt-based interaction was therefore not available and the prior workflow of fine-tuning a pre-trained language model was used, i.e., one of the usual approaches for state-of-the-art processes at the time. GPT-2, the predecessor of the underlying GPT-3 model of the initial ChatGPT release, was used as well as other approaches. Observations regarding hallucinations when explicitly applying LLMs were however also noted in various use cases, see Section 7.1.2. The tasks, experiments and results in regard to hallucinations will be presented there, see Section 7.1.

6.2 Lacking Metrics

The next problem concerns the facet of NLP with which quality and performance of approaches, models and techniques are assessed, meaning the evaluation metrics themselves. While a sound and comprehensive evaluation should also include a qualitative evaluation (sometimes also known as a manual evaluation), which can encompass surveys, user studies, interviews, and many other types,[85] the focus will be on quantitative evaluation for this part of the thesis.

Similarity to Hallucinations

To further narrow down the problems for this section, different types of evaluation procedures and metrics need to be defined first. Evaluation of NLP systems in general can be divided into intrinsic and extrinsic evaluations, similarly to the discussion of the same categories for types of hallucinations, see above in Section 6.1. As was the case for hallucinations, both intrinsic and extrinsic will be relevant for the discussion here.

Intrinsic & Extrinsic Evaluation

Intrinsic evaluation of a model or technique is task-neutral, i.e., independent of the concrete task for which the approach is aimed at later in a "real-world" use case [Eisenstein, 2018, *p.* 139]. This type of evaluation is usually carried out on the training, development and test splits from

[85] See, e.g., Fossey et al. [2002] or Moriarty [2011] for an overview from two other disciplines, psychiatry and social care research respectively, but nevertheless equally applicable for NLP.

a dataset, which are required to not share data instances in order to objectively assess the quality of the model in question [Jurafsky and Martin, 2024, *p.* 38f.]. Extrinsic evaluation incorporates the model or technique in an application or a downstream task, and through testing the application itself, an evaluation can be done in order to gauge the impact of the approach on the larger application in question [Eisenstein, 2018, *p.* 339ff.]. The performance of different techniques within the pipeline of a larger system is one parameter that can be tested in an extrinsic evaluation, e.g., the use of different NER approaches for hallucination detection (see below in Section 7.1.1 for text simplification purposes), which would in the end apply the NER techniques as purely "enabling technology" [Resnik and Lin, 2010, *p.* 273].

Overlap between Intrinsic & Extrinsic

The distinction between extrinsic and intrinsic evaluation is not necessarily as relevant for the remaining part of this chapter since the actual procedures with which they are carried out, e.g., the metrics that are applied, can overlap between the two. That means, metrics such as precision, recall and f1-score can be applied to both smaller intrinsic evaluations of a concrete technique, e.g., part of a search system such as dependency extraction approaches, and larger extrinsic evaluations of the complete search engine, in which the dependency extractor is just one part [Resnik and Lin, 2010, *p.* 272ff.].

Output as Evaluation Dimension

Two further dimensions shall be briefly described here to illustrate the importance of carefully considering the applicable metric in testing an NLP method, first the type of output and second the presence of references for evaluation purposes, e.g., reference summaries in text summarization datasets. The former represents the type of output produced by a text processing approach, which determines how the approach can be evaluated. While this fact seems obvious, the type of output can differ between either a single information, thus comprising a one-to-one link between input and output such as binary text classification, or the generation of multiple outputs for one input request, as in multi-class text classification or in search systems, all the way to the output itself as generated text, see text simplification.[86]

Classification Evaluation

The first category, relevant for example for text classification or PoS tagging, requires the assignment of a single piece of information to the input data. While the scope of input can vary widely, from a single word for PoS tagging to whole documents for text classification or sentiment analysis, the output is similarly brief, mostly one type of category. One

[86] Detailed considerations can be found, e.g., in Resnik and Lin [2010, *p.* 281ff.].

common metric in this regard, accuracy, is simply defined as the ratio of correctly assigned labels to the number of samples from the test set, while the inverse of accuracy, known as error rate, can also be found in research [Resnik and Lin, 2010, *p.* 282].

Search System Evaluation

The next paradigm is applicable for search systems or text alignment tasks, in which the single input data, usually a request in case of information retrieval systems, is matched with the documents in the test set and a ranking is produced. The ranking is usually assessed by calculating two metrics, precision and recall, or an approximation of combining the two as their harmonic mean, the f1-score or f-measure. Precision is the degree to which the search system is able to produce output that is relevant, while recall captures the degree of relevant documents that are retrieved. The aspect of relevancy is a critical assumption here since all documents are ranked as either relevant or not relevant. This binary classification is a simplification, ignoring aspects such as context, familiarity of the user with the topic at hand or hierarchical relationships between documents.[87]

NLG Evaluation

Lastly, the generation of another text as the output of an NLP method, e.g., in text summarization, translation or more generally Natural Language Generation (NLG) tasks, requires another adapted evaluation. Differences are apparent compared to the first two paradigms, since neither the agreement between input and classes – as representative for many other types of such evaluations – nor a relevancy ranking in respect to queries can be directly adapted to evaluate the generated text itself, which can be a summary, translation, simplification or other types of generated text based on the input [Resnik and Lin, 2010, *p.* 284f.].

Manual & Crowdsourced Evaluation

Manual evaluation via human/expert judges is another possible, albeit costly approach. Other related variants such as crowdsourced evaluations[88] via remotely hired or otherwise acquired workers are also a popular alternative to the manual search for experts, which however come with their own set of limitations or disadvantages [Jurafsky and Martin, 2024, *p.* 280f.]. The reliability, experience, expertise and honesty of crowdsourced participants differ between platforms and other related factors, see more details in Peer et al. [2022], in which the authors present a large-scale study in order to compare different platforms.

Evaluation Metrics for Text Overlap

The most common strategy for automatic evaluation for such NLP tasks is characterized by the degree of overlap between a unit or units of text from both the reference text and the generated text, which was produced

[87] See more regarding the influence of such parameters in Henrich [2008, *p.* 62ff.].

[88] See platforms such as Amazon Mechanical Turk (`https://www.mturk.com/`) or Prolific (`https://www.prolific.com/`) among others.

based on the original input data [Resnik and Lin, 2010, *p.* 284]. Examples for such evaluation metrics are BLEU, already mentioned for text simplification in Section 3.5, and ROUGE or METEOR for text summarization, described in Section 3.6, along with others. All noted metrics come with their own set of restrictions. BLEU and METEOR are both based on n-gram matches, therefore penalizing semantically fitting generated texts if they differ due to purely lexical differences that however match semantically [Wieting et al., 2019, *p.* 4344]. ROUGE, while also supporting multiple reference texts, is held back by the rarity of datasets actually containing multiple reference summaries and can thus only capture the fit towards one reference summary [Aumiller et al., 2023, *p.* 202]. Other evaluation paradigms are present in research, such as the output of a model containing structured information, e.g., within a parsing tool, or producing variable numbers on a scale such as ratio scales, e.g., in text similarity tasks [Resnik and Lin, 2010, *p.* 285ff.], but will not be included here.

Reference-Based Evaluation

The second dimension, after noting the different types of output as the first dimension above, concerns the presence of references as comparison. As has been described above regarding output types, text summarization tasks, among others, are evaluated based on the quality of the generated texts. These texts, however, need to either be compared to a reference or a set of references, often described as "gold standard", or be judged based on other criteria. The de facto default evaluation procedures in machine translation, text summarization, text simplification and other NLG tasks are conducted using comparisons to reference texts, as mentioned above. The production of such reference texts by domain experts is time-consuming, costly and thus seldom feasible. If such reference texts, or even a set of references, exist, it is still not guaranteed that the generated text is not produced in such a way that a significant lexical overlap is present, which is due to the lack of coverage between the reference texts and the generated text that is evaluated [Asano et al., 2017, *p.* 343].

Limitations of References

The problem here is the reference texts themselves. Firstly, not all datasets include reference texts at all and if they do exist, coverage, as mentioned just above, is not ensured. Furthermore, reference-based metrics measure the overlap between n-grams, or other units of text depending on the concrete metrics, which are helpful in identifying lexical matches between texts. However, and crucially, aspects such as coherence, fluency, grammaticality and meaning preservation are not captured by lexical overlap and are at most a byproduct of lexical overlap, not measured exclusively because of purely lexical matches [Asano et al., 2017, *p.* 344ff.].

Coherence

Coherence, as just one example, is a complex task, dealing with elements such as discourse coherence – the structure of sentences and their relationships between each other captured as one connected group of sentences – or local and global coherence [Jurafsky and Martin, 2024, *p.* 531ff.]. While local coherence, also known as lexical cohesion and referring to the semantic matches or relatedness between consecutive sentences and their entities in the text, is a phenomenon that might be captured by lexical overlap as described above, global coherence goes far beyond the capabilities of reference-based overlap metrics [Barzilay and Lapata, 2008]. Global coherence captures the quality of a text, which can be attributed through sequential and aligned plotlines in fictional texts or a conventional table of contents and their corresponding sections. The evaluation regarding global coherence is thus even more complex and researched in several studies [Lapata et al., 2005], mostly for narrative texts [Purdy et al., 2018; Zhai et al., 2019].

Lack of References

This discussion on coherence shall suffice here as a multifaceted aspect regarding the evaluation of texts but should be able to showcase the complexity in measuring the quality of a text for different NLP tasks. The dependence of quantitatively evaluating NLP methods based on references has now been addressed. This evaluation paradigm, i.e., reference-based evaluation, is, however, not applicable for all use cases. Another problem entirely persists in such scenarios, in which no gold standard or references exist at all. The distinction between supervised and unsupervised methods within machine learning further categorizes this dichotomy, e.g., presented in Sindhu Meena and Suriya [2019] through classification tasks such as SVM or decision trees for supervised approaches or k-means and other clustering techniques for unsupervised approaches.

Evaluation of Unsupervised Methods

Evaluation, as expected, is a complex problem for such scenarios, when no reference data for, e.g., text summarization or text simplification purposes is available. The evaluation of unsupervised systems can still rely on the presence of annotated test data as a small substitute for whole datasets including reference texts, or on the availability of "pseudo-annotated" test items [Resnik and Lin, 2010, *p.* 294]. If neither are available, annotated test data in general or pseudo-annotated test items, other measures are needed. A selection of evaluation procedures has been surveyed by Pang and Gimpel [2019] in the use case of textual transfer, comprising the use of pre-trained classifiers that measure the accuracy of the produced text, which is however heavily reliant on the training data and prone to overfitting. Therefore, other metrics need to be specifically adapted to the respective use case, e.g., by metrics such as post-transfer classification

accuracy in case of textual transfer, or semantic similarity or fluency for more generic use cases [Pang and Gimpel, 2019, *p.* 140]. More concrete examples will be presented below in Section 7.2.2.

References to Concrete Examples

Similarly to the challenges of hallucinations, text summarization and text simplification will be presented as showcases for this problem, see below in Section 7.2, in addition to keyphrase prediction in Section 7.2.1 as a separate use case with limitations regarding its evaluation.

As has been noted several times in this thesis, the application of techniques and models on languages other than English presents challenges due to lacking or smaller datasets, required adaptations towards the respective language and other aspects. This challenge will be discussed in the next section.

6.3 Availability of Data and Models Primarily for English

The dominance of research in NLP regarding models, techniques and datasets for the English language has been recognized, studied and interpreted, e.g., in Bender [2009] or Ruder et al. [2022], even with sometimes catchy titles such as "Should We Ban English NLP for a Year?" [Søgaard, 2022]. While this title is probably deliberately exaggerating, the rationale still deserves some further discussion in this chapter, through a problem definition and afterwards observations from select use cases, later in Section 7.3.

mLongT5 & mC4

To illustrate the problem with just one practical comparison, the pretrained language model mLongT5 [Uthus et al., 2023], as applied for two summarization tasks, see Section 7.1.2 regarding hallucinations and Section 7.3 for the presently discussed challenge, is capable of processing input and producing output from dozens of different languages. The dataset used in training the mLongT5 model is an updated version of mC4[89], with widely varying corpus sizes for each included language. The data for English has roughly four times as many characters as the next largest language – Russian – and over seven times as many characters as French, German or Italian [Chung et al., 2023, *p.* 19].[90] While size is certainly not the only important factor in processing data, it is nevertheless a metric

[89] mC4 being a multilingual version [Chung et al., 2023] of the Colossal Clean Crawled Corpus or C4, a derivative of the popular Common Crawl, see Raffel et al. [2020].

[90] These four languages, i.e., English, French, German and Italian, comprise the analyzed languages in the multilingual summarization task from Dal Cin [2025].

characterizing the variety and magnitude, for which English data is available.

Domain Corpora

Since mC4 is based on pages crawled from the web, non-English languages are represented at a higher rate in mC4 when compared to other more specialized corpora, which are often only available in English [Aumiller et al., 2022, *p.* 7626ff.]. Apart from the existence of domain corpora for other languages, data quality is another issue for corpora in general, but also specifically for datasets in a multilingual setting or for corpora in languages other than English [Artetxe et al., 2022; Held et al., 2023].

Language Technologies

Furthermore, the availability of tools, techniques and even best practices is overwhelmingly constructed for English or at least geared towards English texts, which was shown, e.g., in a study detailing the state of language technology support for different domains such as speech processing, machine translation and text analysis for 30 different European languages [Ananiadou et al., 2012]. Ananiadou et al. [2012, *p.* 29ff.] measured the quality of available tools for the respective domains and deemed the English results to have "good support" across all domains, whereas all other analyzed languages received worse scores across all domains. While this study was conducted in 2012 and therefore before the rise of LLMs, the results were confirmed in other more recent studies, e.g., in Zhang et al. [2023b].

Biases towards English

The dominance of the English language is apparent not only through datasets and language technologies in general, but so-called protocol biases towards English, see Ruder et al. [2022, *p.* 2345]. The prototypical scenario for NLP applications is geared towards the English language and even in a multilingual setting, English is used as the source language per default for zero-shot cross-lingual transfer [Hu et al., 2020], while other languages might be more applicable depending on the non-English languages [Lin et al., 2019; Ponti et al., 2018]. Ponti et al. [2018] note the impact of verb usage, particularly in determining the tense of the sentence. For example, the future tense in Latin is constructed by verb inflection, whereas English adds an auxiliary verb [Ponti et al., 2018, *p.* 1532]. Such considerations can have an impact when aligning two languages and can exclude English from a primary recommendation. Furthermore, machine translation tasks heavily skew towards English through the availability of language pairs dominated by English as one of the two concerned languages, therefore decreasing the comparability with language pairs not including English [Fan et al., 2021].

Apart from studying protocol biases towards English and other factors, such as organizational biases through the focus on conferences and peer-

Biases in NLP Teaching & NLP Books

review processes, Ruder et al. [2022, *p.* 2346ff.] also study how NLP is taught and how languages are selected in standard textbooks. Regarding the first aspect, most surveyed teachers answered that they were using English for initial NLP experiment for their students, even when the class consisted of students whose native language was not English. As for the second aspect, standard textbooks, including Eisenstein [2018] and Jurafsky and Martin [2024], both quoted throughout this thesis, were analyzed. The examples and experiments therein mainly discuss NLP applications that are once again dealing with English language processing tasks.

Observations for LLMs

The previous studies and observations, while still relevant for the discussion in this thesis, were mostly published before the rise of chat-based LLMs. One study, mentioned briefly above, from Zhang et al. [2023b], analyzed the performance of GPT-3.5 on a series of multilingual tasks including knowledge access, reasoning and translation. The results of this study confirm the outcomes of prior investigations, even for modern LLMs, i.e., worse performance for languages other than English. The authors discuss reasons for this behavior, and propose, as was noted earlier in this chapter, that lacking datasets for multilingual or even parallel data with non-English languages is partly to blame for the poor performance [Zhang et al., 2023b, *p.* 7922]. The dominance of monolingual text in the training data for LLMs, such as GPT-3.5, further leads to the language model learning a representation of knowledge biased towards English texts. To illustrate the latter point, the authors present experiments, wherein the prompts are written in non-English languages, but the results appear to rely on information which is provided through English language properties such as semantic interpretation of synonyms that do not appear in the language with which the prompt was originally written [Zhang et al., 2023b, *p.* 7921f.].

Recent LLM Analysis

In 2025, even more recent LLMs were surveyed by Qin et al. [2025]. The authors present a comprehensive taxonomy based on different model architectures and alignment strategies as well as reviewing the used training datasets and other characteristics of various models. The conclusions are similar to the results presented above in Zhang et al. [2023b], meaning that modern LLMs including GPT-4 and LLaMA for English tasks clearly surpass the performance of the same models for other languages [Qin et al., 2025, *p.* 7] on the employed XNLI benchmark, which is intended to evaluate cross-lingual sentence understanding [Conneau et al., 2018]. Another only recently conducted survey focused on the challenges and performance of LLMs on low-resource language contexts, see Pava et al. [2025]. In their study, the authors analyzed the distinction

between languages from the Global North, including English and many European languages, and the Global South as well as the relevant cultural contexts of non-English languages, which confirm the bias towards English across many dimensions [Pava et al., 2025, *p.* 10ff.].

Recollection

The paper introduced at the beginning of this section with the title "Should We Ban English NLP for a Year?" [Søgaard, 2022] can be considered as a position paper. Within that publication, the author suggests some, in his own opinion, admittedly radical proposals to initiate NLP research for non-English datasets and use cases. Measures from governing bodies such as the Association for Computational Linguistics (ACL) issuing a quota on English models or biased end user groups on the one hand or by issuing higher conferences fees for submissions focused on applications on English on the other hand are strict measures but would "move the needle in the right direction" [Søgaard, 2022, *p.* 5257]. While this thesis does not agree with these radical measures to the same extent, the conscious knowledge of the bias towards English within NLP research, especially when conceptualizing applications for other languages, is important in so far, that expectations are adjusted accordingly.

Now that the limitations and issues due to the dominance of English in past and current NLP research have been discussed, a closer look will be presented in the next section of some varied factors that are also negatively impacting text processing approaches.

6.4 Other Challenges

The three main challenges already addressed in this chapter have all been captured in several applications and use cases by observing them directly or supervising approaches in which they have occurred. All of them will be showcased in Chapter 7. Before taking a more hands-on perspective regarding the challenges present in current NLP, some more factors will be discussed in a more concise manner, since they all were observed at least in a limited manner in experiments and are also applicable for the overarching theme of this thesis, i.e., the relevance of traditional models in contrast to modern LLMs.

6.4.1 Reproducibility

Replication Crisis

The origins of the challenge described in this subsection can be traced back to other research fields, first noticed in medicine in an essay by

Ioannidis [2005] with the rather dramatic title "Why most published research findings are false". Nevertheless, the problems mentioned in that paper, such as sample size, biases and comparable relationships within the tested approaches among others were the first symptoms of the event known as the *replication crisis* [Wiggins and Christopherson, 2019].[91] The impact of the initial paper, as well as follow-up studies and reflections within, contributed to the increase of procedures such as meta-analyses and the emergence of the research field of metascience.

Overarching Reproducibility Studies in NLP

Reproducibility, when focusing on NLP, has been studied in comprehensive and wide-ranging ways, e.g., in Belz et al. [2021]. In their SLR, the authors classified reproducibility in multiple ways, such as experimental settings differing between reproductions under the same conditions or under varied conditions [Belz et al., 2021, *p.* 383f.]. Other distinctions are drawn between multi-test, i.e., conducting multiple studies of the same type, and multi-lab, wherein more than one research team is performing the same study.

Reproducibility for this Thesis

The reproducibility of entire studies has been the focus of the influential paper by Ioannidis [2005] and the replication crisis in general. While the importance of replicable studies in this manner is obvious, a more narrow or small-scale perspective seems as natural – and is certainly also studied within the paradigm of the replication crisis on a more micro-level as well – in particular for the discussion of use cases presented in this thesis. Abstractive and generative approaches have prompted the analysis of reproducibility for NLP tasks, e.g., due to increased difficulty in evaluating such methods and techniques that may produce different results for each execution. Measures in order to enable reproducibility, even for generative approaches, can be conceptualized as the adaptation of parameters such as temperature[92] for LLMs or by using multiple execution runs of an LLM and computing the confidence interval based on these runs [Lin et al., 2024]. The use of temperature as configurable parameter would result in sampling strategies that produce text, which can be graded on a scale of being coherent and reproducible on one end of the scale, to being

[91] Otherwise also referred to as the reproducibility or replicability crisis. The distinction between the terms for this thesis has already been discussed in Section 4.2, which was first noticed for the field of psychology, and emerged therein but was not exclusively limited to that research area. Due to the ubiquity of the term, *replication crisis* will be used for the event as mentioned in this chapter, while *reproducibility* will be used for the phenomenon discussed in other parts of this thesis.

[92] The parameter temperature has been known to regulate randomness in sampling processes for stochastic models since the 1980s, first introduced in Ackley et al. [1985], and still in use today to measure randomness, or nowadays also a degree of creativity [Peeperkorn et al., 2024].

creative, structurally more flexible and less reproducible on the other end [Peeperkorn et al., 2024]. However, the applied sampling strategies need to support the use of such a temperature parameter, which is not the case for many prominent representatives of sampling strategies, see Holtzman et al. [2020, *p.* 3ff.]. Also, LLMs, when used in a chat-based interface, often do not present technical parameters to the user, which prevents the use of similar configurations, while the use of prompt-based interaction via APIs is usually possible in some way.[93]

Traditional methods, as defined in this thesis, are in contrast trending towards being easily reproducible, ranging from extractive summarization approaches, rule-based extraction techniques, or various classification methods.[94] The efforts towards reproducing the results of experiments are a worthwhile cause, for many different reasons such as comprehensibility and transparency of the applied approaches. After ensuring or at least endeavoring towards reproducible results, the interpretation of the generated texts presents the next challenge, which is captured in the following.

6.4.2 Explainability

Terminology

The emergence of LLMs has renewed the interest in explainable artificial intelligence, which has been a smaller part of research in fields such as expert systems since the 1980s [Miller, 2019, *p.* 1]. The explosion of interest in this field of study is worth a comprehensive discussion, which cannot be achieved within the scope of this thesis, but can be glimpsed via surveys, e.g., in Madsen et al. [2022] or Zini and Awad [2022]. In discussing explainability, this thesis will refer to the terms *explainability* and *interpretability* as equal concepts, with a very brief definition of interpretability as "the degree to which an observer can understand the cause of a decision" [Miller, 2019, *p.* 8].

Role of Explanations

In general, modern (large) language models can be categorized as black-box models [Madsen et al., 2022, *p.* 1]. As mentioned earlier, the risks of using LLMs, see Section 4.2, can include biases and unexpected beha-

[93] See, e.g., `https://cloud.google.com/vertex-ai/generative-ai/docs/learn/prompts/adjust-parameter-values/` for Google's Gemini or `https://platform.openai.com/docs/api-reference/backward-compatibility/` for OpenAI's ChatGPT. However, the latter link already refers to backwards compatibility measures, i.e., modern LLMs are not shipping with such parameters as default or desired options.

[94] With exceptions such as random forest algorithms, for which the randomness is an explicit feature of the procedure, and which will also be mentioned below in Section 6.4.2 regarding explainability.

vior as well as producing toxic content [Patil and Gudivada, 2024, *p.* 30f.]. Safely testing, evaluating and later also actually using LLMs in critical use cases with sensitive data still presents a challenge and is far from a solved issue. Since the black-box nature of modern language models is a prevalent factor of modern NLP methods[95], the contrasting white-box characteristics of other approaches deserve a short explanation as well.

Intrinsic vs. Post-Hoc

White-box models are intrinsic methods [Madsen et al., 2022, *p.* 5], i.e., they are interpretable through their model characteristics. Representatives of such models are decision trees and rule-based approaches, among others [Du et al., 2019, *p.* 68f.]. In contrast, post-hoc methods attempt to provide explanations for decisions or generations retroactively, i.e., after the model in question has been trained and also already been applied [Madsen et al., 2022, *p.* 4]. When comparing the two categories, different characteristics become apparent. While post-hoc methods usually apply to methods that are more generalizable through their training procedures, they are also harder to accurately explain, whereas intrinsic methods are by design explainable, but much more fine-grained in terms of their intended use case as well as depending on more knowledge about the desired task [Du et al., 2019, *p.* 70]. Other categories such as the degree of abstraction, referring to local explanations, i.e., explaining a local singular decision, in contrast to global explanations, i.e., explaining a model in total by understanding its structure and variables, are in use in this research field, but will not be further discussed in this thesis.[96]

Explainability for Textual Data

When focusing back on explainability within NLP techniques, a difficulty in visualizing or accurately representing decisions based on the textual representations through values and vectors becomes apparent once more. Text as a medium is often unstructured, which is why complex architectures using embeddings spaces have been initialized in order to capture semantic relationships, among other goals, see above in Section 2.3. Explanations regarding embeddings or single embedding dimensions, however, are not useful since they are "opaque representations" [Zini and Awad, 2022, *p.* 3], which do not have practical justifications for their resulting generations and also lack a means to easily visualize them, especially

[95] A direct link between internal structures of black-box neural network architectures and specific features of the generated text is an ongoing research problem, e.g., described in the survey by Zini and Awad [2022, *p.* 5ff.].

[96] See for more details, e.g., the discussion in general about machine learning approaches in Du et al. [2019] or in particular for post-hoc methods in Madsen et al. [2022].

in contrast to image processing.[97] As a result, the identification of units of text that led to certain decisions or generations, depending on the use case, is hard to comprehend and interpret.

Explainable Random Forests

Even some traditional models, according to the definition used in this thesis, such as SVMs, are not easily explained. In attempting an early classification above in Section 2.5, random forest classifiers and SVMs were noted as two representatives of traditional models that are not directly explainable. The former relies on the random selection of features, which are used in order to evaluate between classes at each tree node [Breiman, 2001]. Mechanisms to offer some manner of explainability through summary reports, proposed by Petkovic et al. [2018], are intended to provide information on the importance of the selected features. The interpretation of results, in case of random forest classifiers, is furthermore aided by the ability to visualize the results akin to decision trees in a sequential nature, as presented by Neto and Paulovich [2020].

Explainable SVMs

The latter method, SVMs, can be characterized as a black box, in which the hyperplanes separating different classes – when used in a typical classification scenario – are not easily explainable or visualizable. Still, even for SVMs, research has found the ability to extract rules from within the hyperplane, which can be used akin to decision trees, see Núñez et al. [2002]. Most other traditional models, in contrast to the two discussed methods of random forest and SVMs, are directly explainable, such as extractive summarization approaches, decision trees for information extraction methods or linear n-gram classification techniques.

A further dimension is naturally efficiency, at least since the inception of neural networks trained on large datasets, see Section 2.3 for an overview of this development. Efficiency can be interpreted over time and energy dimensions among other aspects and will be defined in the next subsection.

6.4.3 Efficiency

Terminological Overview

The definition of efficiency depends largely on the context it is used in. For information retrieval purposes the main criteria for efficiency is the processing time, i.e., retrieving the desired search results as quickly as possible for the user and their requests [Croft et al., 2010, *p.* 13f.]. More generally, in commercial terminology, efficiency measures the relation-

[97] See, e.g., the study in Lapuschkin et al. [2016, *p.* 2917] on image classification, which detected certain shapes or the presence of copyright tags on images as important features for certain models. A direct analogy to textual data is hardly possible.

ship between cost and resources as input and produced goods as output, through energy or budgetary and temporal means.[98] The research for efficiency as a parameter for different methods is a long-standing field of study, with one of the more famous concepts being Pareto efficiency, invented at the end of the 19th century by Pareto, see, e.g., Luc [2008]. This concept describes Pareto-optimal approaches, for which no other method can achieve a better or equal performance without also requiring higher resource consumption [Treviso et al., 2023, *p.* 835f.]. While the analysis of more complex efficiency studies is certainly worthwhile, this thesis will focus instead on more straightforward and smaller-scale discussions on efficiency with a direct focus on NLP tasks.

Efficiency for NLP Tasks

The factor time is certainly also applicable for NLP research. Time is, however, similar to other facets of efficiency, not a clearly defined factor. The required time in order to train a model can be a relevant dimension in early planning stages, while the amount of time a single execution of a model requires in an actual use case is equally as relevant. Similarly, energy can be characterized in the same manner, on the one hand the energy expended in training a model and on the other hand, the energy requirement of a single execution of a model for one user request or the processing of one dataset. In a more expansive interpretation, hardware requirements can also be seen as one aspect of efficiency. Modern transformer-based language models and in particular current LLMs necessitate recent GPUs with significant virtual memory in order to process user prompts and achieve acceptable generation speed.[99] Therefore, one dimension of efficiency for NLP can be seen as the ability to perform the necessary computations, either for training, execution or both operations, on a less capable machine.

Measuring Efficiency

The measurements regarding the temporal dimension seem straightforward, at least at first glance. Measuring the time needed to process a request, preferable within a setup that allows repeated iterations within the same parameter settings, seems simple enough. While this setup is certainly achievable for smaller experiments, the scale of modern LLMs,[100] their proprietary nature and less than stellar documentation and transparency of models by big LLM providers, decrease comprehensibility. In terms of energy consumption, brief remarks regarding missing standards and best practices have already been mentioned in Section 4.2. The use of

[98] See, e.g., `https://www.merriam-webster.com/dictionary/efficiency/`.

[99] Usually measured as Tokens Per Second (TPS).

[100] With training runs measuring in weeks and months, on large server farms using hundreds or even thousands of GPUs, see above in Section 2.4.

hardware power meters would provide accurate readings for the total energy consumption of a system. However, the required hardware serves as an additional limitation when measuring the energy consumption of an approach. In turn, software-based methods have been studied and shown to strongly correlate to hardware power meters [Jay et al., 2023], therefore simplifying the measuring process. The final dimension of efficiency discussed more closely in this thesis, the necessary hardware requirements, is more straightforward. The ability to process a request or dataset with limited or even consumer hardware in comparison to more capable workstations with dedicated GPUs serves as a simple distinction and will be discussed more detailed in the appropriate section, e.g., in Section 7.4.3 and Section 8.1.

Green NLP

Further dimensions for efficiency are directly tied to the research field of Green NLP, similar to energy consumption, and include the expended amount of water used in training the models, or rather in cooling the systems applied in training the models, and furthermore the carbon footprint of such models. Two observations regarding water consumption include the amount of water required in training GPT-3 being measured as 5.4 million liters or, projected towards the future, the demands of AI systems globally exceeding the total water requirements of entire countries [Li et al., 2025].[101] The continued application of LLMs coincides with additional water usage, which in combination with increasing water scarcity due to the climate crisis is troublesome. Similarly, the carbon impact of large-scale server farms is obfuscated by the lack of transparency in reports by AI providers and deserves a closer investigation in the future [Patterson et al., 2021]. While the measurement of carbon emissions depends on location, hardware setup and other factors, real-time access for carbon intensity is one area of promising research, see, e.g., Treviso et al. [2023, *p.* 836].

GPU Lifecycle

An even more recently conducted study by Falk et al. [2025] analyzes the long-term impact of AI and LLM models through their hardware requirements in addition to surveying training as well as inference phases. The manufacturing stage of GPUs is one main topic of the study and related dimensions such as the expenditure of rare minerals and metals, carbon emissions during production and more are evaluated. A particular focus is placed on the complete lifecycle of the hardware, referred to as "cradle-to-grave", similar to the lifecycle of a product mentioned above for product development as BOL, MOL and EOL from Section 5.1. The study

[101] By 2027, the projected water demand of global AI systems is expected to exceed the annual water withdrawal of countries such as Denmark by several times [Li et al., 2025, *p.* 54].

encompasses a multi criteria lifecycle assessment to measure the environmental impact of hardware across the whole lifespan of a product [Falk et al., 2025, *p.* 4ff.], which in total represents the influence of hardware on NLP methods for LLM models.

Potential of Traditional Models

Research question 3, while not being answered in this chapter, can be introduced through the observations regarding different challenges of modern NLP throughout this chapter. The two distinct manners of approaching traditional methods mentioned at the start of this chapter, see the beginning of Chapter 6, have both been observed here. First, the absence of limitations when applying traditional methods was noted for hallucinations in Section 6.1, since hallucinations cannot appear when applying traditional methods as defined for this thesis. These observations will be further discussed below in respect to concrete applications in Section 7.1. Second, challenges for text processing methods that appear in a more limited manner can be noted for all facets listed in the remaining challenges in Section 6.4, such as efficiency, or at least for many traditional techniques, also for reproducibility and explainability.

The theoretical background presented in this chapter is a necessary foundation in discussing challenges in current NLP research and applications. Nevertheless, observations through experiments and evaluations on concrete use cases will illustrate and reinforce these challenges, as will be shown next, in Chapter 7.

7 Selected Use Cases

Structurally, the current chapter is aligned with the preceding Chapter 6. All challenges presented there have been noticed and studied in various use cases, which will be presented in the current chapter. Observations of common problems in specific NLP applications will be discussed that have been presented more abstractly earlier, beginning with hallucinations in Section 7.1 as counterpart to Section 6.1. The other sections from Section 6.2 to Section 6.4 correspond to the respective use cases in this chapter from Section 7.2 to Section 7.4. Research question (2) – concerning the analysis of traditional models and if they are still competitive with modern systems – will therefore be of relevance for the different use cases presented in this chapter, in particular when the results of traditional models and modern methods are compared.[102]

7.1 Hallucinations in Concrete Scenarios

Appearance of Hallucinations

The appearance of hallucinations is one common problem across generative models and is not limited to one medium such as text but can occur in other mediums such as image or video [Bai et al., 2024]. A distinction can be made concerning abstractive tasks, which are commonly used for text summarization approaches that are also classified as generative methods, in contrast to extractive summarization approaches. Terminologically, abstractive techniques are nearly exclusively used for text summarization tasks, whereas generative approaches refer to all methods that model the data distribution and can sample from it. Abstractive summarization will therefore be described separately in Section 7.1.2, focusing on summaries generated by LLMs. Beforehand, another generative use case will be discussed in Section 7.1.1 for text simplification.

[102] Some of the use cases and experiments presented in this chapter have been realized as part of student theses, which were supervised by the author of this thesis. If observations from such theses are taken into consideration as part of this chapter, a note will be placed at the beginning of the respective section or subsection. Some remarks regarding these theses were already collected in Jegan [2024].

7.1.1 Text Simplification

Some parts of this subsection have been inspired by a supervised thesis, in particular the experimental results, see Fruth [2022].

Hallucinations through Simplification

Text simplification as an application scenario has already been introduced earlier in this thesis in Section 3.5. Early text simplification approaches applied sentence splitting operations in order to reduce complexity of a text or used lexical substitutions by retrieving less complex synonyms for their more complex counterparts from the original text. The former procedure cannot result in hallucinations as no entirely new text is created, while the lexical substitution procedure can merely produce vagueness and ambiguity through the replacements. In a stricter sense, neither can produce hallucinations. The introduction of generative neural network models, which can produce text that can differ substantially from the input, resulted in an increased likelihood of hallucinations within the text simplification domain.

Thesis: Text Simplification

> The NLP task of text simplification is studied as an unsupervised reinforcement learning approach, see Fruth [2022]. This scenario encompasses the simplification of paragraph-level German text for two datasets, comprised of news and Wikipedia texts. Two models are trained based on a reinforcement learning technique, with a hybrid pivot model serving as a baseline. Evaluation was conducted quantitatively using SARI and FRE among others and qualitatively using various simplification parameters.

Unsupervised German Simplification

One application of text simplification for a concrete task will be presented here, namely an unsupervised text simplification approach intended for paragraph-level texts in German [Fruth, 2022], which was also briefly discussed in Jegan [2024]. The pipeline was modeled after Laban et al. [2021], a system based on reinforcement learning techniques and designed for English. Their approach focused on multi-paragraph texts and aimed at optimizing the parameters fluency, salience, i.e., keeping the content of the original text, and simplicity [Laban et al., 2021]. However, adaptations for German were necessary, in particular the selection of a German pre-trained language model instead of the base model used in the original approach, both based on the GPT-2 architecture. Furthermore, the original method was constructed with English grammar and syntax in mind, requiring adjustments to the optimization procedure, e.g., by adding a measure to prevent article repetitions, adding a fourth optimiz-

ation procedure for meaning preservation and addressing hallucinations by a detection mechanism. The latter mechanism will be described in detail in the following, since it relates to this section. Other customizations in adapting the English approach to German will be presented in more detail in Section 7.3.2 with a stronger focus on language transfer.

Hallucination Detection

The detection of hallucinations is present in the original simplification approach as a score called *inaccuracy guardrail*. It employs NER to detect named entities that were added to the output during the simplification procedure and that were not present in the input [Laban et al., 2021, *p*. 6369]. While scoring was a simple procedure, meaning it was binary – either hallucinated named entities were present or not – the adaptation to German was developed further to go one step beyond the simple matching process introduced by Laban et al. [2021]. The newly introduced approach included the matching process of the named entities in the simplified text not by basic string matching, but by calculating the BERTScore between the locations, persons and organizations from the simplified text and the input text [Fruth, 2022, *p*. 28f.]. Thus, similarity scores are calculated and only if the similarity values between the named entities and tokens from the input fall below a set threshold, a hallucination is detected. This more advanced measure not only encompasses lexical substitutions, which would not be recognized through the original mechanism for the English simplification approach but also encloses particularities of the German language such as wording variations due to various grammatical cases.

Undetected Hallucinations

Still, limitations with this approach persist due to the available NER models that were applied for German and their exclusion of dates and numeric values, in contrast to the English models used in Laban et al. [2021]. Thus, hallucinations for temporal information and numerical values could not be detected, which, as the qualitative evaluation in this use case showed, were occurring in the simplified texts [Fruth, 2022, *p*. 55f.]. Furthermore, intrinsic hallucinations with purely syntactical changes or hallucinated words not belonging to named entities, such as added noun phrases, could not be detected through the mechanism and are still a problem for this approach.

Detection Limitation

Nevertheless, two types of strategies have now been presented to combat hallucinations, both by employing similarity calculations on named entities, either through simple string matching or by measuring the BERTScore. Briefly, the pipeline of this approach will be described to display how the hallucination prevention was already a focus in the architecture of the model. The simplification model is realized as a rein-

forcement learning task, in which SCST processes are applied to optimize pre-defined rewards [Rennie et al., 2017]. Laban et al. [2021, *p.* 6370] further developed SCST to include the generation of several candidates to improve the average reward, thus called k-SCST. The hallucination detection mechanism, in addition to other rewards such as simplicity, fluency and meaning preservation as well as further guardrails such as size constraints were used to optimize the texts produced by the generator. Since the optimization is tethered to the already mentioned diverse scores, a balance across all rewards needs to be targeted. A singular focus on simply eliminating hallucinations and thereby penalizing other scores is therefore not feasible, which is why hallucinations are still generated by the final model.

The simplification application used (larger) language models, that were, however, conceptualized before the rise of chat-based LLMs. Thus, a closer look at hallucinations within LLMs is necessary and will be given next.

7.1.2 Hallucinations in LLMs

Some parts of this subsection have been inspired by supervised theses, in particular the experimental results, see Dal Cin [2025] and Möglich [2025].

Hallucinations are one of the most prominent problems in research on chat-based LLMs due to the open-ended text generation architecture, in which LLM-based methods are used to handle NLP tasks [Huang et al., 2025, *p.* 2]. Different categories of hallucinations within LLMs have already been briefly mentioned earlier in Section 6.1, but the distinction between hallucinations in terms of factuality and faithfulness, especially for prompt-based LLMs, will be noted here once again due to the relevance for the observations in this section. The former category, i.e., factuality, referring to intrinsic or extrinsic hallucinations, will be discussed first.

Origin of Hallucinations

Factuality hallucinations comprise factual inconsistencies, i.e., a noticeable difference between the information provided in the generated text and real-world facts [Huang et al., 2025, *p.* 2f.]. The origin of hallucinations is still part of an ongoing debate and continued research aiming at preventing hallucinations from appearing. One explanation is due to gaps in knowledge within the model's training data and lacking context, which in turn leads the model to fill those gaps with hallucinated facts [Hadi et al., 2023, *p.* 28]. Other related issues in the training data of LLMs con-

cern biases and misinformation that can surface in the generated texts [Huang et al., 2025, *p*. 19ff.]. Further exacerbating this problem is the inability of training algorithms to guarantee that the generated texts are required to be accurate and do not contain hallucinations [Jurafsky and Martin, 2024, *p*. 219]. Such guardrails or prevention mechanisms, which fully prohibit hallucinations from appearing, simply do not exist (yet).

Calibrated Systems

The occurrence of hallucinations in LLMs is even more problematic due to the high confidence they are generated with. The relevant concepts in this area are confidence, probability regarding predictions and calibration for uncertainty [Guo et al., 2021]. Even when prompted for confidence scores, studies have shown that language models are overly confident, even in cases when the generated responses are not deserving of such scores due to apparent knowledge gaps [Zhou et al., 2024a]. Calibrated systems, in contrast, entail more trustworthy characteristics. On the one hand, calibrated systems achieve a high correlation between the confidence scores and factual responses produced by the system and on the other hand present lower confidence scores, if the generations are not based on some criteria or the generation is situated within some knowledge gap [Jurafsky and Martin, 2024, *p*. 290]. Due to the open-ended nature of prompt-based LLMs, calibration is naturally difficult and in non-specialized use cases not desired, especially due to the required flexibility of such systems as a key feature.

LLM Benchmarks

Beyond detecting hallucinations, evaluating systems in this regard is another complex issue, see the discussion for text simplification in Section 7.1.1, and the often unclear data basis on which the LLM is trained on. Besides publicly available datasets such as Common Crawl, details regarding the data, its origin and other information regarding preprocessing are not readily available [Jaech et al., 2024, *p*. 1f.].[103] While the evaluation for previously presented NLP tasks, such as text simplification in Section 7.1.1, are conceivable through comparison between reference texts and the generated output, benchmarks regarding general prompt-based LLMs are more complex. Evaluation metrics to measure hallucinations are grouped by Huang et al. [2025, *p*. 17ff.] into two classes: Hallucination evaluation benchmarks and hallucination detection benchmarks. The former applies question answering tasks and analyzes the responses based on accuracy, human judgments or other language models. The

[103] Other issues in this context include copyright, see, e.g., `https://www.nytimes.com/2023/12/27/business/media/new-york-times-open-ai-microsoft-lawsuit.html/`, and personal data used in the training datasets, which shall only be noted here as an ongoing struggle in LLM research and use, see also Yao et al. [2024].

latter applies task-specific evaluation scenarios, more similar to the one presented above for text simplification, for approaches in text summarization or open-domain question answering [Huang et al., 2025, *p.* 19]. Both types of benchmarks are susceptible to LLMs benefiting from adding the test data to their training data, which is known as *benchmark data contamination*, and which has been observed in several studies [Deng et al., 2024; Xu et al., 2024a]. The benchmark limitation is certainly not unique to hallucination benchmarking, but unfortunately a common problem for LLM benchmark suites across different use cases.[104] The problem of evaluating LLMs, especially in terms of hallucination detection, is still part of active research and discourse [Chakraborty et al., 2025].

Filtering Mechanisms

Despite the yet unsolved problems of preventing hallucinations, mitigation strategies have emerged, either inspired by RAG-based approaches or by using knowledge recall, i.e., by providing more context-specific information such as keys or related sentences [Zheng et al., 2024, *p.* 5f.]. Further prevention procedures aim at earlier parts of the LLM lifecycle. One of these measures involves data filtering techniques intended to prevent biases and false information from being introduced into the training data. These measures attempt to prevent hallucinations from being produced by providing a high data quality of the training data instead of processing the generated text in some way after the fact [Huang et al., 2025, *p.* 20].

Other Prevention Procedures

RAG-based approaches are located at the other end of the pipeline. They incorporate additional information, in the form of contextual clues, into the user's prompt. The additional context specifies the request and thereby helps to prevent hallucinations from appearing [Wang et al., 2024]. In-between filtering and RAG-based techniques are procedures based on model editing that apply updates to parameters and activations for many layers of the neural network model. However, model editing has been shown to decrease overall performance of the model and is costly in terms of large-scale edits [Huang et al., 2025, *p.* 21f.]. Other mitigation strategies exist, targeting either the pre-training or inference stages of an LLM, which are presented in Huang et al. [2025, *p.* 22ff.], but will not be described here in detail.

Analogy to Web Search

After surveying the problem of hallucinations in the context of LLMs in a more theoretical manner, observations regarding a few use cases will be collected in the following. As a start, one observation regarding the use of LLMs can be found for web search systems pre- and post-arrival of prompt-based LLMs. When searching the web in a non-AI-assisted way,

[104] As early as 2023 in a position paper [Sainz et al., 2023], data contamination was noticed for LLMs in testing models across simple string matching operations.

a possible outcome may be that users are not capable to find what they are looking for with a specific query. This occurrence requires them to adapt the query or even abort the search while having some sense of certainty that the required search did not retrieve a satisfactory result. Studies have confirmed this observation regarding misinformation in a medical context during the COVID-19 pandemic, demonstrating the ability of users to discern the credibility of information when conducting the web searches themselves [Williams-Ceci et al., 2024].

Assessing User Behavior for LLMs

While the cited study is small in scope, the addition of LLM-assisted search has been shown to dilute the search results when compared to users conducting the web searches themselves [Nahar et al., 2025]. The high confidence with which LLMs present their generated answers, as described earlier in this section, contributes to this problem. Studies in this research field have been small in scope thus far. However, the impact of LLMs in web search has been felt widely in studies [Spatharioti et al., 2023] as well as recently through a number of direct AI-assisted search models as early as 2023, such as Bing generative search[105]. Other similar search models include ChatGPT search[106] and in a multitude of ways Google AI and their Gemini model[107], which is in use for the AI overviews used by Google in their Search Engine Result Page (SERP) since 2024, see also the discussion later in this thesis in Section 10.1.

After presenting observations from benchmarks, prevention mechanisms and web searches – all from the perspectives of hallucinations – two use cases will be presented, first on text summarization and second on question answering, once again focused on hallucinations. For both use cases, factuality, as already discussed, and the second category mentioned at the beginning of this section, faithfulness, are of concern. The latter, i.e., faithfulness, refers to problems due to the prompt-based nature of LLMs, see Section 6.1. In particular, instructions given in the prompt that were not adhered to or problems when processing the context given by the user are typical occurrences within this type of error. Both hallucination categories were noticed in the following use case, which will be described next.

[105] See `https://blogs.bing.com/search/July-2024/generativesearch/`.
[106] See `https://openai.com/index/introducing-chatgpt-search/`.
[107] See `https://blog.google/products/search/google-search-ai-mode-update/`.

Thesis: Multilingual Text Summarization

> The NLP task of multilingual text summarization is approached through five distinct application scenarios, see Dal Cin [2025]. These scenarios encompass the common news summarization task as well as the generation of scientific abstracts, web snippets, school materials and email threads. Three models were applied to generate summaries, one traditional model using extractive summarization and two abstractive models, one of them representing a modern chat-based LLM. Evaluation was conducted quantitatively using ROUGE and qualitatively using various summarization parameters.

Multilingual Summarization Scenario

When focusing again on NLP applications, hallucinations have been observed in abstractive approaches within the research field of text summarization [Ji et al., 2023, *p.* 15ff.]. The use case of text summarization has been widely studied in NLP and is one of the premier uses of LLMs, especially now as the state of the art is dominated by LLMs [Zhang et al., 2024a]. In order to analyze the impact regarding size of language models, three models were applied in Dal Cin [2025]: One purely extractive model, here LexRank [Erkan and Radev, 2004], was contrasted with a large pre-trained language model based on the T5 architecture, here mLongT5 [Uthus et al., 2023], and a locally executed LLM with available weights, here Mistral NeMo[108]. Evaluation was carried out quantitatively regarding ROUGE (see Section 3.6), length and compression metrics, the degree of extractivity[109] and a measure to detect truncated summaries, which can appear due to the limited output window size for the abstractive models.

Qualitative Evaluation

Qualitative evaluation was also conducted, for which different criteria were evaluated: Grammar, lexical use, structure, coherence, style and use.[110] The most relevant feature for this discussion is the criterion of factual correctness [Dal Cin, 2025, *p.* 28ff.]. Since LexRank as representative of an extractive summarization approach does not rewrite the text, the criterion for factual correctness always achieves full marks since no deduction could logically appear [Dal Cin, 2025, *p.* 29]. However, even LexRank and its extractive approach could produce faulty summaries regarding structure and coherence, due to sentence ordering that did not match the original text or that appeared to be illogical and thus led to texts that were incoherent and structurally unsound [Dal Cin, 2025, *p.* 38f. and *p.* 48f. and *p.* 52f. and *p.* 55 and *p.* 57f.]. Thus, hallucinations in a strict

[108] See `https://mistral.ai/news/mistral-nemo/`.
[109] Defined as *density* in Grusky et al. [2018, *p.* 712].
[110] See Section 7.2.2 for a review of these criteria.

sense did not appear, but due to structural problems and sentences including facts that were not introduced beforehand, it could appear to the reader that faults akin to hallucinations were still produced by the extractive approach.

Appearance of Hallucinations

Both other approaches, mLongT5 as the applied pre-trained model and NeMo as the LLM, produced hallucinations, although in varying degrees. The first model, mLongT5, produces factual mistakes, i.e., hallucinations, in every generated summary including wrong numbers [Dal Cin, 2025, *p.* 40], fabricated definitions [Dal Cin, 2025, *p.* 53 and *p.* 55f.] or unfitting relationship statements between people [Dal Cin, 2025, *p.* 58]. The second model, NeMo, while mostly producing significantly better summaries than the other approaches, tends to fabricate additional information in some cases that is not consistent with the content in the original text in two application scenarios [Dal Cin, 2025, *p.* 44 and *p.* 50]. In other scenarios, NeMo, produces satisfactory summaries, from a factuality point of view [Dal Cin, 2025, *p.* 54 and *p.* 56f.], but also performs worse in the summarization of email threads [Dal Cin, 2025, *p.* 60]. Lastly, regarding faithfulness, NeMo does not always follow the instructions given in the prompt regarding the desired output dimensions [Dal Cin, 2025, *p.* 62f.].

Thesis: Question-Oriented Access to Study Regulations

> The NLP task of question answering was approached in a custom scenario regarding the question-oriented access to study regulations, see Möglich [2025]. After determining typical questions from students regarding their regulations, such as exams, deadlines and modules, three models were applied to identify the relevant paragraphs from the study regulations, i.e., TF-IDF as traditional method, a BERT-based approach and a modern chat-based LLM. Evaluation was conducted quantitatively using a custom evaluation metric based on a survey and qualitatively through analyzing typical failure cases as well as the impact of preprocessing and additional features.

Strategies to Prevent Hallucinations

In another study, one unobtrusive method of potentially preventing hallucinations from appearing was tested, i.e., the use of an additional instruction within the prompt, as applied in Möglich [2025]. The relevant instruction in this case aims to prevent the LLM from generating new text and to only reuse or identify text that was supplied by the user within the input text. Therefore, this method is intended to prevent faithfulness issues and would appear similar to extractive scenarios within information extraction or summarization applications. This strategy of purely select-

ing preexisting text is limited to question answering or extraction scenarios, which do not require an abstractive text generation aspect. In the question answering task from Möglich [2025, *p.* 61], the generation of additional text that was not present in the supplied study regulations was not noticeable and therefore this strategy proved successful in preventing hallucinations.

Summary of this Section

In the end, hallucinations in terms of fabricated content in LLM-produced generations are appearing in diverse use cases and are – at least at the time of writing this thesis – a yet to be solved problem, as continued research[111] and the shared task at SemEval2025[112] demonstrates. Exacerbating the problem of hallucinations and impacting many other facets of the evaluation and assessment of NLP methods in general are issues with metrics themselves, and their application on tasks in particular, which will be detailed in the next section.

7.2 Lacking Metrics

As detailed in Section 6.2, the reliance on quantitative evaluation and therefore on appropriate metrics for assessing the quality of NLP approaches is common practice in the research field. The dependency on reference-based metrics is just one example of the limitations of a widely used evaluation practice. Observations from selected use cases will be presented in the following on NLG approaches in Section 7.2.2. But first, the use case of keyphrase prediction will be showcased, since both aspects of extractive approaches as well as generative methods were applied there, which will detail the challenge of lacking metrics for both traditional and modern methods.

[111] See new research, as of May 2025, such as Ravichander et al. [2025] or Shelmanov et al. [2025].

[112] See *Mu-SHROOM, the Multilingual Shared Task on Hallucinations and Related Observable Overgeneration Mistakes* in Vázquez et al. [2025].

7.2.1 Keyphrase Prediction

Some parts of this subsection have been inspired by a supervised thesis, in particular the experimental results, see Lang [2025].

Thesis: Marginalia Generation

> The NLP tasks keyword and keyphrase generation are approached through one application scenario, i.e., the generation of marginalia, see Lang [2025]. This scenario presents a dataset of textbooks on computer science and studies the different types of available marginalia. Twelve models were applied to generate or extract the keyphrases, seven traditional models with extractive techniques, and five more modern approaches, one being a chat-based LLM. Evaluation was conducted quantitatively using the f1-score, the Mean Reciprocal Rank (MRR) as well as the MoverScore and qualitatively through an online survey.

Marginalia Generation in Textbooks

The use case presented in this subsection revolves around the generation of marginalia, meaning the brief keywords and keyphrases in the margins of books, usually non-fiction books or in this use case textbooks[113], that capture the content of the accompanying paragraph as a brief summary or reference when skimming the book. A distinction regarding the type of marginalia can be made regarding absent and present marginalia. The former type refers to marginalia that do not appear in that exact form in the paragraph itself, while the latter do appear verbatim in the text. Whereas marginalia have been studied regarding the notes of readers themselves annotating books, see, e.g., Jackson [2001], the use of marginalia as a form of glossary provided by the author of a book, either in a quantitative or qualitative manner, has not been studied broadly yet.

Marginalia Prediction Study

Likewise, the automatic prediction of marginalia has not been researched yet, as of 2024. Furthermore, the contrast between abstractive and extractive systems was another focal point of this use case. Therefore, a study was conducted to analyze the feasibility of approaching marginalia prediction using available NLP techniques such as text summarization, keyword extraction, keyphrase generation or even topic modeling, which was also published in Lang et al. [2025]. Due to the lack of appropriate datasets on marginalia, data from textbooks[114] concerning information technology were collected in order to assess the fit of these methods

[113] Or the keyphrases used in the margins of this thesis.

[114] To guarantee consistent processing, only textbooks in German and published by the German publisher dpunkt.verlag, a subsidiary of O'Reilly, were selected.

[Lang, 2025, *p.* 9ff.]. During analysis of the dataset, quantitative observations regarding the marginalia revealed that uniformity across the marginalia is not a given. The distinction between absent and present marginalia has already been defined, but many other categories have been noticed, each regarding the respective paragraph, such as marginalia as summaries, as additional information, as questions towards the reader among others [Lang, 2025, *p.* 11ff.].

Keyphrases Selection in NLP Tasks

A decision regarding the desired type of marginalia had to be made since a more narrow and clear problem definition was preferable to further define the processing pipeline. Thus, full sentences were filtered out of the dataset, which is why summarization as an NLP task was eliminated from the list of possible techniques. Likewise, the dimensions of typical paragraphs were deemed too small to appear feasible for topic modeling approaches. Furthermore, since phrases were a possible output dimension of marginalia apart from single words, keyphrases, which can produce words as well as phrases, were the output category of choice rather than keywords. Finally, since both absent and present keyphrases have been considered as valid, extractive and also generative approaches were chosen.

NLP Methods for Keyphrase Extraction

As baselines, statistical methods were selected for this purpose, including TF-IDF [Jones, 1972; Luhn, 1957], which despite its simplicity was shown to successfully process keyword extraction tasks [Yao et al., 2019a], and others such as YAKE! [Campos et al., 2018] and PositionRank [Florescu and Caragea, 2017], all being extractive approaches and consequently only capable of producing present marginalia. Other extractive methods, such as TextRank, that are commonly applied for summarization purposes, can also be executed on a word-level in contrast to the usual sentence-level for summaries [Mihalcea and Tarau, 2004], which is why TextRank was added as method, see, e.g., Yao et al. [2019a] for a similar application regarding keyword extraction. To also enable the generation of absent keyphrases, abstractive or generative approaches were also selected, including SIFRank+ [Sun et al., 2020], an unsupervised keyphrase extraction method based on a pre-trained language model, here ELMo[115], as well as mT5 [Xue et al., 2021] and lastly GPT-4o [Hurst et al., 2024] as representative of LLMs.

Model Evaluation and Performance Comparison

The predicted marginalia were assessed using a quantitative as well as qualitative evaluation. The f1-score, MRR [Craswell, 2009] and MoverScore [Zhao et al., 2019] were evaluated for twelve different models, show-

[115] ELMo, similar to BERT, represents a context-sensitive word embedding architecture [Peters et al., 2018].

ing GPT-4o and mT5 as the leading models in all metrics by a considerable margin compared to all other models [Lang et al., 2025, *p.* 232]. The qualitative evaluation confirms these findings. Therein, users were presented two marginalia candidates and the accompanying paragraph and were asked to choose the more fitting marginalia, or if both are fitting equally as well [Lang et al., 2025, *p.* 233f.]. Only five models were assessed in this manner, due to time constraints for this evaluation. GPT-4o was the best-performing model when comparing the five models against each other, with mT5 once again also placing in the top two models overall, and the other models trailing behind substantially. In order to visualize the performance, the Elo rating [Elo, 2008], often used in rating chess players or in multiplayer video games, was used, which displayed the performance of the five studied models, see Figure 7.1. Choosing a different order of questions would lead to slightly different graphs, but the overall trend would stay the same, which confirms the quantitative results.

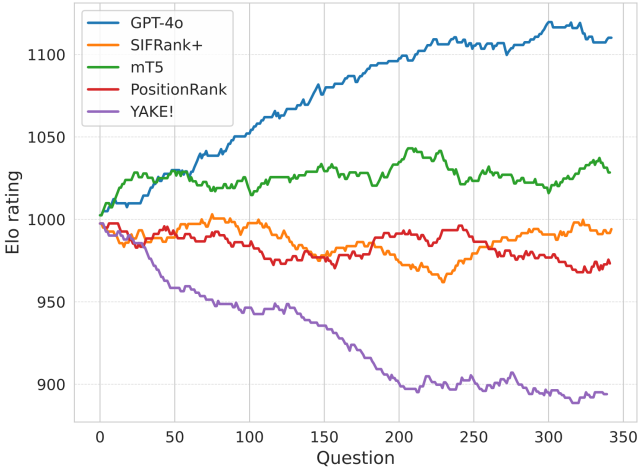


Figure 7.1: Qualitative evaluation results applying the Elo rating based on a user study for comparing the marginalia candidates produced by five different keyphrase generation systems. See Lang et al. [2025, *p.* 234].

Keyphrase Evaluation Metrics

The quantitative evaluation of keyphrase generation is widely dominated by the usage of the f1-score, followed by other metrics such as MRR [Papagiannopoulou and Tsoumakas, 2020, *p.* 19ff.]. However, as with other evaluation practices using references, exact matches have to be distinguished from partial or approximate matches, with the application of exact matches most commonly used [Papagiannopoulou and Tsou-

makas, 2020, *p.* 20f.]. Both versions confirm the findings from above, i.e., exact and approximate f1-scores were applied in Lang [2025]. Approximate matching is implemented using stemming and matching is applied on substrings in both directions, meaning the reference is searched as a substring in the extracted keyphrase and vice versa [Zesch and Gurevych, 2009].

Marginalia Evaluation Methods

Another difficulty is connected to the number of generated marginalia candidates and how they should be evaluated. Many approaches presented in Lang [2025] are able to produce different marginalia candidates, while others, such as mT5 and GPT-4o, the latter being explicitly prompted in that way, only produce one marginalia. Therefore, only f1@1, i.e., scoring only the marginalia candidate that was ranked highest for the models generating multiple candidates, was evaluated to ensure a consistent evaluation across all methods. However, approaches producing several marginalia candidates cannot be evaluated fairly in this manner, if the reference marginalia match the marginalia candidate not at rank one, but at rank two, three, or further down the ranked list of marginalia candidates [Lang, 2025, *p.* 38]. To combat this restriction, the MRR is applied for all methods once again, thereby including multiple candidates for those approaches that produced a list of marginalia candidates. TF-IDF and SIFRank+ achieve higher scores through this measure compared to the f1-score, but still trail behind both transformer-based methods, mT5 and GPT-4o [Lang et al., 2025, *p.* 232].

Micro- vs. Macro-F1-Score

More problems regarding the f1-score concern the distinction between micro-f1 and macro-f1, or rather the reasoning behind both metrics and the fact that in many papers only one of them is presented [Thomas and Vajjala, 2024]. With only one reference marginalia present, the macro-f1-scores are calculated for each individual instance, which are then averaged to a single macro-f1-score [Lang et al., 2025, *p.* 231]. Conversely, other publications apply the micro-f1-score, which is calculated using sums of true positives, false positives and false negatives across all predictions, and only afterwards calculate the total micro-f1-score [Lang, 2025, *p.* 37]. Thomas and Vajjala [2024], at least for keyphrase evaluation, recommend the macro-f1-score, or preferably both, while also stating that not all publications are transparent in this regard, i.e., do not mention which f1-variant is applied [Thomas and Vajjala, 2024, *p.* 9725].

Limitations of MoverScore for Marginalia

While the problem of only using exact string matching is tackled using approximate matching and the production of several candidates are assessed by using MRR, only lexical similarity can be evaluated in this manner. Grammatical and syntactic correctness or even semantic sim-

ilarity, by use of synonyms, homonyms or similar terms cannot be captured, which is why MoverScore [Zhao et al., 2019] is further used to evaluate the generated marginalia [Lang, 2025, *p.* 36f.]. Contextualized BERT-based embeddings are used to evaluate semantic similarity between marginalia and their corresponding paragraphs. During experiments, longer candidate marginalia scored higher in contrast to shorter marginalia candidates, leading to the assumption that MoverScore, which is often used for the evaluation of longer texts in text summarization approaches [Zhao et al., 2019], might not be applicable for highly specialized and rather niche use cases such as marginalia generation that are intended to produce shorter units of text [Lang, 2025, *p.* 42ff.]. This interpretation represents one conclusion that could only have been drawn due to careful observation of the produced metrics in combination with a manual qualitative study of the results and therefore showcases the potential limitation of a metric that is successfully used in similar, but still somewhat distinct applications.

Marginalia in this Subsection

To further illustrate the performance of the studied marginalia generation techniques, the marginalia in this subsection for all previous paragraphs, numbered P1 to P9, starting directly below the thesis synopsis in Section 7.2.1 have actually been produced by the GPT-4o version used in Lang et al. [2025] that produced the best results in their evaluation. The marginalia printed here are exactly as generated by GPT-4o, with the only adaptation being done by capitalizing the phrase in the same manner as was done for all other marginalia. To contrast the generated marginalia with the results from other systems,[116] the results are presented in Tables 7.1 to 7.3 and also in Table 7.4 for the marginalia that were manually created by the author of this thesis, as a point of comparison.

Observations for GPT-4o and SIFRank+

The results of this marginalia application, despite its limited size and scope, confirm the findings from Lang et al. [2025]. GPT-4o produces marginalia that are fitting for each paragraph from this subsection. They are furthermore similar in length and syntax to the manually produced marginalia, see in both Tables 7.1 and 7.4. In contrast, the marginalia produced by SIFRank+ are also displayed in Table 7.1, which are shorter, more general and do not fit the content of the paragraph, e.g., "subsection" as chosen marginalia for paragraph P1 is not thematically appropriate. Also, "marginalia" is generated twice, although the doubling of these marginalia should not be seen as a problem in and of itself, since no prevention of repeated marginalia is considered in the implementation. How-

[116] SIFRank+, mT5, TF-IDF, YAKE! and PositionRank were applied besides GPT-4o here.

P	GPT-4o	SIFRank+
P1	Marginalia generation in textbooks	subsection
P2	Marginalia prediction study	automatic prediction
P3	Keyphrases selection in NLP tasks	marginalia
P4	NLP methods for keyphrase extraction	statistical methods
P5	Model evaluation and performance comparison	marginalia
P6	Keyphrase evaluation metrics	quantitative evaluation
P7	Marginalia evaluation methods	marginalia candidates
P8	Micro vs. Macro F1-score	f1-score
P9	Limitations of MoverScore for marginalia	exact string matching

Table 7.1: Marginalia for the paragraphs of this subsection. The marginalia are listed for each paragraph (P) and per model, here GPT-4o and SIFRank+.

ever, thematically, "marginalia" alone is too broad for both paragraphs P3 and P5. The other marginalia produced by SIFRank+, while not as concrete as the marginalia from GPT-4o or the manually created suggestions from Table 7.4, would certainly be appropriate.

P	mT5	TF-IDF
P1	Use-case	books
P2	Predicting Marginia	information
P3	Through the distinct appearance of marginalia	approaches
P4	baselines	methods
P5	Predicted marginalia	models
P6	Evaluation of keyphrase generation	matches
P7	Another difficulty	candidates
P8	F1-score	f1-score
P9	MoverScore	similarity

Table 7.2: Marginalia for the paragraphs of this subsection. The marginalia are listed for each paragraph (P) and per model, here mT5 and TF-IDF.

Observations for mT5 and TF-IDF

The mT5-system, adapted from Lang [2025] to process English texts instead of the German textbooks in that application scenario, produces unreliable results, see Table 7.2. While the generated keyphrases for the three paragraphs P6, P8, and P9 are all fitting, the others exhibit problems. The marginalia "Predicting Marginia" for P2 includes a typographical error, which surprisingly is spelled correctly for P5 with changed capitalization as "Predicted marginalia", suggesting the interpretation of marginalia as a name in the former case. Moreover, the results for P1, P4, and P7 are too broad. Lastly, the marginalia "Through the distinct appearance of mar-

ginalia" for P3 cannot be easily explained, since a full part of a sentence was chosen here. The second model presented in Table 7.2, TF-IDF, produces the overall shortest marginalia, except for "f1-score" only consisting of one word each.[117] While all marginalia are appropriate and fitting for each paragraph, the limitation to roughly one word restricts the amount of information that can be included in these marginalia candidates. The extractive nature of this approach, however, prevents typographical errors or hallucinations from appearing.

P	YAKE!	PositionRank
P1	keyphrases in the margins	marginalia
P2	approaching marginalia prediction	approaching marginalia prediction
P3	define the processing pipeline	marginalia
P4	capture keyword extraction tasks	statistical methods
P5	qualitative evaluation	qualitative evaluation
P6	matches being most commonly	keyphrase generation
P7	number of generated marginalia	generated marginalia candidates
P8	metrics and the fact	macro-f1
P9	evaluated in this manner	exact string matching

Table 7.3: Marginalia for the paragraphs of this subsection. The marginalia are listed for each paragraph (P) and per model, here YAKE! and PositionRank.

Observations for YAKE! and PositionRank

The final two models, YAKE! and PositionRank, see Table 7.3, present two more traditional models that were both trailing behind GPT-4o and mT5 in the evaluation from Lang et al. [2025, *p.* 232ff.]. YAKE! produces marginalia that are mostly thematically relevant, however incorporates unnecessary parts of the respective paragraphs, e.g., the verbs in the candidates "define the processing pipeline" or "capture keyword extraction tasks" in P3 and P4. The results extracted by PositionRank are comparable to the marginalia produced by SIFRank+ from Table 7.1, with the same results for P3, P4, and P9 or similar candidates in P7 and P8. The appearance of "marginalia" twice, similar to SIFRank+ from above, while being a broad marginalia candidate, should not be discounted since, once again, no measures against repeated results are included in this experiment. Overall, the results for all paragraphs are thematically fitting, but either too broad for P1, P3, P5, P6 and P7 or too specific to be representative of the respective paragraph for P4, P8 and P9, leaving P2 as the sole fitting marginalia candidate. Both YAKE! and PositionRank however still

[117] The assessment of "f1-score" as two words is debatable.

trail behind the leading candidates from GPT-4o from Table 7.1 or the manually created candidates from Table 7.4.

P	Manually Created
P1	Marginalia Types
P2	Automatic Prediction of Marginalia
P3	Marginalia by Keyphrase Extraction & Generation
P4	Methods for Keyphrase Prediction
P5	Marginalia Prediction Performance
P6	Lacking Metrics for this Use-Case
P7	Marginalia Candidates
P8	Micro-F1 vs. Macro-F1
P9	More Advanced Metrics

Table 7.4: Manually created marginalia for the paragraphs of this subsection. The marginalia are listed for each paragraph (P) with a marginalia candidate, as the author of this thesis proposes for the paragraph.

Observations for Manually Created Marginalia

Overall, the manually proposed marginalia candidates are strikingly similar to the generated marginalia candidates from GPT-4o. The candidates that were suggested by the author of this thesis in Table 7.4 were written before the models from Tables 7.1 to 7.3 were executed on this small dataset. Thus, none of the models or model outputs influenced the manual creation of the marginalia. Even this small and constrained experiment confirms the findings from Lang et al. [2025], noting the strong performance of GPT-4o for this use case. The traditional models are in contrast either unreliable, see both PositionRank and YAKE! in Table 7.3, or produce candidates that are thematically fitting, yet broad and generic, in case of TF-IDF in Table 7.2.

Summary of this Subsection

As with other application scenarios, one single evaluation metric does not suffice in accurately assessing the performance of a model tasked with producing marginalia. Keyphrase generation is usually evaluated by using f1-score or MRR, but these metrics lack the ability to consider partial or approximate matches at least in a typical evaluation scenario [Papagiannopoulou and Tsoumakas, 2020, *p.* 19ff.]. In the end, other evaluation procedures are required to address the shortcomings of purely quantitative evaluations. For this study, advanced metrics like MoverScore are applied in addition to other evaluation paradigms provided by the Elo rating to compare different models as part of a qualitative user study. Also, an assessment of errors and a qualitative comparison between results produced by different models are further recommended.

After detailing the use case of marginalia generation regarding the difficulty in accurately evaluating the produced marginalia candidates, a discussion regarding NLG tasks will be presented next in Section 7.2.2.

7.2.2 Text Generation

Some parts of this subsection have been inspired by supervised theses, in particular the experimental results, see Dal Cin [2025], Fruth [2022], Gottwald [2023] and Matschat [2022].

Thesis: Natural Language Generation

> The NLP task of natural language generation is approached through the attempted generation of misinformation, see Matschat [2022]. This scenario encompasses the targeted generation of articles, similar to texts from Wikipedia, that however include intentional misinformation. Five models were applied to generate the texts, two traditional models and three more modern approaches. Evaluation was conducted quantitatively using BLEU, METEOR, ROUGE as well as BERTScore and qualitatively through analysis of the authenticity and stability of the generated texts.

Application Scenario for NLG

NLG as a research field was, as has been the case with many NLP tasks, greatly impacted by the rise of LLMs. NLG is itself also part of various other pipelines such as dialogue systems, machine translation or text summarization [Li et al., 2024, *p.* 6ff.]. The first NLG scenario presented here comprised the targeted generation of misinformation, in order to examine to what extent NLG approaches were able, at the time, to produce fake news akin to Wikipedia articles, i.e., including intentional pieces of misinformation within the text [Matschat, 2022]. The research was prompted by the admission of the developers of the GPT-2 model in their model card as one out-of-scope ability of their model: "Because large-scale language models like GPT-2 do not distinguish fact from fiction, we don't support use cases that require the generated text to be true."[118] Thus, the research question was proposed, if such techniques and other NLG approaches can produce text that is still coherent, grammatically correct and can further pass off as Wikipedia articles, while containing factual errors [Matschat, 2022, *p.* 2].

Implementation Parameter

In order to produce the texts, two traditional models – both variants of Markov chains – were applied in addition to one RNN-based technique implemented as an LSTM (see Section 2.3) as well as two GPT-2 models.

[118] See `https://github.com/openai/gpt-2/blob/master/model_card.md`.

As data basis, a Wikipedia dump was used, which was filtered to only include articles about people to receive a biography dataset. The goal was to initialize the NLG models with a name of a person as input and to generate the text following the name, i.e., ideally a short biography of the person as is standard for biographic articles in Wikipedia.

Quantitative Performance

The evaluation for this specific and non-standard research scenario was determined to include the standard evaluation metrics commonly found in either machine translation – here BLEU, METEOR and BERTScore – or abstractive summarization methods – here ROUGE – which all have been applied to NLG applications [Sai et al., 2022]. Comparing the produced text to a reference biography from Wikipedia might not completely entail the research question for these experiments, but due to the lack of a better metric in this case and the prevalence of the metrics for NLG tasks in general, these metrics were calculated despite the apparent drawbacks. In addition, a qualitative evaluation was conducted to accommodate for these drawbacks [Matschat, 2022, *p.* 42ff.]. The results of the LSTM model trail behind the other approaches, which however still do not show successful results in a consistent manner for all applied metrics [Matschat, 2022, *p.* 50ff.].

Qualitative Evaluation

Qualitative analysis consists of aspects such as cohesion, grammar, vocabulary, stable outputs and authenticity. Once again, the evaluation for the LSTM model revealed flawed generated texts in several categories, which is why the evaluations were focused on the other two models: Markov chains and GPT-2 [Matschat, 2022, *p.* 52ff.]. The former produced mostly grammatically correct text that, however, lacks cohesion and topical focus, and also includes incoherent information, e.g., temporal information without realistic connections. The latter improves upon Markov chains, especially on a cohesion-level, producing text that stays on-topic across several sentences and is also grammatically correct. GPT-2 further produces text that is more similar to the desired biographic nature of the texts, however once again with dates that are inauthentic and other factually incorrect information that are nonetheless presented in a plausible-sounding text. The interpretation of experimental results differs somewhat between the original goal of the approach presented in this NLG task and the overarching goals of this thesis. The specific goal of this task – generating texts that may appear as biographies from a Wikipedia page with built-in misinformation – could be achieved by the GPT-2 models in a mostly convincing manner, but with stronger restrictions for the Markov chain models and unconvincingly so for the LSTM model.

Performance of Traditional Models

Finally, while the aim of producing texts with misinformation was handled most convincingly by the GPT-2 models in this use case, the application of Markov chain models is noteworthy as a representative of traditional models. Efficiency as well as reproducibility, both of which will be discussed regarding this application below in Section 7.4 and Sections 8.1 and 8.2, are features that surpass the more resource-intensive GPT-2 models. Instead, the traditional models lag behind in important factors such as cohesion and inter-sentence relations. These problems can, however, also be attributed to the data basis, which, especially for data-driven stochastic NLG approaches such as Markov chains, is of the utmost importance [Gatt and Krahmer, 2018, *p.* 92ff.].

Metrics for Summarization

In this subsection, two further application scenarios will be analyzed: Text summarization and text simplification. Firstly, text summarization, despite its popularity in NLP research and as one of the more prominent application scenarios for LLMs, is often evaluated simply by using the default evaluation metrics in this field of study, meaning ROUGE, or other related metrics such as BLEU and METEOR [Dal Cin, 2025, *p.* 4ff.], see above in Sections 3.6 and 6.2. Simplification is not as comprehensively researched as text summarization but requires an even more versatile evaluation due to the degree of text simplification as an added dimension. Thus, a purely n-gram-based comparison through ROUGE or other similar metrics would not suffice. More complex metrics, such as BERTScore, are applied consequently.

BERTScore

BERTScore as one of the more computationally intensive metrics goes one step beyond previous metrics that use simple matches between words of the generated texts and the reference [Zhang et al., 2020c]. Instead, BERTScore computes semantic similarity using contextual embeddings and therefore enables matches across synonyms and other related words. Through this measure, the limitations of BLEU and ROUGE regarding their fixed vocabulary matches compared to available reference texts is softened and texts that use other words in their produced output texts are not penalized, if they still preserve the meaning of the reference texts. While one disadvantage of previous metrics is circumvented by BERTScore, other drawbacks appear. Due to the use of pre-trained language models as part of the comparison pipelines, requiring vector similarity calculations, the efficiency in terms of processing time decreases. However, as the authors note [Zhang et al., 2020c, *p.* 8], the difference only amounts to a roughly two to three times slower execution than a comparable BLEU implementation.

Limitations of BERTScore

More critical are two other factors. First is the primary focus of BERTScore on English. While Zhang et al. [2020c, *p.* 9] note that the score is also applicable to other languages if an appropriate replacement regarding the pre-trained language model is chosen, they admit worse performance for languages other than English, which has been discussed in other studies as well, e.g., for German in Aumiller et al. [2023, *p.* 202]. The restrictions resulting from the focus of NLP research on English will be discussed further down below in Section 7.3. Secondly, coherence, as mentioned above in Section 6.2, is still not captured by the more complex metric BERTScore due to the continued focus on lexical overlap. Despite BERTScore's advances compared to BLEU and others, the focus on lexical similarity is a limiting factor for use cases such as text summarization or text simplification, for which a well-structured and cohesive text is necessary or at least desirable.

Applying BERTScore for Simplification

The use of more complex scores in order to evaluate or improve a system can also be noticed in other contexts, e.g., in the text simplification task as part of the scores used in optimizing the rewards in the reinforcement learning approach, see Fruth [2022] and above in Section 7.1.1.[119] The optimization procedure within the text simplification approach applies four main rewards for simplicity, meaning preservation, fluency and guardrails [Fruth, 2022, *p.* 15ff.]. While simpler statistical metrics such as FRE and word frequency are used for lexical and syntactic simplicity respectively, BERTScore was applied for capturing meaning preservation. The reward for fluency is intended to measure grammaticality and readability, which are evaluation categories that are not represented by lexical metrics such as BLEU and BERTScore. Due to the complexity of this task, pre-trained language models have been applied through measuring the probability of the generated texts, see, e.g., Lau et al. [2017] or Laban et al. [2021] for more details on this procedure. Guardrails for brevity are simple in nature and other measures include the hallucination detection as described in Section 7.1.1. The detection mechanism is once more based on BERTScore to identify hallucinated entities even if they are not using the exact same words or phrases.

Scoring Weights for Simplification

The authors of the original English simplification approach did not discuss the weights of the different scores in their paper [Laban et al., 2021], however, the values can be retrieved from their published code base.[120] While the authors conduct an ablation study in order to independently

[119] Also published, more concisely, in Fruth et al. [2024].

[120] See `https://github.com/tingofurro/keep_it_simple/blob/main/train_keep_it_simple.py`.

verify the impact of each of the three main rewards and also discuss possible improvements for some of the rewards, e.g., to increase factuality [Laban et al., 2021, *p.* 6374], the missing discussion of the weights is an oversight there. In contrast, for the German adaptation, the scores are adapted for different training runs and analyzed more closely [Fruth, 2022, *p.* 39ff.]. Suitably adapted weights improve the performance of the overall system compared to default weights [Fruth, 2022, *p.* 42 and *p.* 57ff.].

Analogies between Score Weighting and Evaluation

This short aside is intended to show the analogies between optimizing scores within a pipeline on the one hand and the evaluation of overall systems regarding single metrics on the other hand. Balancing the weights for optimizing an NLP system in a reinforcement learning manner is thus comparable to selecting an appropriate metric for evaluating an approach. A direct link can be drawn here to the trade-off between prioritizing precision and recall for information retrieval systems. While combined and balanced metrics exist, such as the f1-score and its micro and macro variants as noted above in Section 7.2.1 an examination regarding the selection of metrics is necessary when approaching a new NLP task, discussed in more detail in Opitz [2024] for the evaluation of text classification.

Broader Perspective for Evaluation

Up until now, a fine-grained view on evaluation has been presented, i.e., which metrics to choose and the complexity included in this decision. A broader perspective may instead go beyond evaluation metrics and first define what type of output is intended to be produced, what form and structure it should be generated in, and for which target audience – with or without a certain degree of prior knowledge – it is geared towards. While this more far-reaching perspective is applicable to many NLP disciplines, and beyond NLP as well, a particular focus will be given here to text summarization due to its prominence in current research, its wide-spread application in modern LLMs and the experiments conducted for text summarization that are presented in this thesis.

Factors for Text Summarization

Text summarization as a task was analyzed as early as the 1990s regarding various broad factors, which were collected in Jones [1999] as *input*, *purpose* and *output* factors. Purpose factors, such as the intended target audience, and output factors, resembling the intended structure and type of output, will be discussed below. The third factor, i.e., input, relates to the form, scale, genre, subject type, structuredness or lack thereof regarding the input text(s) and finally the unit, i.e., if the input encompasses single or multiple documents. See more details regarding input factors in Jones [1999, *p.* 6f.].

Purpose & Use Factors

The other two factors, purpose and output, are similarly multifaceted. First, purpose factors include information on the intended situation, in

which the summary will be used, which can be known or unknown (*tied* or *floating* in Jones' terminology) [Jones, 1999, *p.* 8]. Another distinction revolves around the audience, which can be *targeted*, thus intended for a specific reader group or even experts, or *untargeted*, e.g., in case of a summary created for a mainstream audience. The targeted audience within the purpose factor differs from the subject type, mentioned just above for input factors, which deals with the context and prior knowledge of the source readers. The audience aspect within the purpose factors concerns the summary readers. The distinction here can matter, e.g., for heavily specialized contexts. Such contexts concern both source and summary readers, which require adapted processing and potentially a stronger focus on preventing hallucinations and the continued use of technical terms.[121] One last purpose factor is *use*, meaning which goal is intended to be accomplished by producing the summary. This goal can differ between refreshing, previewing, substituting or other tasks, but can also include covering advertisements or prompting goals, which try to convince the reader to access the source text [Jones, 1999, *p.* 8].

Multilingual Summarization

The purpose factor has been introduced in more detail due to its role for the text summarization approach in Dal Cin [2025], in which five multilingual application scenarios for different text summarization use cases were analyzed. The most prominent application of text summarization in research is news summarization, represented through the wide popularity of datasets such as the CNN/DailyMail corpus (see Section 3.6). News summarizations, referring to the definition of Jones [1999, *p.* 8], are intended to preview the original text, but could also be seen as prompting the user to access the full article [Dal Cin, 2025, *p.* 9]. Other mainstream uses of text summarization revolve around the representation of entire websites for search systems or aggregators, which use the summary as a means of either previewing the site or prompting the user to access the full webpage [Dal Cin, 2025, *p.* 10].

Different Use Cases for Summarization

The other three application scenarios from Dal Cin [2025] also include further goals and additional intended uses, when categorized according to Jones' classification [Jones, 1999]. Abstract generation based on scientific articles was characterized as multifaceted. The summary can be used as a preview of the full article or as substitution for the full article, if a quick glance suffices for the rapid browsing of many articles, or to refresh an already read article, thus recalling information that was already consumed [Dal Cin, 2025, *p.* 10ff.]. Similarly, at least for the last goal, the summar-

[121] A potential technique here is the substitution of such terms with more simpler words, akin to text simplification approaches.

ization of school material, which comprised the fourth use case, is categorized as refreshing the users' memory for content they already have read in the past. Finally, the last use case encompasses the summarization of email threads. Its intended goal is the same as the school material application scenario from just above, i.e., refreshing one's memory.

Audience Factor

The intention behind this discussion on the factor *use* is to raise awareness regarding the varying goals for summaries, which can differ depending on the use case. Prompting or previewing a text is inherently different from substituting or refreshing one's memory of the original source document. In planning a realization of an NLP task, similar discussions should be included in the conceptualization of an intended summarization task and other NLP tasks. As another example, the factor *audience* plays a strong role in simplification tasks. The distinction between *targeted* and *untargeted* has already been described, but particularly in simplification use cases, determining the degree of complexity of the generated texts should be considered more explicitly. Lexical, syntactical and semantic complexity can once again differ widely depending on the intended readership, varying between age group, people with learning disabilities or for foreign language acquisition. When proposing approaches, one should keep those intentions in mind when realizing a simplification approach, e.g., by defining the simplification procedures through the complexity of the training datasets, by adapting the prompts to include the intended user groups and the contexts, or through other measures.

Format & Material Factors

The third main factor apart from input and purpose, i.e., output, is once more similar, at a first glance, to already presented factors. One such aspect is the *format*, which captures the shape of the produced text, and is clearly related to form and structuredness, thus a continuous text or one being ordered in one way or another, e.g., through headings [Jones, 1999, *p*. 9]. The aspect *material* differs between summaries that are intended to capture the complete content of the source text, while other summarization tasks only require a partial recapitulation of the content from the source text.

Style Factor

The most interesting aspect concerning the output factors is the last main aspect, *style*. Two distinctive classes are of particular interest here, *indicative*, a short introduction to the topic of the source text but crucially not the content of the source text itself, and *informative*, which in contrast to indicative, actually presents the content of the source text and not only what it is about [Jones, 1999, *p*. 9]. Other types of style, such as *aggregative*, when dealing with the summarization of varied sources for the original texts, and *critical*, when also judging the source text in some manner,

among others are also conceivable for different use cases of text summarization.

Indicative vs. Informative Summaries

Once again, after the discussion for the factor *purpose* and its aspect of *use* from above, the application on five distinct summarization scenarios will be discussed for the *style* factor. All scenarios can be grouped into either indicative, informative or both categories [Dal Cin, 2025, *p.* 9ff.]. The summaries produced for news articles, abstract generation and web snippet generation scenarios all have indicative and informative aspects: They present a short introductory text for the main source document and also represent the content from the source in an abbreviated form. Differences between either class are apparent when defining the length of the produced summary. Shorter summaries tend more towards indicative summarization, whereas longer summaries tend towards being informative. While web snippets can be also informative, they are usually rather short and thus indicative. Abstracts are intended to capture the main contributions of a research paper, which concern aspects of an informative summary, whereas the shorter dimension of abstracts, usually ranging from several sentences to two or three paragraphs at most, once again tend more towards being indicative. News summaries are more dependent on the type of summary intended for the article in question. If higher information coverage is desirable, an informative summary is advisable, whereas shorter summaries usually only briefly describe the topic of the article in more general terms [Dal Cin, 2025, *p.* 9]. The other two scenarios can be characterized more clearly. School material summarization is informative, forming a pair with the purpose factor *use* and its goal of refreshing the user's memory on the topic. In contrast, the summarization of email threads is indicative. Here, not all emails need to be covered by the summary, instead the topic and result of the email chain are of higher importance.

Quantitative Multilingual Summarization

The five application scenarios were evaluated not only using criteria such as style and use, but also with quantitative evaluation metrics including ROUGE and other statistical values, i.e., the compression ratio, sentence and word counts, extractivity [Grusky et al., 2018] and the number of truncated summaries. Compression ratio, sentence and word counts are easily calculated. Extractivity returns higher scores if longer extractive segments, that appear exactly as in the original source document, are used in the summary. The number of truncated summaries is calculated since, in certain cases, the abstractive models produced texts in which the summaries are truncated due to limited context windows [Dal Cin, 2025, *p.* 27]. Finally, the n-gram-based matching metrics of ROUGE, using the

R-1, R-2 and R-L variants (see Section 3.6), are the only reference-based evaluation metric applied here.

Observations for ROUGE

The results achieved with the various methods – LexRank, mLongT5 and NeMo – were similar across all three methods when only comparing the scores for a single dataset in just one language. In contrast, when comparing the quantitative results for different datasets,[122] but from the same domain – in this case news summarization – and the same language – here English, German and French, respectively – the ROUGE scores showed high variance [Dal Cin, 2025, *p.* 30f.]. To further investigate this observation, a manual evaluation was conducted, assigning qualitative scores for a selection of summaries regarding five quality criteria[123] in addition to assessing lead bias as well as information dispersion. The manual evaluation did not confirm the findings through the ROUGE scores but instead provided comparable scores for the datasets from the same language, i.e., comparable scores for English, German and French datasets, respectively [Dal Cin, 2025, *p.* 30]. This observation confirms the limitations of reference-based evaluation metrics such as ROUGE. The limitations of ROUGE can also be noticed in another way: Datasets with longer reference summaries tend to produce better results in a quantitative evaluation since longer reference summaries provide more opportunities for n-gram based matches than shorter summaries. The correlation between ROUGE, or other quantitative summarization metrics, and human evaluation in terms of summarization quality for varying factors is an ongoing research problem and is actively studied, e.g., in Fabbri et al. [2021].

After the observations from a multilingual summarization scenario have been displayed, a monolingual use case will be shown next.

Monolingual Summarization

The monolingual scenario regarding German text summarization on different datasets was conducted in Gottwald [2023], which also includes a brief discussion regarding the evaluation metric ROUGE. Other facets of this study will be presented in more detail below in Section 7.3.2. ROUGE was recognized as a matching metric, purely based on n-gram overlap and thus incapable of representing the quality of summaries regarding coherence, semantic consistency or the structure of a summary [Gottwald, 2023, *p.* 45f.]. While ROUGE is certainly not intended to fully encompass all such parameters, the prevalence of the metric in text summarization re-

[122] For the news scenario, two different datasets for each English, French and German were tested, see more details below in Section 7.3.2 and Footnote 150.

[123] The five criteria are: Lexicon/grammar, structure/coherence, factual correctness, style and use [Dal Cin, 2025, *p.* 28ff.].

search and its importance despite its limitations are nevertheless noteworthy.

Thesis: Domain-Specific Text Summarization

> The NLP task of monolingual text summarization is approached through four domain-specific application scenarios for German texts, see Gottwald [2023]. These scenarios encompass summarization tasks on datasets comprising legal texts, news, how-to guides and Wikipedia articles. Four models were applied to generate summaries, one traditional model using extractive summarization and three abstractive models. Evaluation was conducted quantitatively using ROUGE and qualitatively using spot-checks of selected summaries in a comparative manner.

Summarization Planning

In the end, text summarization tasks should be analyzed beforehand, and the realization of the text summarization problem should be adapted towards the summary and its input, purpose and output factors, defined by Jones [1999], as mentioned earlier in this subsection. In addition to the already presented theoretical classifications for summarization approaches and observations for use cases, concrete measures can be recommended for prospective summarization tasks. Characteristics such as defining the length of the summary through simple parameter selection of traditional extractive models such as LexRank or by providing contextual information and guardrails within a prompt for an LLM are measures to conform to predefined requirements. Other factors, such as the intended audience or preserving the semantic complexity to adapt towards a sensible *style*, e.g., indicative or informative, are harder to realize, but should still be considered during implementation.

Other Flaws

When returning to the overarching topic of this section, lacking metrics, a deeper investigation into text simplification as well as text summarization evaluation, metrics and other evaluation aspects regarding the quality and requirements of a proposed NLP method have been presented. Unfortunately, flaws in datasets are increasing the difficulty in accurately evaluating and comparing approaches and techniques. Examples of such flaws are data quality issues, e.g., duplicates or uncharacteristic compression ratios, and issues with reproducing results of summarization papers, which have been studied for text summarization purposes with a particular focus on German in Aumiller et al. [2023, *p.* 208ff.]. Duplicates appear for example in the MassiveSumm [Varab and Schluter, 2021] dataset as well as other data quality issues such as frequent incomplete data samples in the same dataset, which contribute to compression ratio issues.

Lack of Gold Standard

Depending on the NLP task, only limited and small-scale datasets might be available for training and evaluation. Therefore, the problem arises, as already introduced above in Section 6.2, how to evaluate when no gold standard is available. One such use case will be presented below in Chapter 9 as one showcase of the lessons and targeted procedures created in this thesis. Briefly, the problem of text segmentation, as presented in Jegan and Henrich [2025b], concerns the processing of video transcripts into segments, for which no gold standard exists and therefore no comparisons can be carried out. Thus, purely statistical measures were applied as well as an efficiency analysis and finally a qualitative evaluation in order to compare different methods. This NLP task is highly variable since boundaries between segments can be set and interpreted in a multitude of ways, which is why even the existence of one single gold standard for comparison's sake might result in the same restrictions mentioned above during the discussion of other reference-based evaluation metrics and also in both Section 3.6 and Section 6.2. More details regarding the text segmentation scenario will be presented below in Chapter 9.

Summary of this Subsection

In summary, the problems presented in this section reveal the need for analyzing not only the applied metrics, but also the evaluation approach in general and the need for a qualitative evaluation as well as an analysis before implementing a concrete NLP approach. The realization of such qualitative evaluations can vary. User studies or evaluations by domain experts would be recommended in most use cases, however budgetary or temporal reasons might restrict these proposals. Crowdsourced evaluations or spot-checks by a limited number of participants are just two measures for circumventing excessive expenses or efforts to at least enable a small-scale qualitative evaluation. Regardless of the applied scale of evaluation, a purposeful analysis of the goals of both the evaluated approach[124] as well as the evaluation itself needs to be conducted first. In the end, a preferable way of approaching NLP methods is proposed here in addition to studying purely quantitative metrics, as it is in other parts of this thesis as well, see Sections 6.2 and 9.3. Similarly to Table 5.1, the aspects presented for evaluation and metrics include observations gained from supervising and implementing text processing projects, which are not necessarily generalizable to other use cases in the respective task areas.

Still, as recommendations, these discussions hold value and are helpful for future NLP methods, as will be further presented in Chapter 10. Fur-

[124] See the discussion of different goals within text summarization regarding various factors proposed by Jones [1999] earlier in this subsection.

ther problems of use cases regarding the primary focus on approaches intended for English will be presented in the next section.

7.3 Prevalence of English Data & Models in Current NLP

The previous problems discussed above dealt, firstly, with hallucinations in Section 7.1 impacting abstractive or generative approaches, and secondly, with lacking metrics in Section 7.2, which are a broader problem that influence all manners of methods. This section details the focus on English-based approaches and discusses diverse application scenarios including extractive scenarios presented in the first subsection with less invasive adaptations for a German information extraction use case in Section 7.3.1. Other more expansive tasks applying extractive summarization or even classification scenarios among others, will be described further below in Section 7.3.2.

7.3.1 Low-level Adaptations for German Tasks

Some parts of this subsection have been inspired by supervised theses, in particular the experimental results, see Pulver [2023] and Lang [2025].

Information Extraction Use Case

The first application scenario, dealing with the difficulty when porting approaches intended for English to German, concerns the task of information extraction, see also Jegan [2024]. In this study, two parts of information were extracted from a dataset of meta-information on a large number of businesses [Pulver, 2023]. First, the legal form, such as "GmbH"[125] or "AG"[126] was extracted from the business name, and second, the type of industry or trade such as "Windpark", German for wind farm, or "Anwaltskanzlei", German for law firm, again extracted from the business name.

[125] Most closely related to the legal entities of a private limited company in the United Kingdom or limited liability company in the United States.

[126] Referred to similarly as a public limited company in the United Kingdom or stock corporation in the United States.

Thesis: Information Extraction

> The NLP task of information extraction is approached through two application scenarios, see Pulver [2023]. These scenarios identify information from a dataset with information on businesses: On the one hand the extraction of the legal form of a business and on the other hand the type of industry or trade. Depending on the task setup, this scenario can also be approached as a classification task. Four models were applied to extract the information, three traditional models and one more modern approach, with additional ensemble techniques combining the approaches. Evaluation was conducted quantitatively using accuracy, precision, recall, f1-score as well as runtime measurements and qualitatively through an analysis of incorrectly labeled data.

German Stemming & Lemmatization

While the adaptations for the German language were rather minuscule for this approach compared to other use cases, e.g., see the later discussion for Dal Cin [2025] and Fruth [2022] in Section 7.3.2, the preprocessing needed to be updated to properly reflect the language change. In place of English stemming algorithms such as the popular Porter stemmer, exclusively available for English, see Porter [1980], the same author initialized the Snowball programming languages, with which stemmers can be written for various languages [Porter, 2001]. The resulting Snowball stemmer for German was thus used and similarly, lemmatization was conducted using the UDPipe lemmatizer, applying the Universal Dependencies treebank for German.[127] Further preprocessing necessitated the rule-based processing of dictionaries before stemming was applied in order to prevent the stemming of names regarding persons or places, which were found to produce problems. Thus, German dictionaries for given names, surnames and place names were applied and if matches were found, stemming was not applied in those cases. For more information on the preprocessing, see Pulver [2023, *p.* 27].

German Compound Words

Another update regarding the pipeline was necessary, i.e., the processing of compound words. In contrast to many other languages, German uses single orthographic words for compound words, meaning a single word is used instead of multiple words representing a single concept.[128] The pervasiveness of this phenomenon was studied in Baroni et al. [2002, *p.* 4f.], which analyzed the German APA corpus and found that 47% of word types were compound words, however noting that only

[127] See `https://universaldependencies.org/`.

[128] E.g., the English words "death star" represent the same compound as the single German word "Todesstern".

7% of all tokens appear to be compounds, or put another way, over 80% of compound word types were used five times or less. Compound words are split for this use case by applying the n-gram-based algorithm presented in Tuggener [2016], which calculates probabilities at different positions within words and selects the highest probability for splitting the word.[129]

Rule-Based Processing

One further adaptation should be mentioned here, regarding the rule-based processing of the dataset. In identifying the legal form of the business names, the rules in order to recognize entities such as "GmbH" needed to be written by hand.[130] Since similar rule-based systems, even for English, need to be adapted for the specific use case at hand, this workflow is not a limitation of languages other than English, and should not be included as a restriction in this subsection, but should just briefly be mentioned as part of the processing pipeline.

Identification of Business Type

In summarizing the performance of this approach, only the identification of the type of business will be discussed here since the application of the rule-based approach for predicting the legal form is more fittingly placed below in Sections 8.1 and 10.2. Comparing the results from the approach on German discussed in this subsection to other datasets or implementations is difficult, since the used dataset is not openly available, because it was supplied by an industry partner and was intended to solve a business use case for that partner. At least in theory, comparable studies, e.g., in Shapiro [2020], reported an f1-score of 60.2% for their approach on small business classification for 66 classes of business types on English data. In contrast, the system in Pulver [2023, *p.* 44ff.], reaches an f1-score of 58% for the evaluation of the dataset containing 10 classes using a FastText classifier [Bojanowski et al., 2017], see Section 2.3. Other, more traditional models, such as Naive Bayes, SVMs and decision trees were applied here as well with Naive Bayes barely trailing behind the performance of FastText, while the others achieve worse scores [Pulver, 2023, *p.* 47].

Adaptation for Marginalia Generation

Similarly to the information extraction use case already discussed, the required adaptations for the methods applied in the marginalia generation task were limited in scope. The extractive methods only required small updates, e.g., the selection of a German stop word list. Another update, which is noteworthy but would also be required for other languages including English, is the restriction on selected PoS tags for the marginalia

[129] See also `https://github.com/dtuggener/CharSplit/`.

[130] The work done for information extraction in this approach was conducted mostly in 2022, thus before the rise of chat-based LLMs and therefore manually. Nowadays, LLMs can certainly also play a role in writing or at least suggesting such rules.

candidates [Lang, 2025, *p.* 29]. After manual evaluation of the reference marginalia, some PoS categories were excluded in this manner for extractive methods such as SingleRank and TextRank that appeared in only a select few reference marginalia. Therefore, only often used PoS tags were allowed for marginalia candidates. PositionRank goes one step further by allowing the addition of regular expressions as a simple grammar to dictate the specific order of desired keyphrase candidates [Florescu and Caragea, 2017, *p.* 1109], which was also adapted to conform to the reference marginalia from the dataset.

Transformer-Based Approaches

As with the approach presented above from Pulver [2023], the adaptation of the transformer-based approaches requires slightly more extensive updates. For the SIFRank+ model, a German version[131] of the ELMo model [Peters et al., 2018] was used as the pre-trained language model, as well as several more measures for the preprocessing pipeline such as updated Stanford CoreNLP [Manning et al., 2014] models for German or language-specific stop word or word frequency lists. Other measures include the use of German language-specific stemming or lemmatization approaches, which will be more thoroughly discussed below in Section 7.3.2. Both information extraction and keyphrase prediction tasks were thus adapted by updating existing methods in a limited manner, by simply substituting smaller constituents or by providing German resources, such as stop word lists.

Influence of Preprocessing Measures

These results present an insight into approaches, in which the adaptation towards another language is achievable through only a few adjustments. The appropriate selection of stemming and lemmatization algorithms is a simple measure, applying commonly used software packages like the Snowball stemmer. Moreover, the treatment of compound words is a more language-dependent aspect due to varying degrees of importance of this phenomenon depending on the language. For German, the resolution of compounds can be a helpful measure in improving the performance of NLP tasks. In Pulver [2023, *p.* 45f.], the influence of different preprocessing methods was analyzed, and the inclusion of separated compound words proved to be the most successful single measure during preprocessing.[132] More direct comparisons between languages will be

[131] See `https://github.com/t-systems-on-site-services-gmbh/german-elmo-model/`.

[132] Other tested preprocessing methods included stemming, lemmatizing, adding potential synonyms and resolving abbreviations [Pulver, 2023, *p.* 38ff.]. Compound splitting was done in two variants: Either by replacing the compound words with their split words that are stemmed or by adding the split words to the compound words instead of replacing them [Pulver, 2023, *p.* 46].

presented in the next subsection, which also include more expansive adjustments for the respective languages. These more extensive measures are necessary for text classification, text simplification and text summarization tasks, which will be shown in the next subsection.

7.3.2 Wide-ranging Adaptations for Non-English Language Tasks

Some parts of this subsection have been inspired by supervised theses, in particular the experimental results, see Dal Cin [2025], Fruth [2022], Gottwald [2023], Harms [2024] and Raab [2023].

This subsection presents the influence and dominance of English on the NLP landscape for three distinct application scenarios: Text classification, text simplification and text summarization. While they share similarities, which will be noted, the distinct purposes of each task justify a separate discussion.

Text Classification

Thesis: Text Classification

> The NLP task of text classification is approached through two distinct application scenarios, see Raab [2023]. These scenarios encompass the processing of two domain datasets, one from the legal domain and the other representing the insurance domain. Three models were applied to classify the texts, one traditional model based on SVMs and two more modern models. Evaluation was conducted quantitatively using f1-score and qualitatively using cluster visualization techniques.

Legal & Insurance Text Classification

The application for classification discussed here is rooted in domain-specific datasets, from the legal and insurance domains. The former is an existing dataset, presenting a subset of clauses from the German Civil Code[133] that are categorized into nine semantic classes and have been published in Glaser et al. [2018]. The classes represent various aspects of the clauses represented in the dataset, which concerns a part of the BGB related to tenancy law.[134] The latter dataset was, similarly to the information extraction task presented just above, constructed by cooperating with an

[133] Commonly known in German as the Bürgerliches Gesetzbuch (BGB).

[134] These classes are: "Permission", "Duty", "Consequence", "Objection", "Reference", "Continuation", "Prohibition", "Definition" and "Indemnity". They constitute different semantic categories of tenancy law.

industry partner, which provided General Terms and Conditions (GTC) used by German insurance companies, see Raab [2023, *p.* 25ff.], which included twelve classes. Semantically, these classes represent different types of business transactions that are relevant for the assessment of an insurance company regarding a claim, its rating as well as associated risks.[135]

ProtoBERT & SetFit

Both datasets are intended to be approached as a multi-class classification problem. As for models, SVMs were used as baselines and representatives of traditional models, while two modern methods based on few-shot learning were tested, in this case ProtoBERT and SetFit. The first model, adapted from Tänzer et al. [2022] and their ProtoBERT model, combines the capabilities of pre-trained language models and prototypical networks to handle little training data in a few-shot learning scenario. The second model, using the SetFit architecture from Tunstall et al. [2022], is using a two-stage process. First, a pre-trained sentence transformer model is fine-tuned, and second, the trained model from the first stage is applied to generate text embeddings, which serve as the basis for the training of a classification system.

Choice of Language Model

The quantitative evaluation for the classification tasks in terms of performance will be presented below in Section 8.3. More relevant for this chapter are the required adaptations to the German language, as also noted in Jegan [2024]. Since pre-trained language models are applied in both modern systems, embeddings that are trained with data from the language in question are required, which limit the variety of choices. Also, since pre-trained language models usually are not primarily trained on data from the specific domain of the use cases, another factor regarding the downstream task is relevant, i.e., fine-tuning. Therefore, the final fine-tuned model for this specific use case requires domain-specific datasets, which are scarce in many languages and only limitedly available for German. In this case, legal data was available, however, insurance data did not exist prior to this work.

Cased vs. Uncased

While the choice of language model solely due to the selection of its training data is crucial, one further aspect has not been discussed yet, i.e., the decision regarding cased and uncased words as well as special

[135] These classes are, in German (and translated into English): "Tod" (Death), "Rückkauf" (Surrender), "Wiederinkraftsetzung" (Reinstatement), "Beitragszahlung" (Payment of Premiums), "Kosten" (Charges), "Beitragsbefreiung" (Waiver of Premiums), "Umwandlung" (Conversion), "Reduktion" (Reduction), "Erhöhung" (Increase), "Rentenphase" (Payout Phase), "Überschussbeteiligung" (Participation in Profits) and "Irrelevant" (Irrelevant). They are types of transactions that are described in the GTC of an insurance provider.

language-dependent letters such as the umlaut[136], see Raab [2023, *p.* 32ff.]. German nouns are capitalized, i.e., they carry semantic information solely due to the capitalization of its first character. Also, the normalization of special characters into Latin script, thus reducing the contextual information of all letters into the 26 letters of the English (or Latin) alphabet removes information, that could potentially help in deciphering synonyms that in their original form were clearly distinct.[137] The cased versions of ProtoBERT, processing German noun capitalization and the umlaut letters appropriately, surpasses the performance of uncased models by more than 11% regarding the f1-score for the legal dataset Raab [2023, *p.* 37].[138]

Other Languages

Lastly, the role of non-English domain-specific classification has been recognized and evaluated in other studies as well, e.g., for the domain of German legal text classification [Glaser et al., 2018, *p.* 2]. Italian legislative texts [Biagioli et al., 2005; Francesconi and Passerini, 2007] and Dutch legislation [de Maat et al., 2010; Steegh and Sileno, 2023] serve as examples for European representatives of approaches in the legal domain on non-English languages, all with more or less extensive adaptations regarding the respective language.

Text Simplification

Unsupervised Text Simplification

In the text simplification approach from Fruth [2022], which was already discussed regarding hallucinations in Section 7.1.1 and in terms of lacking metrics in Section 7.2, many adaptations regarding German were applied that surpass the efforts from the classification approach from Raab [2023], see above, in both complexity and scope, as also referenced in Jegan [2024]. The first problem regarding text simplification in general is the lack of parallel datasets, even more so for non-English languages. One potential approach to the lack of parallel data, or at least some sort of mitigation, is an unsupervised system. Another argument in favor of unsupervised approaches for generative purposes in general and not exclusively for simplification is the dependency on references for training

[136] Language-specific letters due to historic sound shifts, for German *ä*, *ö* and *ü*.

[137] See, e.g., the difference between "schon" (already) and "schön" (beautiful) or "Mutter" (mother) and "Mütter" (mothers) [Raab, 2023, *p.* 32]. However, both can be problematic depending on the implementation details. A simple substitution of "ö" with "o" instead of the appropriate substitution with "oe" in an alphabet with 26 letters would result in a semantic change in the former example, while the latter produces a mismatch between singular and plural when replacing "ü" with "u" instead of "ue".

[138] Since punctuation is recognized by these models, no differences are noticeable in determining sentence borders for both cased and uncased models [Raab, 2023, *p.* 33].

and evaluation, see the discussion on reference-based evaluation metrics above in Section 6.2.

Measuring Meaning Preservation

The concrete measures in adapting the English pipeline from Laban et al. [2021] have already been partly mentioned in Section 7.2 as part of the discussion in using complex metrics in order to model languages other than English more accurately. One such example is the metric BERTScore, which was used to measure meaning preservation between the original and simplified texts. The development of this BERTScore-based method differs from the original method from Laban et al. [2021, *p.* 6369], which was necessary due to the unavailability of corresponding German models and datasets for calculating meaning preservation and experimental implementations. This measure, however, did not achieve sufficient performance for use in the simplification pipeline, see more details in Fruth [2022, *p.* 21]. This concrete detail serves as just one example to showcase the difficulty in porting an approach intended for one language to another potentially more complex language with less resources available.

Measuring Fluency

Another customized part of the pipeline was required for measuring fluency. For the German simplification pipeline, a BERT-based model was fine-tuned on German Wikipedia articles, which proved to be more appropriate for the German texts when applied to the datasets [Fruth, 2022, *p.* 25]. In evaluating the fluency scores, however, the complexity of German in contrast to English resulted in both worse scores overall and the appearance of another particularly German phenomenon: The presence of different definite articles depending on gender, case and number of the associated noun. The repetition of multiple definite articles resulting in higher scores led the generator to generate texts with flawed article usage, which motivated the introduction of an article repetition penalty [Fruth, 2022, *p.* 26].

Lack of Comparison Models

After customizing the approach to conform to the characteristics of German, another problem unfortunately became apparent, the lack of other models in non-English languages for a comparison. For evaluating NLP approaches the usual procedure, at least for a quantitative evaluation, is composed of comparisons to baselines and other models processing the same data as the approach currently being tested. For language-independent tasks or for popular research scenarios like text summarization or text classification, a myriad of models is available, which is, however, not the case for more narrow and less researched applications like text simplification. Thus, a lack of models for paragraph-level text simpli-

fication for German was noticed and no comparable method for testing could be found at the time.[139]

Pivot Model

Consequently, to solve this lack of comparison, a pivot model was introduced [Fruth, 2022, *p.* 47ff.]. The pivot model translates the original text into English, processes the translated text with the approach from Laban et al. [2021] into a simplified form, and translates the simplified text back to German. The translation was applied using transformer-based models trained on the opus dataset,[140] which were state-of-the-art at the time of implementing this work [Tiedemann and Thottingal, 2020]. While the translation introduces new complexities, without using this measure of procuring a comparison model, no quantitative evaluation and therefore only a qualitative evaluation would have been possible. The evaluation showed promising results for the adapted German simplification model, especially for meaning preservation and the degree of achieved simplifications but also acknowledged a higher degree of fluency for the texts produced by the pivot model in contrast to the German simplification model [Fruth, 2022, *p.* 48 and *p.* 54].

The simplification scenario discussed in this subsection is intended to showcase the difficulty in adapting an approach for English to other languages and also the subsequent complexity in evaluating approaches in less researched NLP tasks.

Text Summarization

Extractive Summarization

For text summarization approaches a plethora of models and datasets are available. In three studies, Dal Cin [2025], Gottwald [2023] and Harms [2024], extractive and abstractive summarization approaches were tested. The former work represents a multilingual setting, as already described above for hallucinations (Section 7.1.2) and lacking metrics (Section 7.2), whereas the latter two approaches process German texts only, i.e., both are monolingual approaches.

Monolingual Summarization Use Cases & Adaptations

Adaptations for the monolingual approaches for German texts were not necessary, i.e., the LexRank [Erkan and Radev, 2004] and TextRank [Mihalcea and Tarau, 2004] summarization methods were applied without any customizations. The first scenario on German texts, see Gottwald [2023], revolved around the summarization of German legal texts[141], news

[139] Most of the work done here was conducted in the first half of 2022, thus before the rise of general-purpose LLMs.

[140] See `https://huggingface.co/Helsinki-NLP/opus-mt-de-en/` and `https://huggingface.co/Helsinki-NLP/opus-mt-en-de/`.

[141] Using the LegalSum dataset from Glaser et al. [2021].

articles[142], instructions from the WikiHow website[143], and Wikipedia pages[144], thus a selection of domain-specific datasets to gain an understanding of differences between these domains. The second scenario, see Harms [2024], also processed German texts, but for a dataset that was manually procured. The goal of this study was to summarize school websites, which present a challenge for extractive and abstractive models due to their diverse structure, layout and content as well as their contrast to the training data the models are initialized with. In terms of extractive approaches, LexRank [Erkan and Radev, 2004] was once again applied for this use case and was not adapted or customized.

Thesis: Text Summarization for School Websites

> The NLP task of monolingual text summarization is approached for a custom scenario, on a manually created dataset, see Harms [2024]. This scenario encompasses the summarization of school websites. Four models were applied to generate summaries, one traditional model using extractive summarization and three abstractive models, one of them representing a modern chat-based LLM. Evaluation was conducted qualitatively by scoring relevance, consistency, fluency and coherence of the generated summaries.

Abstractive Monolingual Summarization

The underlying choice of the language model within abstractive summarization approaches is the first modification regarding non-English text summarization in order to improve the performance for other languages. Bert2Bert [Chen et al., 2022] and T5 [Raffel et al., 2020] were tested for both monolingual German use cases,[145] and additionally Pegasus [Zhang et al., 2020a] and mT5 [Xue et al., 2021] were tested for the first German monolingual application scenario [Gottwald, 2023, *p.* 32ff.]. Either multilingual capabilities in a general-purpose performance or their fine-tuning regarding German texts were tested in both approaches.

Lack of Transparency in Model Documentation

A concrete observation regarding the performance of the Pegasus model revealed problems in correctly reproducing German language-specific characters, e.g., an umlaut [Gottwald, 2023, *p.* 40]. This issue can be attributed to the pre-training on purely English datasets, see Zhang et al. [2020a, *p.* 11331ff.], and the dominance of the fine-tuning on the Eng-

[142] Using the MLSUM dataset from Scialom et al. [2020].
[143] Using the WikiLingua dataset from Ladhak et al. [2020] and also see `https://www.wikihow.com/Main-Page/`.
[144] Using the XWikis dataset from Perez-Beltrachini and Lapata [2021].
[145] In the base T5 version from Raffel et al. [2020] used in Gottwald [2023] and the LongT5 variant from Guo et al. [2022] applied in Harms [2024].

lish dataset XSUM [Narayan et al., 2018] in contrast to the second smaller German dataset 10kGNAD[146]. A lack of information regarding the models in Gottwald [2023, *p.* 46] led to problems when processing German datasets.[147] This occurrence showcases the problems in using (large) language models: The lack of information regarding their pre-training and fine-tuning as well as the limited transparency in their documentation, which can all lead to unexpected results and problems regarding the suitability for a specific use case.

Summarization before LLMs

The research in Gottwald [2023] was conducted for the most part at the end of 2022, right around the emergence of LLMs, which were thus not included in this application scenario. The results from this study were inconclusive since the varying structure, layout and vocabulary proved to be a challenge, even for modern pre-LLM models. In the end, the Bert2Bert models generated the most fluent and coherent texts [Gottwald, 2023, *p.* 38], while either Bert2Bert or mT5 produced the highest scores in the quantitative evaluation for the different datasets [Gottwald, 2023, *p.* 43ff.].

School Website Summarization

The second study, processing a dataset of school websites, was conducted with LLMs in mind [Harms, 2024]. In addition to Bert2Bert and T5 as the representatives of abstractive summarization systems and LexRank as the extractive model, the Mixtral 8x7B variant [Jiang et al., 2024] was chosen as the comparison LLM model. The selected LLM was trained and published by Mistral AI, and was, at the time of conducting this research, one of the most performant systems that could be run locally and furthermore also was supposedly capable of processing multilingual prompts.[148] When evaluating these four models on the tasks of summarizing school websites, the performance of the abstractive models Bert2Bert and T5 were only slightly better than LexRank, with drastically worse efficiency in terms of processing time [Harms, 2024, *p.* 16 and *p.* 22ff.], see also Section 8.1. The general-purpose capabilities of neural summarization systems without any additional fine-tuning are thus a limiting factor and their use, especially in contrast to much more efficient extractive systems, is hardly reasonable.

LLM Performance on School Websites

In contrast to comparable results from both extractive and abstractive models from above, the results from the LLM Mixtral surpass all other tested techniques [Harms, 2024, *p.* 23ff.]. The evaluation was conducted on four different criteria, i.e., relevance, consistency, fluency, and

[146] See `https://tblock.github.io/10kGNAD/`.

[147] The models were applied using information registered on Hugging Face, which were labeled as German summarization models and achieved high scores in leaderboards.

[148] See `https://mistral.ai/news/mixtral-of-experts/`.

coherence for this study, in which Mixtral achieved the highest scores for all criteria. The conclusion after qualitatively evaluating the summaries revealed the impressive capabilities of Mixtral, especially for domain-specific datasets, which in this case proved to be particularly challenging for the other non-LLM models mentioned above.

Interpretation of LLM Results

The immense data LLMs such as Mixtral are trained on surpass even BERT-based models. Their adaptability to specialized use cases for which no training datasets for fine-tuning exists, is one strong argument in favor of LLMs in use cases like the application presented here, processing school websites with diverse layouts and contents. However, a small limitation was recognized regarding the criteria of relevance and consistency, which received slightly lower scores compared to fluency and coherence. This observation "[...] points to occasional challenges in accurately reproducing the identified important information within the summaries." [Harms, 2024, *p.* 25]. Still, even these lower scores surpass the scores achieved by all other models considerably.

Multilingual Adaptations

The multilingual text summarization approach from Dal Cin [2025] required few adaptations due to the explicit selection of models with multilingual capabilities. For the extractive approaches, the adaptations were rather minuscule. In the multilingual news summarization scenario for German texts, the parameters of the applied LexRank model were set to allow one more German sentence when compared to French and English due to observations in a manual evaluation, which showed that two sentences are ideal for English and French, whereas for German the parameter is set to three sentences [Dal Cin, 2025, *p.* 22]. The other scenarios were processed with the same parameters regardless of language choice. The abstractive model mLongT5 was by design multilingual, see Uthus et al. [2023], and similarly, Mistral NeMo[149], was also published with multilingual features in mind.

Multilingual Quantitative Evaluation

A more comprehensive evaluation was conducted for five distinct application scenarios in the multilingual summarization scenario, including both quantitative and qualitative evaluation for all scenarios and all respective datasets. The quantitative results present a clear picture for the news summarization scenario, which shows the highest scores across all models and all languages for the English application on the CNN/Daily-

[149] See `https://mistral.ai/news/mistral-nemo/`.

Mail dataset [Dal Cin, 2025, *p.* 31].[150] The limitations for ROUGE notwithstanding, see above in Section 7.2.2, the scores achieved for CNN/DailyMail show the distinct dominance for the most widely used dataset, independent of which model was applied, for both abstractive and extractive summarization techniques. Surely contributing to this factor is the prevalence of the CNN/DailyMail dataset in summarization research and potentially the fact that during training and implementation of the models by the original authors of each model, the performance for this dataset may have been considered. A second scenario was tested similarly, the summarization of scientific publications, which confirm the findings from the first scenario, i.e., the higher performance of all models for the English dataset in comparison to the French dataset.[151]

Multilingual Qualitative Evaluation

The qualitative evaluation is based on the analysis of samples taken from the produced summaries for each of the three tested models, and scored according to five criteria, which include *style* and *use*, see Section 7.2.2, and more categories, including a focus on grammar and lexical choices on the one hand, and structure and coherence on the other hand. Factual correctness, lead bias and finally information dispersion were analyzed, too. The latter criterion deserves a brief explanation, since it presents a specific perspective regarding the relationship between the original text and the produced summary. Information dispersion, as shown in Goel et al. [2021], represents the degree to which the summary captures information gathered evenly from all parts of the source text. The metric is thus somewhat aligned with lead bias, but whereas lead bias only represents the degree to which all information in the summary is taken from the beginning of the source document, information dispersion can capture other biases or uneven distributions as well. The qualitative evaluation is rather small-scale, conducted using a small number of spot-checks, but some observations regarding the performance of different models in various languages and a certain agreement with the evaluations using ROUGE can be noticed [Dal Cin, 2025, *p.* 37ff.].

Language-Dependent Observations

LexRank produces similar scores across the qualitative metrics for the news summarization scenarios in all applied languages [Dal Cin, 2025, *p.* 38]. This observation was to be expected since LexRank is language-independent and only slight adaptations regarding the sentence tokeniz-

[150] The datasets CNN/DailyMail [Dernoncourt et al., 2018] and Newsroom [Grusky et al., 2018] were tested for English, 20 Minuten [Kew et al., 2023] and MLSUM-de [Scialom et al., 2020] for German, as well as MLSUM-fr [Scialom et al., 2020] and OrangeSum [Eddine et al., 2021] for French.

[151] The PubMed [Cohan et al., 2018] dataset for English was contrasted with the TermITH [Bougouin et al., 2016] dataset for French.

ation procedure were required in order to properly distinguish between sentence units. When switching to abstractive summarization, the scores for summaries produced by mLongT5 for English and French summaries were similar, but both surpassed the scores for German summaries [Dal Cin, 2025, *p.* 42ff.]. Problems were noticed in redundancy especially and through a higher rate of factual mistakes and hallucinations in the German summaries, as well as a primarily simple syntactical style.[152]

LLM Performance on Multilingual Data

The results produced by the LLM Mistral NeMo receive significantly higher scores across most metrics in comparison to the other models, and nearly uniformly for all languages. Small limitations were noticed regarding language-specific attributes such as the appropriate articles for German nouns or some rare cases with non-French words used in the French summary, usually in English [Dal Cin, 2025, *p.* 44ff.]. The other scenarios were evaluated in the same manner, and once again, show similar results, i.e., the NeMo model produced the best results across all qualitative metrics with the other models trailing behind except for one single scenario, the abstract generation scenario for French data, which is intended to summarize research publications [Dal Cin, 2025, *p.* 48ff.]. The generated French abstracts include impersonal verb forms, uneven information dispersion and overall problems in processing long-form documents such as research publications. The latter observation is potentially due to the limitations in processing longer documents, not necessarily because context windows were unlimited. Rather, the problem arises from a narrow focus on only certain parts of publications, which leads to low information dispersion. In some cases for this data, the summarization was generated as English text instead of the desired French text, thus showing another limitation of processing non-English texts specifically with LLMs [Dal Cin, 2025, *p.* 50]. Overall, NeMo produced the best summaries across quantitative and qualitative metrics in nearly all scenarios, with the English results surpassing the other languages and not exhibiting the mentioned problem for French research publications.

Quantitative vs. Qualitative Evaluation

In comparing the ROUGE scores and the scores achieved in the qualitative evaluation, the news summarization scenario presents one instance in which the scores do not align [Dal Cin, 2025, *p.* 63]. The qualitative evaluation clearly presents NeMo as the summarization model with the best

[152] The latter aspect might be attributed to the fine-tuning of mLongT5 with the German dataset Klexikon, which uses a simplified version of the source texts as summaries, see Dal Cin [2025, *p.* 42] and Aumiller and Gertz [2022] and more discussion regarding this aspect in Section 7.4.2. Also, an overlap between training data from mLongT5 and the French test datasets might lead to better results in a similar manner [Dal Cin, 2025, *p.* 43].

scores, whereas three datasets (both French datasets and CNN/DailyMail) were scored similarly in the quantitative evaluation across all three models. Another discrepancy regarding ROUGE is observed in the abstract generation scenario for both English and French. In this scenario, the worst ranked model in the qualitative evaluation – LexRank – was ranked higher than NeMo in ROUGE [Dal Cin, 2025, *p.* 34 and *p.* 48ff.]. One factor influencing the discrepancy between both evaluation variants might be the dependence of ROUGE on the length of reference summaries. The link between higher ROUGE scores and longer reference summaries was already presented above in Section 7.2.2.

Now that limitations and restrictions for non-English NLP tasks have been collected in Section 6.2 and displayed through several use cases in this section, more observations regarding other challenges from Section 6.4 will be shown in the next section.

7.4 Other Problems

Of the problems discussed above in the corresponding Section 6.4, all of them predominantly affect LLMs and generative models in general. Reproducibility with NLG, text simplification and text summarization approaches, explainability with even more tasks and efficiency in different forms have all been observed and will be displayed in the following. First, reproducibility will be presented.

7.4.1 Reproducibility

Some parts of this subsection have been inspired by supervised theses, in particular the experimental results, see Dal Cin [2025], Fruth [2022] and Matschat [2022].

In studying reproducibility, this section will align with the previous discussion on lacking metrics for text generation from Section 7.2.2, starting with the NLG use case from Matschat [2022], followed by the text simplification scenario in Fruth [2022] and concluded by the text summarization experiments in Dal Cin [2025].

Reproducibility in NLG

Regarding the NLG application, see details in Section 7.2.2, the best results were achieved in a quantitative and qualitative manner by the GPT-2 models, followed by Markov chain methods and lastly the LSTM-based approach. A limiting factor for these results and their interpretation, however, is the reproducibility of the generated results [Matschat, 2022,

p. 60]. The LSTM model, while producing the worst results across all other criteria, produces stable outputs when provided with the same input text. The results produced by both the traditional models based on Markov chains and the two transformer-based models, both of which are based on GPT-2 with one of them including an additional fine-tuning phase, all deliberately exhibit randomness as a feature. For texts produced by GPT-2, several observations were noted. While the structure of the texts remains consistently similar to the desired biographies, vocabulary and inter-sentence relations differed across generated texts even when provided with the same input. The inter-sentence relations affect the results in two ways, for both text layout and content [Matschat, 2022, *p*. 57ff. and *p*. 60]. The former results in either listed items or a cohesive text, both commonly found in Wikipedia articles, which are emulated by the task. The latter problem concerns topic shifts, which can vary between staying completely on topic for the whole generated text or shifting widely between themes and topics. The structure and length of the results remain the most consistent within this application for the texts produced by the fine-tuned version GPT-2 when compared to the other models [Matschat, 2022, *p*. 60ff.].

Reproducibility in Simplification

A deeper analysis regarding sampling strategies during model training was conducted in Fruth [2022] for the task of text simplification. Experiments using beam sampling, typical sampling as well as random sampling were conducted, before the decision to train the model with a combination of nucleus and top-K sampling was made [Fruth, 2022, *p*. 36]. The attempts at applying different sampling algorithms revealed the importance of parameters such as temperature for random sampling, i.e., setting lower values for the temperature typically resulting in less random and more coherent text and conversely higher values meaning improved creativity with lesser quality overall.[153] Other problems regarding sampling strategies such as random sampling have been noted and studied, e.g., in Meister et al. [2023]. The sampling strategies have been applied in training both models in Fruth [2022]. However, the effect of sampling strategies for the whole system has not been explored in detail in that study, since the effect of different scores and guardrails regarding the simplification and the comparison between two training runs and the pivot model has been the focus in that study, see Fruth [2022], Fruth et al. [2024] as well as Section 7.1.1 and Section 7.3.2 in this thesis.

[153] See, e.g., Holtzman et al. [2020] or `https://huggingface.co/blog/how-to-generate/` for details on the effect of different sampling strategies.

Reproducibility for LLM Sampling

The influence of temperature regarding LLM-generated texts as well as the impact of sampling strategies and other parameters have been studied for reproducibility purposes, e.g., in Blackwell et al. [2024], who noted lower temperature values corresponding to more deterministic responses and higher values increasing creative outputs, similar to the experiments with random sampling in Fruth [2022, *p.* 32ff.]. The same study focuses on temperature-based sampling but also features preliminary experiments [Blackwell et al., 2024, *p.* 5] that show similar results for the final sampling strategy used in Fruth [2022], i.e., nucleus and top-k sampling.

Reproducibility in Summarization

The reproducibility of output generated by LLMs has also been noted for the summarization tasks in Dal Cin [2025] regarding the length of the produced summaries. All approaches were directed through parameter settings or through instructions within the prompt to respect size limitations regarding the different summarization scenarios. The traditional extractive technique LexRank complied with this restriction, whereas both abstractive models, mLongT5 and NeMo, did not adhere to the given rules in different summarization scenarios [Dal Cin, 2025, *p.* 45ff.]. This occurrence aligns with observations in the literature and has already been researched heavily, e.g., early on after the emergence of LLMs in Webson and Pavlick [2022].

Understanding Prompts

The focus in the study by Webson and Pavlick [2022] is not only on reproducibility, but the more general problem of fit between prompt and generated output by LLMs. Therefore, their research question concerns the semantic level and also the questionable capability of LLMs of *understanding* the meaning of what is asked of them through a prompt by the user. Initial studies analyzed the effect of template-based prompts as well as the effect of so-called target words, i.e., labels for classes in a classification task [Webson and Pavlick, 2022, *p.* 2303ff.]. The study of connecting prompts, the meaning behind them and the generated output through LLMs has spawned new fields of study such as instruction tuning. This research area is known as supervised fine-tuning, i.e., the additional training of LLMs using instruction-output pairs, see Peng et al. [2023], or more simply on the user side, prompt engineering, see Sahoo et al. [2024]. The latter area of interest in particular is not without criticism, or at least with a manifold of challenges and problems, see, e.g., Geroimenko [2025] or Zamfirescu-Pereira et al. [2023]. In turning back to the observations in Dal Cin [2025] regarding the missed instructions by the LLM, it can be argued that understanding the prompt, following the instructions given by the prompt and finally being able to repeat the process multiple times can all be subsumed under the challenge of reproducibility.

Reproducibility Experiments

The NLG task from Matschat [2022] and the text summarization task from Dal Cin [2025] and two other text summarization use cases, Gottwald [2023] and Harms [2024], will be explicitly analyzed regarding reproducibility, see below in Section 8.2 for the methodology and results of this case analysis. As has been mentioned in Section 4.2, in Section 5.2 and also in Table 5.1, the ability to reproduce a result given a certain input can be a critical requirement and should be regarded when choosing a method, even when the quality of the results would deem one model as the best-performing technique. However, even when the quality of the produced output meets the desired goal, although the output may differ depending on the model applied as has been shown in this subsection, it may become necessary to investigate the causes of why a certain text or patterns within the outputs have been produced. Therefore, explainability will be discussed based on text summarization and other tasks in the following.

7.4.2 Explainability

Some parts of this subsection have been inspired by supervised theses, in particular the experimental results, see Dal Cin [2025], Fruth [2022], Hebeis [2025], Matschat [2022] and Möglich [2025].

Explainability due to Training Datasets

Previously introduced in Section 6.4.2, the research field of explainable artificial intelligence has seen immense growth and interest due to the rise of neural networks. The same has been the case for NLP, due to the prevalence of LLMs in today's research in that realm. Current LLMs are seen as black boxes [Madsen et al., 2022], but interpretations for local explanations, i.e., the reasoning behind a single decision or generation by a model, see Section 6.4.2 for more details, can still be attempted through various means. One such aspect can be analyzed by looking at the data the applied language model is trained on. For the multilingual summarization task from Dal Cin [2025] and the German summaries in particular, it was observed that several output sentences by the abstractive summarization model mLongT5 are often short and syntactically simple, which might be due to the training data for the German portion of the multilingual mLongT5 model [Dal Cin, 2025, *p.* 42]. The training data of the fine-tuning that was applied on the mLongT5 model[154] used for this task concerns the sumstew[155] dataset. A major part of the German data within

[154] See Uthus et al. [2023] for the main model architecture and `https://huggingface.co/Joemgu/mlong-t5-large-sumstew/` for the concrete model.

[155] See `https://huggingface.co/datasets/Joemgu/sumstew/`.

the sumstew dataset is the Klexikon dataset, a joint summarization and simplification dataset [Aumiller and Gertz, 2022]. The simplifications included in Klexikon might explain the overall simpler sentences produced by the mLongT5 model for German. The other languages in this summarization scenario did not exhibit this behavior, since, potentially, the other datasets lack these simplified data samples [Dal Cin, 2025, *p.* 23f.].

Degree of Extractivity

Uncertainty regarding the manner of extractive and abstractive summarization is another problem concerning explainability, especially for the mLongT5 model. The degree of extractivity can be calculated as the "[...] the average length of the extractive fragment to which each word in the summary belongs" [Grusky et al., 2018, *p.* 712], with some threshold defining summaries being mostly abstractive, mostly extractive or mixed. When calculating this score, e.g., for the German datasets, a score indicating mostly abstractive summaries is observed. However, after examination of the results, small grammatical mistakes and spelling errors led to small variances compared to the source texts, which in turn led to decreased extractivity scores [Dal Cin, 2025, *p.* 33f.]. After correcting these errors and once again calculating this score, the degree of extractivity changes considerably, leading to a mixed summary, i.e., between extractive and abstractive. The degree of extractivity is just one aspect of evaluating summarization approaches, in particular for the performance of abstractive models and the expected abstractive summaries produced by them. A conclusive evaluation for this feature of extractivity is only possible after a careful analysis of the metric itself and a qualitative look at the generated data, due to potential problems as noted in this study on German texts.

Generic Prompts

Another observation concerns prompt-based LLMs, in this case Mistral NeMo. To preserve comparability across all models, i.e., the extractive models as well as the other neural network-based model mLongT5, which was specifically fine-tuned for summarization purposes, the prompts were kept mostly generic for the LLM [Dal Cin, 2025, *p.* 24f.]. Except for two parameters, the prompts remained consistent across all five summarization scenarios. One of two remaining parameters was the language, meaning that the prompt was also written in the language that was used in the source document and the desired summary.[156] The second exception regards the length of the summary, which was specified with respect to the intended usage scenario since the other models could be similarly

[156] One such example in this manner reads: "Fassen Sie den folgenden Text auf Deutsch zusammen", or translated: "Summarize the following text in German" [Dal Cin, 2025, *p.* 78].

instructed to only produce output with a given length.[157] Apart from these usage-dependent parameters, the prompts remain generic, without any further information regarding summarization parameters such as style, use, context or others, see Section 7.2.2 for a discussion on these factors in summarization evaluation. However, smaller experiments have shown that prompts adapted to the use case of the summary are helpful in providing more context for the task and enable the LLM to provide more purposeful and sensible summaries [Dal Cin, 2025, *p.* 63].

Impact of Fine-Tuning

The NLG task from Matschat [2022] was already presented in this chapter regarding the unstable nature of texts produced by NLG systems in Section 7.4.1. While this attribute of the produced texts also falls within the scope of explainability or rather lack of explainability, another observation deserves to be mentioned here. The two applied GPT-2 models in the NLG task, see Matschat [2022, *p.* 32f.], offer an opportunity to differentiate explainability in terms of fine-tuning. The first GPT-2 model is used as-is, thus no additional fine-tuning regarding the NLG task was conducted, whereas the second GPT-2 model was fine-tuned in order to produce text that is similar to the desired output. While both GPT-2 models surpass the other tested approaches – LSTM and Markov chain models in this case – regarding fluency, grammar and thematic focus, they remain rather unstable, as was discussed in Section 7.4.1. Furthermore, after manually comparing the quantitative results with a qualitative study, it was found that the fine-tuned GPT-2 model not only achieved the highest BERTScore, beating the generic GPT-2 model, but also produced the most fitting output format that is similar to the desired layout [Matschat, 2022, *p.* 62], surpassing once again the generic GPT-2 model. Therefore, when attempting a post-hoc explanation, see Section 6.4.2 regarding explainability dimensions, the fine-tuning in this case did not change the performance regarding grammar, vocabulary and other textual features, but improved upon the generic GPT-2 model regarding the fit for this specific task.

Impact of Reinforcement Learning

The reinforcement learning model used in the SCST approach applied in Fruth [2022] in order to simplify German texts on a paragraph-level provides another perspective for explainability. More details regarding the study on this text simplification task were already presented in Sections 7.1.1, 7.2.2 and 7.3.2. In the latter two discussions, the rewards and scoring procedure within the reinforcement learning framework were described, which not only required extensive adaptations due to the trans-

[157] An example for this exception would be: "Summarize this text in no more than 8 sentences" [Dal Cin, 2025, *p.* 78].

fer from English to German, but also enabled a new means to approach explainability for the simplification procedure. During the manual qualitative evaluation, the impact of certain rewards and scoring functions have become apparent through direct links to the generated simplifications. One such link becomes obvious when analyzing the lexical simplifications. Word length reductions are rated highly by the score focusing on lexical complexity, which in turn encourages the text generator to split complex German compound words in two [Fruth, 2022, *p.* 51f.]. As a consequence and side effect, however, semantic modifications were produced, sometimes altering the meaning from the original texts.[158]

Explainable Hallucinations

Other problems such as structural changes and grammaticality are common across the evaluated simplification models but are harder to directly link to certain scores or training procedures [Fruth, 2022, *p.* 52ff.]. One other observation, regarding hallucinations, as discussed in Section 7.1.1 in terms of applied measures to prevent them from appearing, is noteworthy in this regard. While the score measuring hallucinations and, in turn, encouraging the text generator to not produce them, was applied for the two training runs of the k-SCST model in a comparable manner, they both diverge during training [Fruth, 2022, *p.* 43]. The main model *GUTS-2*, the better model overall, did not improve upon the hallucination score, likely due to adjusted training settings and hyperparameters. In contrast, the first model *GUTS-1*, while trailing behind in most other evaluated categories, was improved drastically during training in the hallucination score. These findings were confirmed during the qualitative evaluations of the results, which showed more hallucinations regarding added names for GUTS-2 than for GUTS-1 [Fruth, 2022, *p.* 55f.]. Conversely, the pivot model[159] added entirely new, hallucinated sentences, which did not appear for the two newly trained models [Fruth, 2022, *p.* 56] and is potentially due to the less complex hallucination detection mechanism from the original paper from Laban et al. [2021].

Explainable Question Answering

The issue of explainability was also observed for the question answering task from Möglich [2025]. The transformer-based methods, the BERT-based system as well as the LLM Mistral Large[160], were evaluated as having

[158] These modifications can be beneficial for text simplification purposes, e.g., replacing "Schlossräume" (castle rooms) with "Räume" (rooms), but can also be undesirable, e.g., replacing "Präsidentenflugzeug" (presidential plane) with "Präsident" (president) [Fruth, 2022, *p.* 51f.].

[159] The pivot model translates the German source text into English, simplifies the translated text by applying the original approach from Laban et al. [2021], and translates the simplified English text back into German, see Section 7.3.2.

[160] See `https://mistral.ai/news/mistral-large-2407/`.

less comprehensible and explainable results in the qualitative evaluation [Möglich, 2025, *p.* 58]. The traditional model concerns a simple retrieval system based on TF-IDF scores between query and dataset. The match between words within the query and the retrieved texts as part of the evaluation could be directly linked to explain both irrelevant and relevant results. Through these direct links, the issue of compound nouns in German as cause of several errors could be identified [Möglich, 2025, *p.* 34]. A new task regarding entity matching presents more perspectives regarding explainability.

Thesis: Entity Matching

> The NLP task of entity matching is approached through the reconciliation of two large authority databases, see Hebeis [2025]. This scenario is focused on the matching of person records between two data sources, one traditional institutional authority file and one open collaborative dataset. Seven models were applied on the entity matching task, six of which are traditional methods and the remaining approach a more modern technique, with each model being applied using different preprocessing procedures. The evaluation was conducted quantitatively using f1-score as well as runtime analysis and qualitatively through an analysis of incorrectly labeled data.

Entity Matching Task

This entity matching use case is also applicable to the explainability dimension of NLP methods, see Hebeis [2025]. Traditional models as well as modern methods were applied in the task of matching person records. Data from the Gemeinsame Normdatei (GND)[161] and WikiData[162], both authority file providers, were crawled, preprocessed and used as training data. The traditional models comprise a range of different classifiers, which each have already been referenced before in this thesis, see mostly in Section 2.2.2 and Section 3.3, and include decision trees, random forests, SVMs, Naive Bayes and logistic regression. XGBoost [Murphy, 2022, *p.* 619f.] is one system applied by Hebeis [2025] that has not been mentioned yet, as an offshoot of boosting classifiers, see Section 2.2.2 for a brief introduction to that type of classification algorithms. The BERT architecture selected for this use case was DITTO [Li et al., 2020], a system based on a sequence-pair classification task and applied with BERT [Devlin, 2018], DistilBERT [Sanh et al., 2019] and RoBERTa [Liu, 2019] as language models. In order to process German

[161] See `https://www.dnb.de/DE/Professionell/Standardisierung/GND/gnd_node.html`.

[162] See `https://www.wikidata.org/`.

texts included in the German authority data, a multilingual DistilBERT variant[163] was used.

Explainable Entity Matching

The evaluation of the entity matching approach was conducted in a quantitative manner by calculating the f1-score and by analyzing the results statistically as well as qualitatively. Both BERT-based methods and traditional models were able to produce strong results in terms of general feasibility for the intended use case [Hebeis, 2025, *p.* 54]. A distinction becomes apparent when trying to approach explainability for the applied models. The traditional machine learning-based approaches were in many cases able to correctly match corresponding persons between the different authority files and even if they failed, sparse data from one of the two data sources was to be blamed [Hebeis, 2025, *p.* 46ff.]. This missing information could be identified accordingly, e.g., missing information on a person's birth or profession. However, qualitative evaluation revealed flaws for the BERT-based method for people with rich metadata fields that were correctly identified by the traditional machine learning classifiers [Hebeis, 2025, *p.* 47ff.]. The reasons for this behavior remain opaque due to the black-box nature of BERT-based classifiers.

Additional Information

As a contrasting point to the last observation, contextual and latent semantic information could only be retrieved with the BERT-based models, which seems natural, since traditional machine learning classifiers lack the capability to extract such complex or hidden information [Hebeis, 2025, *p.* 50f.]. Regarding the input data, the requirement of a schema was analyzed. While the dependence on a dedicated schema mapping as a preprocessing requisite for mapping the metadata fields of the two authority files seems like a clear disadvantage for the machine learning classifiers, even the BERT-based models produced better results when provided with the schema mapping than when applied without the schema [Hebeis, 2025, *p.* 52]. The improved performance using the schema mapping, as well as efficiency trade-offs, see below in Section 7.4.3, further confirms the need for properly analyzing the results of different models with various parameter configurations and also corroborates the continued relevance of traditional models for this classification task.

Lessons Learned

A summarizing overview of this subsection presents one overlap across all use cases, namely the need for a qualitative evaluation in order to examine and interpret the results, which in turn enables a small-scale but focused rationale of single generations and generated outputs, or patterns

[163] See `https://huggingface.co/distilbert/distilbert-base-multilingual-cased/`.

therein. These results can be seen as a post-hoc explainability study but are not to be considered as equal to dedicated analysis regarding XAI, as described, e.g., by Dwivedi et al. [2023]. To fully attempt a study regarding explainability in modern NLP methods, the pipeline or even the machine learning lifecycle needs to be adapted to include an understanding phase, comprising training and quality assurance of a model, as well as an explaining phase, including a real-world deployment of the model or study of data applied to the model [Dwivedi et al., 2023, *p.* 3ff.]. The focus of the evaluations mentioned in this subsection is not on providing a comparable wide-reaching study on explainability, but rather on smaller-scale interpretations for the applications in their respective use case, i.e., NLG, text simplification and text summarization. Still, even smaller observations, e.g., the impact of certain contextual information within prompts or an analysis of training datasets that were used in fine-tuning, can show glimpses in order to approach explainability to some degree, and to link observations to certain procedures and model features or parameters.

Reproducibility and explainability are both dimensions, which have been observed in the use cases through evaluation, i.e., by analyzing the produced results and generated texts, both quantitatively and qualitatively. In contrast, the next dimension, efficiency, is naturally observed during training and execution by simply tracking time or other facets such as the expended energy and can be simply compared afterwards.

7.4.3 Efficiency

Some parts of this subsection have been inspired by supervised theses, in particular the experimental results, see Gottwald [2023], Harms [2024], Hebeis [2025], Matschat [2022], Möglich [2025] and Pulver [2023].

Runtime & Energy Consumption

Efficiency as a challenge within text processing tasks can comprise not only time-based metrics, but also other facets such as energy consumption. Whereas Section 8.1 will also address the expended energy in addition to runtime, this subsection will focus on further temporal aspects, which can be grouped into two classes: On the one hand the time that was needed to train models, training time for short, and on the other hand the time required when executing the models or techniques to produce the indented output, inference for short. The first observations regarding the runtime concern text summarization tasks.

Efficiency in Summarization

Two text summarization approaches, one focusing on a comparative analysis across several different domain datasets by Gottwald [2023], introduced in Section 7.2.2, and the other on both abstractive and extractive summarization of school websites by Harms [2024], see Section 7.3.2, discuss the efficiency regarding the applied summarization models. In the former approach, the T5 model [Raffel et al., 2020] was observed as being extremely inefficient in summarizing legal texts from the LegalSum dataset [Glaser et al., 2021], requiring nearly one minute per summarized text on a common laptop [Gottwald, 2023, *p.* 43], which far exceeded the baseline models LexRank [Erkan and Radev, 2004] and TextRank [Mihalcea and Tarau, 2004]. The second text summarization approach paints a similar picture. Both neural network-based approaches, Bert2Bert [Chen et al., 2022] and LongT5 [Guo et al., 2022], take at least ten times as long per summarization in comparison to the extractive approach LexRank [Harms, 2024, *p.* 16]. In the latter approach, prompt-based LLMs were also tested, however on a smaller scale compared to the other models. In this brief experiment, the summarization produced by the Mixtral model [Jiang et al., 2024] required over 100 seconds per summarization.[164]

Efficiency in NLG

The text generation task from Matschat [2022], as already described in Section 7.2.2, discusses efficiency regarding two aspects. First, the applied models in the NLG task can be grouped into models requiring a dedicated training phase and into models that can be used as-is without training. The LSTM model as well as the fine-tuned GPT-2 approach both included dedicated training phases, with correspondingly increased hardware requirements, see Matschat [2022, *p.* 39f.]. The other three applied models did not require a separate training phase, i.e., on the one hand the generic GPT-2 without fine-tuning and on the other hand both Markov chain approaches. As for the second aspect regarding efficiency for this NLG task, the models were characterized briefly in the interpretation of the results. The LSTM model was noted as the worst-performing model, not only in terms of quality of the generated texts, but also in terms of training and processing time when generating the results [Matschat, 2022, *p.* 61]. In contrast, the Markov chain models provide decent results with drastically lower resource requirements, while GPT-2 can be recommended due to the overall best results in both variants. The two GPT-2 versions are only slightly disparate, since only the fine-tuned version was able to reproduce the desired layout in the outputs, see Section 7.4.1, while, however, also requiring the additional fine-tuning phase [Matschat, 2022, *p.* 62]. The

[164] A more detailed analysis of efficiency for the runtime as well as energy consumption for both text summarization use cases will be presented below in Section 8.1.

generic GPT-2 approach did not perform as well regarding the desired layout but otherwise produced results of a similarly high quality.[165]

Efficiency in Other Tasks

The question answering scenario introduced in Section 7.1.2 also noted the drastically more efficient performance of the TF-IDF model in contrast to the BERT-based model [Möglich, 2025, *p.* 58]. Furthermore, efficiency in both runtime and required resources were also noted as advantages of TF-IDF in contrast to the evaluated modern methods, in this case Mistral Large and the BERT-based system [Möglich, 2025, *p.* 53]. As part of the entity matching scenario presented in Section 7.4.2 from Hebeis [2025], efficiency was also evaluated, especially regarding the training process. The training for the best-performing machine learning classifiers, here random forest and XGBoost, was measured between 0.33 and 3.19 seconds, whereas the fine-tuning of the BERT-based models are at least an order of magnitude slower, taking between 47.20 and 169.58 seconds, see Hebeis [2025, *p.* 43f.].

Impact of Dataset Parameters for Entity Matching

Another dimension regarding efficiency can be approached by analyzing the required dataset size. For the entity matching study, different subsets of the training data were used to study the impact of the dataset size. In this case, between 1,000 and 10,000 record pairs were used to train each of the classifiers, with increments of 1,000, thus ten different dataset sizes were tested [Hebeis, 2025, *p.* 39f.]. The analysis of dataset sizes reveals the dependence of BERT-based models on a certain minimum dataset size due to performance drops when applying smaller datasets, while the top performing machine learning classifiers instead perform drastically better compared to modern systems even with the smallest datasets and almost consistently outperform the BERT-based model [Hebeis, 2025, *p.* 44f.]. While the performance of BERT-based models converges towards the performance of the best machine learning classifiers with the largest datasets, the dataset used here might still be too small to fully capture the capabilities of neural networks.[166] Apart from size, the type and structuredness of data, as always for any NLP task, plays a role as well. When purely focusing on tabular authority data that is highly structured, the machine learning classifiers beat the more complex and computationally more intensive BERT-based models [Hebeis, 2025, *p.* 43], which was also confirmed by Mudgal et al. [2018, *p.* 27]. It is noteworthy that regardless of

[165] As with the two summarization methods from above in this subsection, the NLG approach will be analyzed in both runtime and energy dimensions in more detail in Section 8.1.

[166] Similar studies regarding the required dataset size have been conducted, see, e.g., Prusa et al. [2015], Nguyen et al. [2021] or Zhao and Chen [2022].

model type, training using larger datasets also increases the training time accordingly, which might be obvious, but should still be stated explicitly.

Efficiency in Information Extraction

The already presented approach on information extraction from Pulver [2023] in Section 7.3.1 was also tested regarding efficiency for both processing time and dataset size. Two models are the clear leading approaches in this use case: On the one hand a traditional Naive Bayes classifier and on the other hand the FastText classifier [Bojanowski et al., 2017], itself an extension of the skip-gram model based on word2vec [Mikolov et al., 2013]. Both produce comparable results in terms of accuracy on the business type identification task, with a small lead for the FastText classifier, while the efficiency evaluation regarding runtime shows a small advantage for the Naive Bayes classifier, with the other two tested classifiers, SVM and decision trees, trailing behind in both dimensions [Pulver, 2023, *p.* 44ff.]. The two leading classifiers are presented with three dimensions in Figure 7.2, i.e., accuracy, performance and processed dataset size.

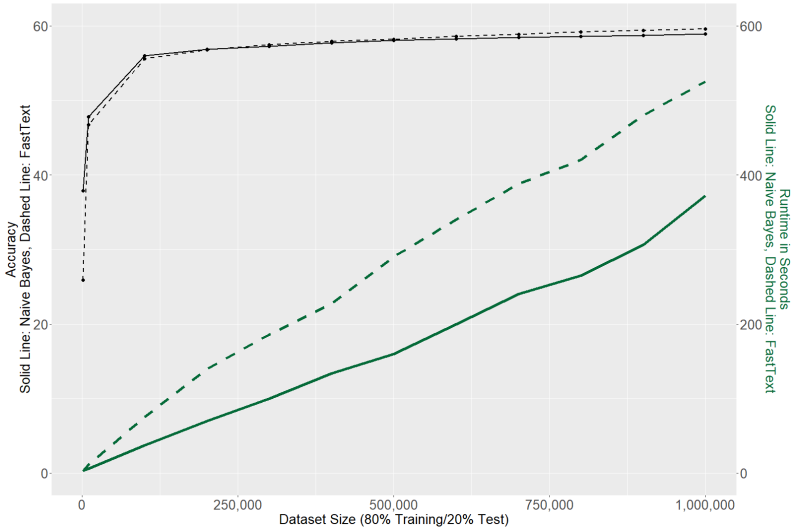


Figure 7.2: Quantitative evaluation results from the information extraction task on business types. Accuracy (in black) and runtime (in green) of Naive Bayes and FastText classifiers are shown, in addition to the dataset size. Adapted from Pulver [2023].

Comparative Analysis

When considering the dataset size, Naive Bayes outperforms FastText in terms of accuracy with the smallest dataset split, while FastText quickly catches up with increasing dataset sizes and surpasses the Naive Bayes

performance slightly for the full dataset [Pulver, 2023, *p.* 61f.]. Naive Bayes and FastText classifiers show a similar trend in terms of runtime, albeit with advantages for the former approach.[167] Therefore, while the traditional model is the leading model in the efficiency evaluation by a slight margin, the FastText classifier performs best in case of accuracy for the larger dataset split. Once again, a use case analysis is advisable for similar scenarios since only an evaluation with multiple distinct parameters can showcase the comparable nature of two markedly different models that perform remarkably similarly.

Supporting Experiments

Some of the use cases presented in this subsection will be more closely inspected regarding efficiency below in Section 8.1. Similarly, reproducibility, as already discussed in Section 7.4.1, will also be analyzed for some of the use cases in Section 8.2. Both runtime in Section 8.1.1 and energy consumption in Section 8.1.2 will be analyzed separately with a quantitative study for tasks in NLG and text summarization and will support the findings from this subsection.

Now that a wide-ranging selection of use cases, with observations, evaluations, both quantitative and qualitative, and corresponding restrictions and limitations have been presented, a collection of benefits available by using traditional models can be gathered. The benefits will be compiled next, in Chapter 8. The competitive nature of traditional models in certain use cases, as affirmation of research question ❷, can be confirmed for the entity matching task discussed in Section 7.4.2, for the information extraction task introduced in Section 7.3.1 and lastly for the text classification task detailed in Section 7.3.2, while other tasks such as simplification and summarization tasks are applied more successfully by LLM-based methods.

[167] The SVM model required nearly 2,000 seconds and the decision tree model took over 7,500 seconds when processing the largest possible dataset respectively, i.e., trailing behind both Naives Bayes and FastText techniques [Pulver, 2023, *p.* 63f.].

8 Benefits of Using Traditional Approaches

The application of traditional approaches on current NLP tasks, while not completely missing in current literature, see Section 4.1 and Jegan and Henrich [2025a], has been relegated to a minuscule weight when compared to the commanding significance of LLMs in recent years. The performance in many tasks is indeed impressive, but problems are manifold, as has been presented in Chapter 6 and further in the preceding Chapter 7 by showcasing use cases with corresponding observations for all mentioned challenges. In this chapter, a different perspective will be applied, i.e., not drawbacks of modern methods but rather benefits and advantages by explicitly choosing traditional models as replacement or complement to modern methods. Therefore, research question 3 – concerning the benefits provided by using traditional approaches – will be directly answered in this chapter.

The benefits, features and traits of traditional models, as they were first briefly laid out in Section 2.5, will be analyzed through the lens of some of the use cases already presented above in Chapter 7. This analysis will culminate in a renewed attempt at another definition of traditional techniques in Section 8.6. In order to do so, a selection of use cases will be thoroughly studied regarding the following characteristics: Efficiency in Section 8.1 both in terms of runtime in Section 8.1.1 and energy consumption in Section 8.1.2, reproducibility in Section 8.2, ease of use and documentation in Section 8.3, explainability in Section 8.4 and the applicability of hybrid approaches in Section 8.5. Whereas some of these sections will be focusing on quantitative evaluations, such as both variants of efficiency in Section 8.1.1 and Section 8.1.2, others will be evaluated in a more qualitative and interpretative manner, e.g., Section 8.4 and Section 8.5. Efficiency as a characteristic of use cases regarding traditional models will be discussed first in Section 8.1.

8.1 Efficiency

Limitations of Efficiency Analysis

The goal of the efficiency analysis was to comprehensively analyze all NLP applications, which were presented above in Chapter 7. It became apparent quickly that not all methods could be analyzed in a streamlined, structured and uniform manner, due to several reasons. The major problems noticed in this study were the setup of development environments and the execution of the provided scripts. Reasons for these issues include documentation flaws, compatibility issues with newer systems, inconsistent package versioning and more. Additional time constraints did not allow a comprehensive analysis of all use cases from above, and therefore only a selection of the discussed applications can be presented in this section. The same problems also affect the subsequent Section 8.2.

Runtime vs. Energy

Efficiency as a feature of NLP systems can be interpreted in several different ways. The first concerns the time spent processing the input data to create the desired output. Other related aspects of language models, such as the training of these models, are relevant of course, but will not be analyzed in this section, see Kaplan et al. [2020] or Naveed et al. [2025] for more details. Instead, inference, meaning the execution of the models, techniques or approaches, either run locally or applied remotely via API[168] will be the focus of Section 8.1.1. In addition to an analysis of runtime, the second dimension regarding efficiency concerns energy consumption, see Section 8.1.2, once again focused on measurements during execution, and explicitly not on training or prior steps in preparing the models. While both dimensions certainly have an interlocking relationship, meaning longer runtime usually resulting in higher energy consumption, a direct correspondence is not necessarily the case. Energy consumption can vary due to the utilization of CPUs or GPUs at capacity, i.e., a hardware component at full load will consume more energy than at a lower capacity, and vice versa. More details will be presented next.

[168] Local analysis of energy and runtime will be presented in detail, whereas metadata of remote executions of models via API requests, such as the OpenAI platform for executing GPT models, see `https://platform.openai.com/`, is not available to the same extent. This lack of transparency of remotely hosted commercial models, often closed source and with fewer insights into parametrization or other details, was already discussed in Section 4.2.

The following NLP tasks were surveyed in this section:

- Natural language generation, as implemented in Matschat [2022] and described earlier in Section 7.2.2 and also in Sections 7.4.1 to 7.4.3.
- Text summarization of German domain corpora, as implemented in Gottwald [2023] and described earlier in Section 7.2.2 and also in Sections 7.3.2 and 7.4.3.
- Text summarization of German school websites, as implemented in Harms [2024] and described earlier in Section 7.3.2 and also in Section 7.4.3.
- Text summarization of multilingual datasets, as implemented in Dal Cin [2025] and described earlier in Section 7.1.2 and also in Sections 7.2.2, 7.3.2, 7.4.1 and 7.4.2.

Brief notes regarding the respective task setup and evaluation methodology for each application scenario will be given in the following.

Details for NLG Use Case

The measurements for the NLG task presented in Tables 8.1 and 8.5 concern the generation of text based on source texts. 500 texts were generated for each model, as done in the original implementation in Matschat [2022, *p.* 40f.]. The applied models comprise two Markov chains [Jurafsky and Martin, 2024, *p.* 369f.], one provided with longer input source text and one with just a single word, as was noted in literature research regarding Markov chains for NLG purposes, see Matschat [2022, *p.* 30]. In addition, one LSTM-based system [Hochreiter and Schmidhuber, 1997] and lastly two GPT-2 approaches [Radford et al., 2019] were used, one being a generic model and the other a custom GPT-2 model, fine-tuned for this NLG task.

Details for Summarization Use Case on Domain Corpora

The measurements for the summarization task detailed in Tables 8.2 and 8.6 present the summarization of texts for German domain corpora. While several datasets were tested, experiments regarding efficiency showed comparable results for all four datasets.[169] Therefore, only the results for LegalSum are presented here, with a selection of 1,000 processed texts from this dataset. In terms of applied models, LexRank [Erkan and Radev, 2004] and TextRank [Mihalcea and Tarau, 2004] were tested as traditional models and as baselines, while Bert2Bert [Chen et al., 2022], MT5

[169] The four datasets in this study were LegalSum [Glaser et al., 2021], MLSUM [Scialom et al., 2020], WikiLingua [Ladhak et al., 2020] and XWikis [Perez-Beltrachini and Lapata, 2021], see more details in Section 7.3.2 and Gottwald [2023, *p.* 28ff.].

[Xue et al., 2021], Pegasus [Zhang et al., 2020a] and T5 [Raffel et al., 2020] were chosen as representatives of modern models, see Section 7.3.2 and Gottwald [2023, *p.* 32ff.].

Details for Summarization Use Case on School Websites

The measurements for the next summarization task in Tables 8.3 and 8.7 detail the summarization of texts from school websites in German. The primary dataset used in these experiments concerns school websites that were crawled manually and which was created specifically for this application scenario, see Harms [2024, *p.* 10f.]. Similar to the previous summarization use case, 1,000 summaries were processed by applying four different models. LexRank [Erkan and Radev, 2004] was chosen as traditional extractive model and Bert2Bert [Chen et al., 2022], LongT5 [Guo et al., 2022] as well as Mixtral [Jiang et al., 2024] were applied as representatives of modern neural network-based methods, the latter also being a chat-based LLM.

Details for Summarization Use Case on Multilingual Datasets

The measurements for the last summarization task in Tables 8.4 and 8.8 showcase the summarization of data from different domains and different languages. Four languages were tested, English, French, German and Italian, with datasets concerning data from domains such as emails [Klimt and Yang, 2004], news [Grusky et al., 2018], research publications[170], school materials[171] and websites[172]. Three models were tested here, once again LexRank [Erkan and Radev, 2004] as traditional extractive model, mLongT5 [Uthus et al., 2023] as transformer-based system and finally as chat-based LLM, Mistral NeMo[173].

8.1.1 Runtime

Measuring Process

The runtime was measured by adding timestamps to the implementation, using the timeit[174] library, which is part of the standard Python library set. Since use cases with widely different goals were analyzed, the focus was to measure only the execution of the respective main procedures. The runtime was therefore simply measured by logging timestamps, starting with the main procedure of the use case up until the pre-determined

[170] Acquired from TermITH, a French organization for terminology and indexation of full scientific papers in humanities and social sciences, see `https://www.ortolang.fr/market/corpora/termith/`.

[171] Retrieved by the author of the Master's thesis Dal Cin [2025, *p.* 20ff.] by crawling websites that offer educational materials for students.

[172] The creation of this dataset mirrors the school dataset, see the previous Footnote 171, only using a set of multilingual websites on a number of topics such as tourism or public health, see Dal Cin [2025, *p.* 20ff.].

[173] See `https://mistral.ai/news/mistral-nemo/`.

[174] See `https://docs.python.org/3/library/timeit.html`.

amount of output has been produced. Since many of the text processing tasks presented below took longer than several days to conclude, time constraints did not permit repeated runs of the full executions. To accommodate for this shortcoming, all experiments were run multiple times with smaller datasets.[175] Since the execution of neural networks is in many cases optimized for or at least benefits from the usage of dedicated GPUs, the analysis was conducted by applying the processing on different systems, one system without a dedicated GPU and two systems with a dedicated GPU. The first machine (**S1** in the tables below), which represents an off-the-shelf laptop, was a Lenovo ThinkPad X13 Yoga, with an Intel 11th generation i7-1185G7 CPU and 32GB of Random Access Memory (RAM). The second machine (**S2**) used an Intel 9th generation i7-9700 CPU, 64GB of RAM and an Nvidia GeForce GTX 1050 Ti GPU with 4GB of Video Random Access Memory (VRAM), while the third system (**S3**) included an AMD Ryzen 7 7700 CPU, 64GB of RAM and a Nvidia GeForce RTX 5070 Ti GPU with 16GB of VRAM.

Containerization

The implementations, as listed above in Section 8.1, differed drastically in their project structure, installation demands and generally in getting the processing up and running. To simplify the workflow, all implementations were containerized by using Docker.[176] This Platform as a Service (PaaS) software product streamlines the build, install and execution workflow through one software stack in order to be run on different systems without additional effort in configuring software environments and package dependencies, which allowed the efficiency analysis regardless of applied systems, see the systems specifications from above. All experiments were conducted without any parallelization or batch processing to accurately reflect the efficiency regarding sequential processing. This sequential operation was chosen to enable a direct comparison between methods without giving any unfair advantage to methods with a higher potential for parallelization.[177] Increased processing speeds through GPU offload of neural network-based processing was however applied for the systems that supported it and the modern techniques that were able to be processed in this manner.

[175] The full statistics for these smaller-scale experiments can be retrieved at `https://github.com/robinjegan/dissertation-experiments/blob/main/efficiency-results_v2.xlsx` in the second, third and fourth tab. The results therein confirm the findings from this section.

[176] See `https://www.docker.com/`.

[177] Further consideration of parallelization and batch processing regarding efficiency for traditional and modern methods would be an interesting perspective of future work.

The measured runtimes are presented in Tables 8.1 to 8.4.[178] The NLG task, first introduced in Section 7.2.2 from Matschat [2022], concerns Table 8.1. For the summarization tasks, results for the scenario processing German domain corpora, also introduced in Section 7.2.2 from Gottwald [2023] are detailed in Table 8.2, and likewise for German school websites, introduced in Section 7.3.2 from Harms [2024] in Table 8.3. Finally, the summarization task for multilingual data, introduced in Section 7.1.2 from Dal Cin [2025], is presented in Table 8.4. For the traditional methods in all Tables 8.1 to 8.4, measurements were only conducted for experiments using CPU-only executions, since they do not benefit from the addition of dedicated GPUs to speed up the processing. Thus, for systems **S2** and **S3** values *n/a* are given for all traditional models in the columns representing the GPU-aided experiments. Runtime Results

Model	S1	S2 CPU	S2 GPU	S3 CPU	S3 GPU
Markov	4,984	4,567	n/a	3,355	n/a
Markov (One Word)	2,264	1,977	n/a	1,472	n/a
LSTM	157,303	237,868	ϟ	85,004	85,591
GPT-2	15,827	16,106	ϟ	16,068	3,627
GPT-2 (Fine-tuned)	17,448	20,598	ϟ	15,579	5,349

Table 8.1: Quantitative results of runtime analysis in seconds for the NLG application generating 500 texts. **S1**, **S2** and **S3** denote the different systems used. For details regarding systems specifications, see the beginning of Section 8.1.1. The ϟ symbol denotes problems occurring when applying the older GPU from **S2**, see details in Footnote 178. Lower values across all tables are desired, i.e., for Tables 8.1 to 8.4 lower values represent faster processing runtimes. The same systems and restrictions were observed in all following tables.

For the first task on NLG, presented in Table 8.1, moderate runtime improvements are observed for the traditional models, the Markov models in this case, when compared to the modern systems. The addition of GPU processing, however, drastically improves the performance of the Runtime: NLG

[178] Due to the recent release of the Blackwell GPU architecture of Nvidia, not all applications were able to be executed using the newest GPU hardware in this analysis. For example, the LSTM model in the NLG task is using libraries that are not yet compatible with these newest GPUs, as of November 2025, see `https://github.com/tensorflow/tensorflow/issues/89272/`. In contrast, the older Pascal architecture from system **S2** is already too old to be compatible with some of the transformer-based implementations from Section 8.1. Therefore, these issues prevent the use of an older GPU from **S2** as comparison point in some cases to a more recent GPU from **S3**.

GPT-2 model for **S3**. Combined with the task-leading quality of the generated texts by the GPT-2 models, see the discussion of qualitative results for this use case above in Section 7.2.2, the somewhat comparable runtime of the GPU-aided GPT-2 model in comparison to the Markov models would seem to present the best cost-to-performance balance for this use case. The model trailing behind, LSTM, received the worst scores across all models, by a significant margin. The observed lacking quality of texts by this model shows its inability in processing this task across several evaluation dimensions. Additionally, the use of the GPU did not lead to an increase in processing speed but instead led to a slight decrease potentially due to insufficient optimization efforts for the LSTM implementation. As for **S2**, slightly better performance can be noted for the traditional models – both Markov techniques – in comparison to **S1** but not for **S3**, while the more modern techniques are trailing behind across all tested configurations for **S2**.

Model	S1	S2 CPU	S2 GPU	S3 CPU	S3 GPU
LexRank	564	327	n/a	283	n/a
TextRank	484	331	n/a	242	n/a
T5	7,080	5,983	↯	4,285	2,386
Pegasus	14,940	10,913	↯	8,901	3,653
MT5	7,131	5,920	↯	5,649	3,773
Bert2Bert	6,648	5,850	↯	4,099	2,966

Table 8.2: Quantitative results of runtime analysis in seconds for the task text summarization of German domain corpora generating 1,000 texts. See details on the terminology and iconography in Table 8.1.

Runtime: First Summarization Task

The first summarization task on German domain corpora, presented in Table 8.2, shows more clear runtime differences across the models. The two traditional models, both extractive techniques, LexRank and TextRank, are processed much more quickly than all tested abstractive models, by at least one order of magnitude across all tested systems. In the end, the two abstractive models that scored highest in both runtime analysis and in the qualitative manual evaluation, Bert2Bert and mT5, are either the most efficient abstractive models, in case of Bert2Bert, or only slightly behind, for mT5. The worst performing model in terms of runtime, the Pegasus model, also produced qualitatively inadequate summaries, due to, e.g., problems with German special characters, such as the umlaut [Gottwald, 2023, *p.* 40], see Section 7.3.2. LexRank, while processing slightly

slower than the other traditional model, was observed to produce more coherent texts than TextRank, and thus would provide a drastically more efficient alternative to abstractive models, albeit with worse qualitatively evaluated summaries [Gottwald, 2023, *p.* 45f.]. The three systems produce processing runtimes that correspond to their hardware capabilities, i.e., the most performant **S3** leading in these results, with **S2** trailing behind and **S1** scoring the worst. The addition of the GPU in **S3** leads to a significant processing speedup, ranging from 28% for Bert2Bert up to 59% for Pegasus.

Model	S1	S2 CPU	S2 GPU	S3 CPU	S3 GPU
LexRank	91	99	n/a	63	n/a
BERT	3,049	2,963	ϟ	1,481	387
mLongT5	57,276	24,335	ϟ	18,414	2,209
Mixtral	252,902	124,586	71,435	42,376	11,735

Table 8.3: Quantitative results of runtime analysis in seconds for the task text summarization of German school websites generating 1,000 texts. See details on the terminology and iconography in Table 8.1.

Runtime: Second Summarization Task

For the second summarization task, processing data from school websites, the same observations from the previous Table 8.2 can also be noted in Table 8.3. The traditional model, here LexRank, is drastically more efficient in terms of runtime than competing abstractive models. The differences for the three abstractive models are more pronounced than in Table 8.2. The BERT model produces summaries much quicker when compared to mLongT5 and even more so than Mixtral, at least for **S1**. These observations differ for the other systems, due to more working memory, faster CPU and better cooling without thermal throttling for **S3**. These hardware improvements are noticeable due to better optimizations regarding the available hardware for the locally run LLM Mixtral.[179] The addition of GPU-enabled processing improved the performance for all three applicable abstractive methods by cutting the runtime by at least 72% compared to the CPU-only version, for **S3**. When also factoring in the qualitative evaluation from Harms [2024, *p.* 24f.], the LLM produces the best summaries across several qualitative dimensions. A trade-off regarding runtime and textual quality needs to be met since Mixtral improves

[179] The applied software, LMStudio, see `https://lmstudio.ai/`, offers more detailed hardware settings for locally run LLMs than other transformer-based language models such has BERT or T5.

upon all other systems drastically, while trailing behind in runtime. As for the other abstractive models, BERT and mLongT5 only achieve competitive qualitative scores with LexRank while being much more inefficient in runtime [Harms, 2024, *p.* 21ff.]. As for **S2**, interestingly, Mixtral was processed twice as fast when using the CPU from **S2** when compared to **S1**, which can be explained through higher power draw, as was already observed for **S3**, see Footnote 179. The improvements of the GPU-assisted version of **S2** trail behind the increase from **S3** but nevertheless achieve a decrease in terms of runtime by 42% when compared to the CPU-only version. The reason for this difference lies in the slower and older GPU with less VRAM when compared to **S3**.

Model	S1	S2 CPU	S2 GPU	S3 CPU	S3 GPU
Email-Dataset – English					
LexRank	1.78	1.94	n/a	1.66	n/a
mLongT5	545	349	657	198	60.5
Mistral NeMo	1,446	704	180	418	9.60
News-Dataset – English					
LexRank	1.70	1.78	n/a	1.62	n/a
mLongT5	282	239	241	141	50.0
Mistral NeMo	763	370	152	203	7.94
Papers-Dataset – French					
LexRank	3.32	3.50	n/a	2.48	n/a
mLongT5	1,330	688	655	670	50.1
Mistral NeMo	2,797	1,539	339	853	16.5
School-Dataset – German					
LexRank	3.72	4.27	n/a	3.41	n/a
mLongT5	879	498	499	295	78.1
Mistral NeMo	3,277	1,709	961	994	45.6
Web-Dataset – Italian					
LexRank	1.99	2.15	n/a	1.76	n/a
mLongT5	237	189	194	111	40.2
Mistral NeMo	2,310	1,297	424	689	17.0

Table 8.4: Quantitative results of runtime analysis in seconds for the task text summarization of multilingual data processing different datasets. See details on the terminology and iconography in Table 8.1.

Runtime: Third Summarization Task

The last summarization task, in Table 8.4, processes multilingual data for five different domains. The noted values are all drastically lower than the values for Tables 8.1 to 8.3, which is due to smaller datasets that were

processed in this use case.[180] Still, even for a smaller number of texts, efficiency measurements for the runtime are noteworthy. LexRank, once again, produces task-leading efficiency scores. The abstractive models trail behind, much more drastically in the CPU-only execution runs, while the addition of GPU processing enhances the performance for both mT5 and Mistral NeMo dramatically for **S3**. The improvements for the latter are more noticeable than for the former, due to the same hardware optimization processes noted for the second summarization task from Table 8.3 available for modern LLMs, see Footnote 179, at least for **S3**. The modern LLM Mistral NeMo produces the best summaries when evaluated qualitatively, besting the extractive technique LexRank and the pretrained language model mLongT5 [Dal Cin, 2025, *p.* 60ff.]. However, due to the drastic differences in terms of runtime, LexRank was noted as a cost-effective alternative for the news and website domains in particular [Dal Cin, 2025, *p.* 65f.]. **S2** also improves the runtime performance by adding GPU capabilities, at least for Mistral NeMo. The processing of mLongT5 is roughly comparable regardless of GPU usage or not. One exception to this last observation can be noted for the email dataset with its runtime increasing by 88% for **S2** when using the GPU-assisted configuration compared to the CPU-only variant. One possible explanation might be the type of text included in this dataset, which consists of several emails per text document that is intended to be summarized. The dataset variant used in the efficiency experiments includes all metadata and HTML markup from the email threads, which might cause the mLongT5 model to process these files more slowly when applying the older GPU used in **S2**, since the parallel processing on both CPU and GPU might lead to inefficient overhead computations.

Overall Runtime Results

The traditional models are processed more quickly than modern systems across nearly all observed tasks and systems, see Tables 8.1 to 8.4, the only exception being the GPU-assisted GPT-2 version of **S3** in the NLG-task, see Table 8.1. However, the quality of the output differs, and in certain tasks, e.g., the NLG task from Table 8.1, the quality of the generated texts by the modern systems surpasses the texts produced by traditional models by such a degree that the efficiency advantages cannot compensate for the decreased quality. In other tasks, such as the summarization tasks from Tables 8.3 and 8.4, traditional and extractive models present a cost-effective alternative to more inefficient models, with textual quality that is

[180] The smaller datasets comprise between four and ten texts, due to limitations when creating the manually built datasets from Dal Cin [2025, *p.* 12ff.] and to maintain comparability between the datasets.

trailing behind. In the end, a trade-off between quality and efficiency can be met here with either traditional models or modern methods, depending on which factor reigns supreme. Efficiency will be analyzed according to another dimension in the next section, i.e., energy consumption.

8.1.2 Energy Consumption

Software-Based vs. Physical Power Tracking

Due to the complexity of modern systems, with CPU, GPU, working memory, hard drives and even network interfaces consuming energy, clearly separating the energy expended solely for a concrete NLP processing task is hard to break down. While physical power tracking, using external power meters, see Section 6.4.3, provides the most accurate tracking for a whole system, software-based tools have been shown to correlate strongly with hardware-based versions, see Jay et al. [2023, *p.* 111]. Additionally, software-based power tracking allows the separation of the power draw between hardware components, thus, e.g., the energy expended for the CPU can be analyzed separately from the GPU. Based on the results of the experiments from Jay et al. [2023, *p.* 116], the authors recommend several software-based methods, first and foremost Code Carbon[181] for analyzing the power consumption of approaches applied using Python. Code Carbon not only measures CPU and GPU power draw separately, but also the energy consumed by the working memory and the expended carbon dioxide emissions.[182]

Energy Tracking Results

The energy consumption measurements are presented in Tables 8.5 to 8.8. Three systems were used once again, abbreviated as **S1**, **S2** and **S3**, with the same specifications as mentioned at the beginning of Section 8.1.1. The NLG task, first introduced in Section 7.2.2 and see also Matschat [2022], concerns Table 8.5. For the summarization tasks, results for the scenario processing German domain corpora, introduced in Section 7.2.2 from Gottwald [2023] are detailed in Table 8.6, and likewise for German school websites, introduced in Section 7.3.2 from Harms [2024] in Table 8.7 and finally for multilingual data, introduced in Section 7.1.2

[181] See `https://github.com/mlco2/codecarbon/`.

[182] The methodology of the measured data is referenced at `https://mlco2.github.io/codecarbon/methodology.html`. While the carbon dioxide emissions are only an estimate, the calculations include adjustments regarding the country, for which an average of energy sources are used, i.e., which type of energy production is common in the respective country (such as low-carbon fuels, e.g., solar power or hydroelectricity, but also fossil fuels such as coal or natural gas and even nuclear sources).

from Dal Cin [2025] in Table 8.8 respectively.[183] The measurements in Tables 8.5 to 8.8 represent the total expended energy measured by the respective system for the quoted task, in terms of total energy, i.e., CPU, working memory, and if applicable, GPU, all combined in kWh, meaning kilowatt-hour.[184] The same restrictions regarding the older GPU used in **S2** are once again of concern for the energy consumption analysis, similarly to the runtime analysis from above, see Section 8.1.1.

Model	S1	S2 CPU	S2 GPU	S3 CPU	S3 GPU
Markov	0.027	0.056	n/a	0.05	n/a
Markov (One Word)	0.012	0.024	n/a	0.022	n/a
LSTM	1.049	3.47	ϟ	0.77	1.45
GPT-2	0.087	0.20	ϟ	0.24	0.12
GPT-2 (Fine-tuned)	0.095	0.25	ϟ	0.23	0.15

Table 8.5: Quantitative results of energy tracking in kWh for the NLG application generating 500 texts. Lower values across all tables are desired, i.e., for Tables 8.5 to 8.8 lower values represent lower energy consumption during processing. See details on the terminology and iconography in Table 8.1.

Energy Consumption: NLG

The energy consumption of the models applied on the NLG task aligns well with the runtime measurements from Table 8.1. The traditional Markov models are task-leading, i.e., they consume the least amount of energy by a significant margin. The LSTM model is once again trailing behind all other models, which is natural, due to the longest processing time, as measured in Table 8.1. The impact of adding an additional component, the GPU, to the processing can be observed here as well. While the GPU naturally consumes an additional amount of energy, the speed up due to optimized processing compensates for this additional power draw by decreasing the processing time in so far, that the GPT-2 models using the GPU both consumed less energy than the CPU-only version respectively, because the processing time was also drastically shorter. The

[183] The same restriction for energy tracking as was observed for the runtime analysis from Footnote 178 applies here as well, regarding incompatibility of GPU architectures with the applied implementations from the beginning of Section 8.1.

[184] CodeCarbon provides detailed measurements, i.e., the consumed energy by different hardware components is tracked separately. Due to legibility and to keep the tables from becoming too cumbersome, the combined sum of expended energy was chosen for Tables 8.5 to 8.8. The full statistics with measurements for each component are available at `https://github.com/robinjegan/dissertation-experiments/blob/main/efficiency-results.xlsx`.

LSTM model, conversely, did not improve in processing speed by adding the GPU, which resulted in higher power draw, and almost doubled the power draw compared to the CPU-only version. When comparing the traditional and modern techniques, the same outcome as for the runtime analysis can be observed. Traditional models are much more efficient, however, as noted, the quality of the produced texts trails behind the modern methods, see Section 7.2.2.

Model	S1	S2 CPU	S2 GPU	S3 CPU	S3 GPU
LexRank	0.0038	0.0048	n/a	0.0047	n/a
TextRank	0.0032	0.0048	n/a	0.0041	n/a
T5	0.047	0.087	↯	0.062	0.066
Pegasus	0.10	0.16	↯	0.13	0.11
MT5	0.048	0.086	↯	0.082	0.092
Bert2Bert	0.044	0.085	↯	0.060	0.073

Table 8.6: Quantitative results of energy tracking in kWh for the task text summarization of German domain corpora generating 1,000 texts. See details on the terminology and iconography in Tables 8.1 and 8.5.

Energy Consumption: First Summarization Task

The first summarization task, which processes the German domain corpora datasets, measured in Table 8.6, also aligns well with the runtime measurements from Table 8.2. Traditional models are more energy efficient by roughly one order of magnitude compared to all other measured techniques. The two abstractive models that produced the most appropriate summaries as confirmed by the qualitative evaluation, i.e., Bert2Bert and mT5, include the most efficient neural network-based model, Bert2Bert, with mT5 only trailing slightly behind. The Pegasus model once again performs worse, consuming about double the energy of the other modern models. The two traditional models consume similar amounts of energy, too, with slight advantages for TextRank, whereas LexRank was shown to provide better summaries when evaluated qualitatively although still trailing behind the best modern techniques [Gottwald, 2023, *p.* 45ff.]. Across the systems, energy consumption increases for CPUs with higher power draw, see the increase for all models from **S1** to **S2**, whereas the newer, more efficient architecture of **S3** leads to small energy improvements for the CPU-only configurations compared to **S2** across the board. The addition of the GPU, while improving the runtime for **S3**, see Table 8.2, did not lead to increased energy consumption, due to lower runtimes despite the higher power draw.

Model	S1	S2 CPU	S2 GPU	S3 CPU	S3 GPU
LexRank	0.00060	0.0014	n/a	0.00090	n/a
BERT	0.020	0.043	↯	0.022	0.014
mLongT5	0.38	0.35	↯	0.27	0.10
Mixtral	9.27[(a)]	9.69	5.56	6.45	1.95

Table 8.7: Quantitative results of energy tracking in kWh for the task text summarization of German school websites generating 1,000 texts. See details on the terminology and iconography in Tables 8.1 and 8.5.

Energy Consumption: Second Summarization Task

The second summarization task from Table 8.7, which processes a dataset of school websites, confirms the findings of the prior experiments regarding energy consumption and the runtime analysis from Table 8.3 as well. LexRank as traditional model provides the most efficient summarization processing by far, with all modern methods trailing far behind. A staggered observation can be drawn, i.e., the BERT model being the most efficient neural network-based model, while being much more inefficient than LexRank. mLongT5 and Mixtral provide even worse results when observing the CPU-only versions. The drastically increased power draw in Table 8.7 for Mixtral measured by **S1**, marked as *(a)*, can be explained with a higher wattage enabled by the Mixtral implementation. Here, 120 W are drawn for the CPU, contrasted to only 14 W for the other models in this experiment for **S1**. If this power draw is considered with the other observed values, they fall right in line with the other measurements.[185] The same observation can be confirmed for the other systems when using Mixtral. All modern methods once again profit from the addition of the GPU for **S3**, most noticeably for Mixtral, its energy consumption being cut by 70%, confirming the findings of the speed up for runtime, see Table 8.3. The improved hardware, which for Mixtral resulted in much quicker processing even for the CPU-only version of **S3**, see Table 8.3, also led to a slight advantage in comparison to **S1**, however to a lesser degree, since the improved hardware also led to a higher power draw. In the end, a trade-off between LexRank, being most efficient but displaying purely extractive summaries, and BERT, displaying capable abstractive summaries but also being less efficient, and finally Mixtral, being highly inefficient but producing the best summaries, has to be considered. The findings

[185] Full power measurements for CPU, working memory and GPU, if applicable, are available at `https://github.com/robinjegan/dissertation-experiments/blob/main/efficiency-results.xlsx`.

for **S2** also confirm the results from the runtime analysis in Table 8.3, i.e., the addition of the GPU for the Mixtral model leading to decreased energy consumption overall, while still trailing behind the more modern hardware from **S3**.

Model	S1	S2 CPU	S2 GPU	S3 CPU	S3 GPU
		Email-Dataset – English			
LexRank	0.000002	0.000005	n/a	0.000002	n/a
mLongT5	0.0036	0.0051	0.0096	0.0029	0.0018
Mistral NeMo	0.0096	0.010	0.0026	0.0061	0.00056
		News-Dataset – English			
LexRank	0.000001	0.000003	n/a	0.000001	n/a
mLongT5	0.0019	0.0035	0.0035	0.0020	0.0014
Mistral NeMo	0.0051	0.0054	0.0022	0.0029	0.00044
		Papers-Dataset – French			
LexRank	0.000011	0.000028	n/a	0.000015	n/a
mLongT5	0.0088	0.010	0.0095	0.0098	0.0019
Mistral NeMo	0.011	0.022	0.0049	0.012	0.0012
		School-Dataset – German			
LexRank	0.000014	0.00004	n/a	0.000027	n/a
mLongT5	0.0059	0.0072	0.0073	0.0043	0.0025
Mistral NeMo	0.022	0.025	0.014	0.014	0.0032
		Web-Dataset – Italian			
LexRank	0.000003	0.000009	n/a	0.000005	n/a
mLongT5	0.0016	0.0027	0.0028	0.0016	0.0011
Mistral NeMo	0.015	0.019	0.0062	0.010	0.0011

Table 8.8: Quantitative results of energy tracking in kWh for the task text summarization of multilingual data processing different datasets. See details on the terminology and iconography in Tables 8.1 and 8.5.

Energy Consumption: Third Summarization Task

Lastly, the third summarization task detailed in Table 8.8 regarding energy consumption for multilingual datasets aligns once more with the earlier analysis of the runtime from Table 8.4. Due to smaller datasets, the values are much lower than in the other Tables 8.5 to 8.7, but also confirm the findings therein. The traditional model LexRank consumes the least energy across all models and scenarios in this task, with mLongT5 trailing behind in the CPU-only versions and Mistral NeMo coming in third. However, the addition of the GPU introduces some uncertainty for **S2** and **S3** regarding a clear recommendation purely from the perspective of energy consumption. For **S2**, Mistral NeMo requires more energy than mLongT5 for the school and web scenarios, and vice versa for the other three scenarios. This observation mirrors the results from the runtime analysis for

the GPU-assisted measurements for **S2**, see Table 8.4. As for **S3**, Mistral NeMo trails behind mLongT5 in the school scenario for the GPU-assisted execution runs, while requiring the same amount of energy for the website dataset and surpassing mLongT5 for the other three scenarios. This unclear result does not match with the runtime analysis from Table 8.4, in which the LLM produces results faster than mLongT5 in the GPU-enabled tests across all five datasets. The additional hardware therefore results in faster processing, beating mLongT5 in this dimension, however trailing behind in certain scenarios in terms of energy consumption due to the same factor, i.e., the added hardware, which requires a higher degree of energy due to the optimizations mentioned in Footnote 179. In the end, the quality of the produced summaries is strongest through Mistral NeMo, as mentioned above for the runtime in Section 8.1.1, with niche use cases for LexRank as a cost-effective alternative with accompanying worse quality [Dal Cin, 2025, *p.* 65f.].

Further Perspectives on Efficiency

The benefits of applying traditional methods have now been documented through a quantitative evaluation with measurements regarding runtime and energy consumption as well as an assessment of usefulness and the trade-off between efficiency and quality in the respective use cases. More benefits of traditional models in building less resource demanding systems have also been proposed in the literature such as Jurafsky and Martin [2024, *p.* 515f.], who describe classifiers such as logistic regression, SVM or random forests for coreference resolution, which are more efficient and might serve some special use cases, despite slightly worse results when compared to neural networks in their studies. Whereas this observation regarding one specific use case is certainly not applicable for all NLP tasks, the rising importance and popularity of the research areas *Green AI* [Emer et al., 2025; Schwartz et al., 2020] and in particular *Green NLP* [Hershcovich et al., 2022; Jin et al., 2021, *p.* 3107; Maronikolakis and Schütze, 2021] due to the climate crisis, increasing energy costs and other factors proves the impact of efficiency on NLP in general.

Recent Developments

There have been recent studies and analysis published by LLM providers such as Google, that apparently produce optimized workflows with drastically lower energy consumption for LLM requests.[186] However, even the drastically lower estimates are still higher than the execution of traditional models, see, e.g., the analysis of a further use case also regarding efficiency in both Jegan and Henrich [2025b] and Section 9.3.

[186] See, e.g., Elsworth et al. [2025] and a discussion on the comprehensiveness of this study as well as trustworthiness in `https://arstechnica.com/ai/2025/08/google-says-it-dropped-the-energy-cost-of-ai-queries-by-33x-in-one-year/`.

8.2 Reproducibility

The traditional techniques presented and applied in the use cases from this thesis near unanimously share the feature of reproducibility. The exception are non-deterministic methods such as random forest classifiers, which use a random seed initially [Ho, 1995], and other traditional models that operate with a random seed during initialization. The other techniques discussed here, from extractive summarization models across SVMs for classification tasks and ranging to rule-based approaches for information extraction, all are reproducible when applying the same input. In stark contrast, LLMs in many if not most cases do not reproduce the same output given the same query, despite parameters or strategies that are available to technically achieve reproducibility for LLMs, see Section 6.4.1 for a brief discussion of such measures.

Methodology

To analyze the reproducibility of approaches within NLP, the use cases studied above for efficiency – listed at the beginning of Section 8.1 – were surveyed here once again. The process was straightforward. Each application was initialized several times and the outputs were compared against each other. If the generated output was consistent and remained the same across several initializations, the method was found to be reproducible. In all other cases, the methods were declared not reproducible. While this binary classification seems simple, a more complex and detailed study was out of scope for this thesis. Still, even this simple distinction will suffice for a first observation regarding reproducibility, especially regarding a distinction between traditional and modern neural methods.

Reproducibility: NLG

As was already noted above in Section 7.4.1, reproducibility was already examined in Matschat [2022, *p.* 60]. The results from that study are confirmed here, i.e., only the worst performing model – the LSTM model – generates reproducible texts in this NLG task. All other models, meaning the two Markov models and the two models based on GPT-2, produce varying results that, while remaining similar in length, cannot be noted as reproducible. Thus, even the traditional models, both based on Markov chains, are not reproducible, which is caused by random seeds executed in choosing the next tokens within the Markov chains, see Matschat [2022, *p.* 30 and *p.* 34]. Similarly, the parameter temperature, see Sections 4.2 and 6.4.1, was set for the GPT-2 models to allow a certain degree of creativity for the generated texts, thereby also decreasing reproducibility [Matschat, 2022, *p.* 38].

The first summarization task, see Gottwald [2023], for German domain corpora, is entirely reproducible when choosing the traditional methods.

Reproducibility: First Summarization Task

Both summarization systems, i.e., LexRank and TextRank, produce the same results across multiple execution runs. For this use case, notably all modern models are also reproducible. Thus, all four transformer-based summarization systems, Bert2Bert, MT5, Pegasus and T5, share this characteristic with the traditional models. These four transformer-based models are specifically fine-tuned for text summarization purposes and would not be classified as general-purpose LLMs, therefore impacting reproducibility.

Reproducibility: Second Summarization Task

Once again, the traditional summaries produced by LexRank for the summarization task on German school websites, see Harms [2024], are entirely reproducible. Similarly to the first summarization task from Gottwald [2023], the dedicated summarization models based on neural networks, in this case Bert2Bert and mLongT5, are also reproducible. In contrast, the texts produced by the LLM Mixtral, while appearing similar on the surface and in some cases producing the same output, differ in both syntax and wording for most summaries. Semantically however, the summaries are remarkably similar.

Reproducibility: Third Summarization Task

The summaries produced by the traditional model LexRank for the multilingual summarization task, see Dal Cin [2025], are once more completely reproducible. For mLongT5, the same can be confirmed, thus no changes were noticed across multiple execution runs. The summaries produced by the LLM, Mistral NeMo, did not generate reproducible outputs, i.e., repeated executions yielded widely varying results across all tested datasets. Furthermore, for the summaries produced by both mLongT5 and also Mistral NeMo, the predetermined size limitations of the intended summaries were not always adhered to, see Section 7.4.1. In contrast to the summaries produced for the second summarization task from Harms [2024], the results generated by the LLM in this third task are markedly different, in particular regarding length and syntax. A contributing factor to these differences is certainly the length of the input data, which on the one hand differs drastically depending on the dataset, five of which are tested here, see Sections 7.1.2 and 7.3.2, and on the other hand are in many cases much longer than in the summarization task from Harms [2024]. These variations within this third summarization task along with the uncertainty and lack of reproducibility inherent in common LLMs, see also Section 6.4.1, are causing these reproducibility issues for Mistral NeMo.

Interpretation of Results

A few lessons can be glimpsed from this small case analysis on reproducibility. The traditional models within summarization, i.e., LexRank and TextRank, offer complete reproducibility, whereas the Markov models for

the NLG task fall short in this category. Interestingly, the dedicated neural network-based abstractive summarization models across all three summarization tasks are all reproducible as well. However, this only holds true for applying the same model with the same model weights. When a model is trained, due to attributes such as random initialization at the start or other randomized features in batch-processing and other parts of the pipeline, the exact same weights are usually not reproducible if the model is trained once again, see also the study by Alahmari et al. [2020]. As for LLMs, the more generic, prompt-based LLMs – except for GPT-2 systems from the NLG task, which are a precursor of modern LLMs as known today – do not exhibit reproducible behavior. While parameters such as the temperature of a model, see Section 6.4.1, exist to adapt the degree of creativity and in turn also generate reproducible or at least similar outputs with repeated executions, reproducibility as a feature is much more restricted for LLMs, as shown in these experiments.

After showcasing two dimensions of NLP tasks, efficiency and reproducibility, a less tangible and quantifiable characteristic will be discussed in the next section, i.e., the ease of use and degree of documentation regarding NLP methods.

8.3 Ease of Use & Documentation

The study on efficiency and reproducibility characteristics of traditional and modern models as presented in Sections 8.1 and 8.2 was carried out by executing the approaches listed at the start of Section 8.1. For efficiency, the measurements regarding runtime and energy consumption were noted and for reproducibility, experiments confirm or disprove this characteristic for each model. In contrast, the relevant features of this current section on documentation and the ease of use regarding the applied NLP models were not evaluated in a quantitative manner, but rather by qualitatively reflecting on the available libraries and software and providing examples in this manner. The documentation of available software will be presented first, before the ease of use will be displayed afterwards.

Documentation

As part of the early definition of traditional models in Section 2.5, one benefit of using models with a long history since their introduction is a breadth of documentation, tutorials and widely used packages. The implementation of traditional models for NLP tasks is usually initialized by using libraries that enable the application of said traditional models in a simple manner. One option is larger NLP libraries – Natural Language

ToolKit (NLTK)[187], Stanford CoreNLP[188] or spaCy[189] to name but a few – which are well-documented, have been in use for years and offer preprocessing capabilities as well. In addition to these general-purpose libraries, which can be used to accomplish a wide variety of different NLP tasks, task-specific libraries are also available that enable text classification[190], text summarization[191] and other use cases.

Comprehensibility

Furthermore, the availability of the source code enables a higher degree of transparency and confirmability. This means that the actual procedure of applied processing steps is visible to the developer and problems or observations can be traced back to certain operations within the source code. Examples of such procedures are presented above in Section 7.3 in the adaptation of existing models for English text processing tasks in order to process non-English data, which is possible due to the availability of the source code for the respective NLP tasks. The developers of modern language models, when presented in a research context, often publish the corresponding source as well, but due to the black-box nature of LLMs, a similar degree of transparency and simplicity is not always the case in comparison to traditional models. Additionally, and in stark contrast to the documentation of traditional models, providers of commercial LLMs seldomly publish their data, source code or weights due to trade secrets and the competitive nature of the field, see the discussion on this topic above in Section 2.4.

Ease of Use

The benefits of a well-documented library and availability of the source code itself present further opportunities for the practitioners, who are assigned to solve an NLP problem. Due to the long-standing availability and development history of the already mentioned libraries, a multitude of tutorials and examples can be found, which while varying in quality, offer broad opportunities in approaching a new method.[192] Not only libraries, but also the adaptability is enhanced through traditional models, which in many cases feature only a small amount of code[193] that can be com-

[187] See Bird et al. [2009] and `https://www.nltk.org/`.

[188] See Manning et al. [2014] and `https://stanfordnlp.github.io/CoreNLP/`.

[189] See Honnibal et al. [2020] and `https://spacy.io/`.

[190] See, e.g., `https://scikit-learn.org/stable/auto_examples/text/plot_document_classification_20newsgroups.html`.

[191] See, e.g., `https://github.com/miso-belica/sumy/`.

[192] Examples of high-quality tutorials are `https://scikit-learn.org/stable/auto_examples/text/index.html` and `https://www.nltk.org/book/ch06.html`, both for classifying documents.

[193] While the number of lines of code alone is not evidence alone that guarantees the quality of the software, a more concise implementation is certainly helpful in debugging and surveying the project.

prehended and updated to conform to the requirements of the desired application.[194]

Use Case Text Classification

One concrete example of a well-documented application for an already presented use case can be found in the SVM implementation for the classification task presented in Raab [2023], which was detailed in Section 7.3.2. The approach there, using SVMs as a baseline, was implemented using the popular Python package Scikit-learn [Pedregosa et al., 2011], which is an all-purpose framework for machine learning methods. The actual SVM implementation is done within minimal lines of code using a linear SVM classifier.[195] The simplicity of applying this classifier, benefited by the comprehensive documentation of Scikit-learn, showcases an advantage when applying established and traditional models. This SVM classifier was originally only intended as a baseline, but due to its performance was observed as outperforming the transformer-based approaches on one of the two use cases, i.e., the classification task on insurance datasets, see Raab [2023, *p.* 48f.]. The applicability of SVMs will be further discussed below in Section 10.3.

Use Case Text Summarization

Regarding summarization, the LexRank implementation applied in Dal Cin [2025] was similarly simple. In using the implementation available through the Sumy library, see Footnote 191, only a few adaptations were required and just the necessary parameters such as desired length of the summary and input data needed to be provided. The abstractive model based on the mLongT5 architecture, however, required the setup of more parameters and preprocessing steps. Lastly, the chat-based LLM method was realized using as little code as the extractive system. However, the desired number of sentences was included in the prompt as additional parameter similar to LexRank, but this instruction was not always adhered to – in contrast to LexRank – see the discussion regarding reproducibility in Section 7.4.1.

Compatibility Issues

A final observation for this section, which mostly relates to ease of use but also towards documentation, comprises the compatibility issues documented for the efficiency measurements, see Section 8.1 and especially the limitation noted in Footnote 178. The interrelation between GPU architecture and versioning of software packages prevents the use of a wide spread of system configurations from utilizing GPU acceleration to speed

[194] One example is the TextRank implementation of the Sumy summarization library, see `https://github.com/miso-belica/sumy/blob/main/sumy/summarizers/text_rank.py`.

[195] See `https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html`.

up the processing of the transformer-based or neural network-based approaches in general.[196] As was observed during the efficiency tracking in Section 8.1, GPUs prior to the introduction of LLMs are not easily applicable in accelerating the execution of the models due to compatibility mismatches between software packages. While the limited VRAM would only serve as a small catalyst for such systems, the incompatibility prevents a comparison between older and newer GPUs. In contrast, the newest GPU architecture is not supported by all necessary software packages, which prevents the application on the newest hardware.[197] These observations are naturally only snapshots in time, but in developing software for NLP tasks, similar issues have been noted during the whole time working on this thesis and are not uncommon in NLP applications. However, both traditional models, which are usually only executed using the CPU and therefore demand less dependency configurations, and the application of chat-based or API-based use of LLMs simplify the implementation in this regard and enable a quicker and easier approach.

8.4 Explainability

As has been shown in defining the challenge of explainability in Section 6.4.2 and showcased through the relevant respective use cases in Section 7.4.2, the need for understanding and correctly identifying the reasons and causes of NLP models in creating certain texts or making a decision becomes apparent. Not only critical use cases in the legal, medical or business domain benefit from explainable techniques, but also use cases such as typical text summarization for news or research texts, see Sections 3.6 and 7.1.2, can take advantage of models that are able to explain their generated summarizations.

Limitations for Traditional Models

Two restrictions regarding the explainability of traditional models can be observed, as briefly already stated in Section 6.4.2, which can be attributed to the model architecture. In classification tasks, SVMs are the technique most closely related to neural networks in terms of being black

[196] The dramatic increase in performance and popularity of neural networks in machine learning in general and also for NLP depended on the application of GPUs to vastly speed up the processing of calculations necessary to train language models, measured usually in FLOPS, see Dally et al. [2021] for more details on the impact of GPUs on the revolution in deep learning.

[197] As noted in September 2025 for the current Nvidia GPU architecture Blackwell, which was at the time incompatible with TensorFlow, one of the most popular machine learning libraries for Python, see `https://github.com/tensorflow/tensorflow/` and Footnote 178.

boxes, since their hyperplanes are not directly comprehensible. Still, techniques from the research field of XAI such as Local Interpretable Model-agnostic Explanations (LIME) can be applied here to produce local explanations, while global explanations can be retrieved from a qualitative evaluation, comparison of the datasets and identification of the most important features, see, e.g., Yong et al. [2023, *p.* 33f.]. The second restriction is even more natural, due to the deliberate inclusion of randomness for methods such as random forest classifiers. Whereas this characteristic of random forest techniques prohibits reproducibility, see above in Sections 6.4.1 and 6.4.2, explainability is only slightly hindered due to the sequential nature of random forests, meaning that the importance of single features or single decision trees as intermediate trees within the whole forest can be explained or at least ranked [Gulowaty and Woźniak, 2021; Petkovic et al., 2018].

Overall Perspective for Traditional Models

Many other traditional models, irrespective of use case, are based on statistical algorithms or simple rule-based approaches, for which direct analogies between the input data and produced decisions can be drawn. One example is the use of rule-based approaches in order to detect the legal form of a business from a company name, see Pulver [2023] as described in Section 7.3.1, where the specific hand-drawn rule can be logged that was responsible in producing the decision for the chosen legal form. Another example is the use of TextRank techniques for summarization purposes, mentioned in the previous section as part of the discussion on ease of use of traditional models. The TextRank algorithm ranks sentences according to "a 'similarity' relation between them, where 'similarity' is measured as a function of their content overlap" [Mihalcea and Tarau, 2004, *p.* 6], which can be calculated and scored for each sentence. The ranking can then be manually inspected and evaluated, and attributed to the direct operations within the code, see Footnote 194.

Limitations of XAI

The methods from the research field of XAI are, however, not without their drawbacks or criticisms. In Freiesleben and König [2023], the authors present a comprehensive discussion on different limitations within current XAI research, that in parts mirror the limitations of metrics presented in Section 6.2, such as the reliance on quantitative metrics in lieu of an additional qualitative evaluation or the need for a purposeful discussion of the evaluation results [Freiesleben and König, 2023, *p.* 50ff.]. As one sample, LIME as explanation technique is critiqued by Freiesleben and König [2023, *p.* 58f.] as a limited metric, which is only able to provide restricted explanations and should consequently only be used consciously and combined with other metrics that are not susceptible to the same lim-

itations. The need for a combined quantitative evaluation – with multiple metrics showcasing different features and characteristics of the studied techniques – as well as a qualitative analysis is therefore recommended here once more, with a special focus on the perspective of the downstream task or the user at the far end of the technique in question.

Explainability for this Thesis

Similarly to the previous Section 8.3 on documentation and ease of use regarding traditional techniques, a quantitative analysis of explainability is not intended and was not conducted. Instead, a brief discussion of the limitations of certain traditional models is meant to showcase the possibilities of explaining the outputs or decisions produced by traditional models, see the observations for SVMs and random forests as well as small examples for information extraction and text summarization, all from this section. As Jurafsky and Martin [2024, *p.* 515f.] also note, these traditional systems are not only lightweight and more efficient, but also helpful in detecting problems within their results or even when used as part of larger and more complex architecture applying modern models.

8.5 Applicability of Hybrid Models

Definition of Hybrid Models

The final benefit presented in this chapter will be a brief discussion of hybrid models as an alternative architecture variant. The definition of hybrid models in this thesis is in part comparable to the concept of *ensemble methods*, i.e., the use of multiple learning algorithms and combining their results into one system [Zhou, 2025]. Boosting, see Section 2.2.2 and Freund et al. [1996], and bagging, see also Section 2.2.2 and Breiman [2001], are two famous and older representatives of such architectures. These architectures are essentially trying to correct mistakes by either combining the strengths of multiple different learning algorithms into one system, resulting in boosting systems, or by splitting the datasets into non-overlapping subsets and training independent learning algorithms before combining them as votes for classification purposes or averaging for regression tasks, achieving a bagging system [Zhou, 2025, *p.* 23ff. and *p.* 48ff.]. While boosting and bagging are two prominent types of ensemble methods, many more variants have been introduced, often also referred to as combination [Mohandes et al., 2018], averaging [Zhou, 2025, *p.* 68ff.] or voting techniques [Bauer and Kohavi, 1999]. Further architecture types have also been applied, depending on the NLP task, e.g., combining several stages within a pipeline such as back-translation and pivot models for machine translation purposes [Kim et al., 2019]. After defin-

ing the terminology, two types of hybrid architectures will be presented in this section, first in an information extraction task and afterwards for two generative approaches in text summarization and text simplification.

Stacking for Information Extraction

One architecture related to ensemble methods called *stacking* [Wolpert, 1992] was applied in Pulver [2023] to combine several information extraction algorithms to extract the type of business from information on companies, see Section 7.3.1 for more information on this use case. To further improve the accuracy of the identified classes, i.e., the types of business for the companies, stacking was applied by combining different text classification algorithms, in this case Naive Bayes, SVMs, decision trees and FastText. The combined stacking classification system nearly reached the performance of the best-performing single classification system in these experiments, here FastText, but did not lead to an improvement through the hybrid model in this use case [Pulver, 2023, *p.* 45]. For this information extraction application, the hybrid system only led to additional runtime and correspondingly higher energy consumption, without surpassing the accuracy of using a single classification model. Therefore, the hybrid model would not be advisable here.

Ranking for Text Summarization

The second architecture shows that hybrid does not necessarily mean combining multiple algorithms of the same type to produce one decision. Instead, multi-stage systems combining two distinct algorithms can be envisioned. A summarization system can be conceptualized in such a way with two components. This approach, as introduced in Aumiller et al. [2022], computes centrality scores to measure the similarity and connections between sentences by applying a sentence-transformer model, which is fed to the graph-based LexRank summarization model. In the original LexRank approach, the centrality scores were calculated using the cosine similarity, applied on word overlap and IDF-weighting [Erkan and Radev, 2004, *p.* 475]. Thus, a more complex and costly operation was used as intermediary step to improve the "downstream task" of scoring the sentences to produce an optimized extractive summary. This version was further analyzed in the use case on Italian extractive summarization on legal texts, briefly presented in Section 4.1 as part of the results from the SLR in Jegan and Henrich [2025a]. In the Italian legal use cases from Licari et al. [2023], the extractive nature was of highest importance to prevent hallucinations and other undesired generated texts in the abstractive summaries.

Pivot Model for Text Simplification

In a similar vein, two models can be combined to produce an alternative output for text simplification purposes. As presented in Fruth [2022], see also Sections 3.5 and 7.1.1, the main unsupervised reinforcement

learning technique was adapted for German based on the English technique introduced in Laban et al. [2021]. To compare the introduced German system, a pivot model was created, see above in Section 7.3.2, which applies the original English simplification model on the German input texts that were translated beforehand by a state-of-the-art translation model [Tiedemann and Thottingal, 2020] into English and after simplifying, translating the simplified English text back into German. Multiple evaluation metrics were applied to capture the varying facets of simplification, see Sections 3.5 and 7.2, and while no overall clear best model across all metrics was identified, the pivot model led in metrics such as SARI and other quantitative metrics, but trailed behind in metrics showcasing semantic simplification features [Fruth, 2022, *p.* 48ff.]. As for qualitative evaluation results, the pivot model produced more fluent texts with less grammatical mistakes. Instead, however, hallucinations as newly introduced sentences were observed in contrast to the models explicitly trained for German. Also, as a further disadvantage, the use of two additional systems as part of the full simplification pipeline adds new potential for mistakes through the translation procedures. The efficiency drawbacks of the previous systems mentioned in this section regarding information extraction and text summarization, i.e., worse efficiency regarding runtime and energy efficiency, also apply for this simplification use case.

Summary for Hybrid Models

In the end, hybrid models – as referred to in this thesis – comprise different architecture types of text processing systems. Multiple models attempting the same problem and combining their results to produce one final output is one potential architecture. Likewise, a pipeline of two or more subsequent systems with different goals are equally conceivable, e.g., the pivot models mentioned above for text simplification purposes. While efficiency and complexity are usually impacted negatively by using hybrid systems, they can be applied to cover a wider spread of different algorithms in case of stacking for text classification, see Pulver [2023]. Moreover, hybrid models can deal with domain-specific requirements such as the demand for extractive summaries, as used by Licari et al. [2023], or to provide comparisons for studies with few comparative datasets or models, see Fruth [2022]. The applicability of such architectures depends on the concrete task and its context, e.g., the domain, available datasets and models for comparison.

After noting various hybrid systems and concluding the benefits captured by traditional models, an updated and improved definition of the term *traditional techniques* will be presented next.

8.6 Traditional Techniques: Refined Definition

Broad Classification

Traditional techniques have now been discussed from various perspectives. Through classifying techniques as part of the SLR from the related work in Section 4.1, a broad view on different NLP tasks has been gained. Applications with ongoing uses of traditional methods have been collected that fit within the early definition of traditional techniques from Section 2.5. The challenges of current NLP research and practice enabled a further perspective in Chapter 6, which was discussed through NLP methods and in concrete application scenarios, in which those challenges were noted and evaluated, see Chapter 7. Lastly, and consequently after the analysis in Chapters 6 and 7, benefits of methods and techniques were collected in the current Chapter 8 for those methods that could be considered traditional methods.

Characteristics of Traditional Techniques

The noted factors of traditional techniques from Section 2.5, i.e., reproducibility, efficiency, explainability and documentation, are all factors that still play a key role in determining if a technique or model would be classified as traditional as used in this thesis. However, not all four factors need to be equally met in order to categorize a model as traditional since these four are broad categories, see the distinction regarding efficiency in runtime and energy consumption as evaluated in Section 8.1. Reproducibility and explainability are two factors which are somewhat restricted, as noted earlier, once again in Section 2.5. Random forest classifiers, by their randomized initialization, see Section 8.2, regarding the former factor and SVM classifiers, which are only explainable through XAI techniques such as LIME, see Section 8.4, regarding the latter factor, are two examples of traditional techniques that do not directly support the four factors from Section 2.5. Efficiency is more clear-cut, however even modern techniques such as FastText, as noted in Section 8.1 and Pulver [2023, *p.* 44ff.], can in certain application scenarios reach runtimes comparable to the fastest traditional techniques, in this case a Naive Bayes classifier. Nevertheless, FastText, similarly to the borderline case of word2vec, is not easily grouped into either category. The architecture of both word2vec and FastText using large embedding spaces to represent semantic meaning of words would appear more similar to modern language models, however with distinctly more efficient processing capabilities.

Limitation of Definition

Therefore, a clear-cut distinction by applying a whitelist and blacklist approach is insufficient here. This limitation is caused by the uncertainty regarding the four factors from Section 2.5 – reproducibility, efficiency, explainability and documentation – and in certain dimensions comparable

measurements from more modern techniques such as word2vec and FastText. The use of a staggered architecture, as used in boosting architecture, see Section 2.2.2 or earlier in this chapter as hybrid architectures, see Section 8.5, introduces additional vagueness. The application of several classifiers in a boosted approach to learn the best classification method [Freund et al., 1996] or the use of both extractive and abstractive text summarization methods to prevent hallucinations among other goals [Aumiller et al., 2022], decreases efficiency and explainability in contrast to single traditional methods.

Other Categorizations

Thus, a different approach was attempted to further define traditional methods, which will be presented later in Chapter 10. Two perspectives will be displayed there, the first from the point of view of different approaches. As given in the early definition in Section 2.5, reviewed in the SLR in Section 4.1 and identified in Chapter 7 in concrete use cases, techniques will be presented and a classification will be proposed for which NLP tasks the concrete techniques are applicable, see Sections 10.1 to 10.4. The second perspective takes the other point of view, using the NLP tasks as basis and presents options regarding techniques, which can and should be applied. In this practical interpretation of techniques, using a task-centered and technique-centered perspective, a more granular recommendation as well as applicability study will be presented, which is much more advisable than a simple black and white distinction.

As a conclusion to this chapter, benefits of traditional models in different dimensions such as efficiency for both runtime and energy consumption in Section 8.1, reproducibility in Section 8.2, ease of use and documentation in Section 8.3 as well as explainability in Section 8.4 have been collected to provide a basis for answering research question 3. While the benefits of hybrid models are somewhat reduced regarding efficiency, they, i.e., hybrid models, provide other advantages to purely neural network-based models, e.g., explainability. After comprehensively discussing the benefits and limitations of traditional models while also noting alternative hybrid system architectures, it seems natural to test the listed benefits from this chapter on a concrete use case. The task of text segmentation, introduced in Section 3.4, will serve as the chosen NLP task analyzed in the following Chapter 9.

9 Case Analysis on a Concrete NLP Task: Text Segmentation

The overarching themes of this thesis shall now be applied in such a way and by following the benefits from the prior Chapter 8 that a concrete usage scenario is studied, and the benefits are explicitly stated in regard to this scenario. In order to do so, a new use case will be presented in Chapter 9, with a problem analysis of the scenario, the implementation and realization of the task with both traditional and modern methods as well as an evaluation.

Identification of the Use Case

The scenarios and use cases already collected in this thesis were contextualized through observations and an interpretation of the results for each of the applications in Chapter 7, supported by the challenges that were presented in the prior Chapter 6. These use cases and the benefits included within traditional approaches were detailed in the previous Chapter 8, both for single use cases and in general. As noted above, research questions ❷ & ❸ were already studied as part of prior chapters, for research question ❷ in Chapters 4 and 7 and for research question ❸ in Chapters 6 and 8 respectively, but both are relevant for the current chapter as well. The competitive nature of traditional models in contrast to modern LLM-based methods will be critically analyzed for research question ❷. In the evaluation for this use case in Section 9.3, benefits will be further analyzed regarding research question ❸. Extending the scope of the observations from Chapters 6 and 7, an analysis for a new scenario from the ground up was conducted for an NLP task that was not already researched during this thesis.[198] Text segmentation was chosen due to several reasons.

A prerequisite for the scenario was a less researched field of study, so that new insight could be gained. In realizing this scenario, lessons

[198] The inclusion of text segmentation in Section 7.2 notwithstanding, since the presentation of text segmentation as an NLP task in that chapter was done only after the research in this use case was finished. Section 3.4 was included for the sake of completeness in the earlier part of the thesis, to collect the major NLP tasks discussed in this thesis in one chapter.

learned from analyzing the engineering and design theory domains, see Chapter 5, were applied on this task, therefore further discussing research question 4 – referring to the systematic selection process of choosing an appropriate method for an NLP task. While some research has been done for text segmentation in a number of approaches, see Section 3.4, a comparison of traditional models and modern LLM-aided systems was, at least at the time of conducting the study in early 2025, not done beforehand. Additionally, an established approach, in this case TextTiling by Hearst [1997], was available as comparison point to the outputs produced by LLMs. Finally, a dataset and corresponding scenario were also available, with prior experience and observations for an earlier segmentation approach, and interestingly enough, with an earlier GPT version. Thus, the differences not only between a traditional and modern LLM-based approach could be analyzed, but also between two modern approaches could be assessed.[199]

Selection of Text Segmentation

9.1 Scenario

The text segmentation process is one part of a larger text processing architecture, from which the scenario described in this section has been identified and studied. The encompassing project as described in Lehmann and Landes [2024] aims at providing a recommender system for students, who are using learning videos and accompanying metadata to gain further insights into the contents of a software engineering course at a German university. The text segmentation is part of a pipeline, which uses transcripts that have been generated from lecture videos of the mentioned course on software engineering as the input data.[200] After segmenting the transcripts, keyword extraction and topic detection approaches are used as foundation for further computation to produce an ontology, from which topics from within the transcripts are collected. Through the ontology, a recommender system is built to serve students related videos segments, if they appear to need further assistance to grasp the topics of a certain video. For more details of the intended use case, see Lehmann and Landes [2024, *p.* 2f.].

Concrete Scenario

[199] The use case presented in this section was also published in Jegan and Henrich [2025b].
[200] The transcripts were produced by using the automatic speech recognition system Whisper, see Radford et al. [2023].

Prior Segmentation Approach

An experimental segmentation in the approach from Lehmann and Landes [2024] was carried out using GPT-3.5.[201] The results in that implementation, however, revealed unreliability regarding the produced output [Lehmann and Landes, 2024, *p.* 4]. Even though the LLM was instructed to return the original video transcripts only with the addition of marking the segment boundaries, the language model returned summarizations or lists of key points of the input transcript. The instructions included in the prompt were therefore not followed by GPT-3.5 and the produced output was not usable for future processing within the pipeline. In order to still follow up with the intended goal of a recommendation system, a segmentation still had to be realized. The alternative from Lehmann and Landes [2024, *p.* 4] was simply time-based, thus 20 seconds were chosen as intervals. These intervals still include thematically relevant contexts and information, which were useful for downstream keyword extraction tasks. Thus, a basic, rule-based approach was chosen, due to unreliability of a modern approach, which is one of the problems described above regarding reproducibility (in Section 6.4.1 and Section 7.4.1) as well as explainability (in Section 6.4.2 and Section 7.4.2).

Concrete Task

The goal of the study in this chapter was to analyze different segmentation approaches, with two different perspectives in mind. The first is the comparison between one traditional approach, here TextTiling, and one modern approach, here GPT-4o, in order to gauge whether a modern method is required to achieve adequate text segmentation results for further processing steps. The second goal is connected to the undesirable outcome of the prior GPT-3.5 results described above, i.e., the unreliable output formatting produced by the prior OpenAI model. Thus, the capabilities in general and more specifically the ability to closely follow instructions given through prompts are assessed for this second goal.

Dataset Details

The dataset was provided by the authors from the original paper Lehmann and Landes [2024], including the raw transcripts produced by the automatic transcription software as well as accompanying metadata. After some filtering and preprocessing procedures, 172 video transcripts remained. The size of these transcripts, however, is not uniform. While the average in terms of number of sentences is 89.07 sentences per transcript, the standard deviation is 94.70, with a maximum of 580 and a minimum of 12 sentences. Likewise, the number of tokens paints a similar picture, with an average number of tokens per transcript of 1,115.03 tokens, a standard deviation of 1,040.97, a maximum of 5,519 and a min-

[201] GPT-3.5 is also the GPT variant powering the initial ChatGPT release, see `https://openai.com/index/chatgpt/`.

imum of 176 tokens. The widely differing transcript sizes present just one difficulty in properly segmenting the input data. Another complexity is presented since the dataset consists of spoken language. The syntactic markup of spoken language differs markedly from written language. Whereas written language, at least in scientific and non-fiction genres, typically consists of somewhat structured texts that contain a consecutive or sequential nature, spoken language does not necessarily contain the same composition. Asides, repetitions, spontaneous ideas or longer chains of thought akin to structures such as streams of consciousness all appear when transcribing spoken language and lead to more flexible and less sequential structures. These characteristics compound the difficulty in segmenting the transcripts.

9.2 Realization

Working Method

The lessons from other disciplines such as product development described in Chapter 5 are applied from the onset. The processing workflow first presented in Figure 5.1 regarding the interaction between a universal model and concrete data, which in combination enable a working method and result in product features and attributes, can be applied here, see Figure 9.1. The universal model, in this case, would be an applicable model able to segment the text. While this seems simple enough, the difficulty in segmenting spoken language, mentioned above, and achieving the desired extent of segments deserves some analysis in this early step. The data part of this workflow is straightforward, using the transcripts provided by the colleagues from the original paper from Lehmann and Landes [2024]. The third part of the workflow, the working method and approach from the original Figure 5.1 would in practice involve the preprocessing of the data, which can differ depending on the intended technique and execution of the segmentation approach. The resulting product features and attributes in Figure 5.1 would in this case be an analysis of the results in terms of quantitative as well as qualitative evaluation with a special consideration on efficiency, as one major goal in comparing traditional and modern techniques.

Layered Development

The sequential nature of processing is the focus of the model from Figure 9.1, with the distinction between different methods not at the forefront of this visualization. Instead, the layered development and construction process, described in Chapter 5 in a more general manner in Figure 5.2, presents just this distinction, with different design or construc-

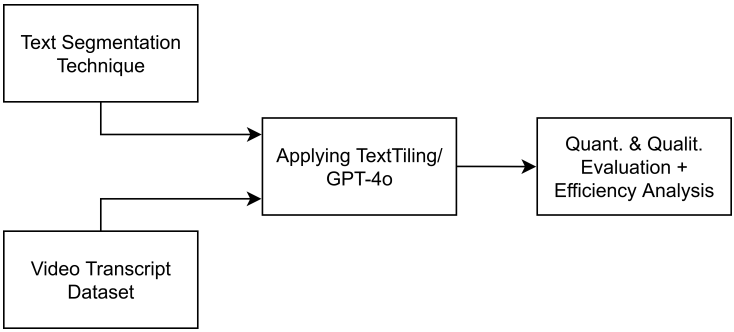


Figure 9.1: Processing workflow during method realization for the text segmentation task. Adapted from Feldhusen et al. [2013, *p.* 12].

tions steps as representatives of applying various models to a problem. In an earlier exemplary visualization in Figure 5.3, a text summarization task was applied using extractive techniques on the popular CNN/DailyMail dataset in this domain. For the text segmentation task, the visualization would simply result in exchanging the entities in the three layers, see Figure 9.2.

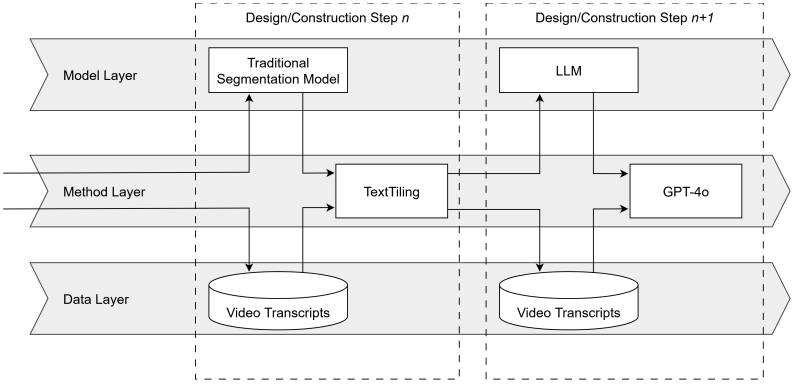


Figure 9.2: Layered development and construction process for the text segmentation task. Based on Feldhusen et al. [2013, *p.* 13].

More Links to Product Development

Further conceptual links to Section 5.1 can be drawn. The consideration between sequential and concurrent development for this study was strongly in favor of sequential development. Due to the small nature of the segmentation task, in terms of scope, manpower, budget and time constraints, no concepts such as iteration cycles or reflections to earlier parts

of the development process were necessary. Nevertheless, lessons learned from the evaluation would certainly lead to adaptations for a renewed realization of this task. Another reference to the lifecycle of products can be seen in the rather static nature of this segmentation task. The dataset studied here was fixed with no new data incoming or streaming during this project. Therefore, the method was constrained to the static dataset and no software in production or active use was required, rather a processing of the fixed dataset to further pass onto the next step in the pipeline from Lehmann and Landes [2024]. Budget, as another development dimension was somewhat constrained, which is why only one execution run was intended for the API calls for GPT-4o to reduce costs, however with trial runs to safeguard against unforeseen formatting or other problems with the segmentation by the LLM.

Selection Criteria for Segmentation Task

In contrast to many modern text processing workflows, often implemented using Scrum, see Section 5.2, the task was approached using a systematic process. After initially analyzing the data, in exchange with the original authors from Lehmann and Landes [2024], a literature review was started to gauge potential models applicable for this segmentation task, all while keeping the dataset as well as the research question of contrasting traditional and modern models in mind, see once again Section 5.2. Criteria to conform to a systematic approach comprise clarity and simplicity regarding the chosen method and safety principles, which in many cases are mostly applicable to physical product development, see Pahl et al. [2007, *p.* 518ff.] for a comprehensive overview on such criteria. However, even for NLP tasks, simple and reproducible approaches can still be surveyed and afterwards selected. Also, interdisciplinary work and communication with other stakeholders are not only critical in product development but can also be observed for the task presented here, which was done by communicating with the data providers from Lehmann and Landes [2024]. Many of the already mentioned dimensions can also be found above in the general list in Table 5.1 from Chapter 5, which is applied to the task of segmenting the video transcripts below in Table 9.1.

Parameter	Characteristics for NLP Method			
Budget	-	Small	Medium	Large
Time Frame	-	Short-term	Mid-term	Long-term
Headcount	Single	Small	Medium	Large
NLP Skills	N/A	Basic	Intermediate	Proficient
Developer Skills	N/A	Basic	Intermediate	Proficient
Other Skills	N/A	Basic	Intermediate	Proficient
Data Availability	N/A	Little	Medium	Much
Data Size	Single Token	Sentence	Paragraphs	Large
Data Quality	Unknown	Low	Medium	High
Data Structuredness	None	Low	Medium	High
Model Count	Single	Two	Multiple	-
Metrics	Quantitative	Qualitative	Quant. & Qualit.	-
Hardware Resources	-	Low	Medium	High
Customer Size	None	<100	100-10,000	>10,000
Availability	None	Selected Times	Fixed Intervals	Constant
Portability	None	Low	Medium	High
Adaptability	None	Low	Medium	High
Explainability	None	Low	Medium	High (Required)
Reproducibility	None	Low	Medium	High (Required)
Risk Factors	-	Low	Medium	High
Documentation	-	Low	Medium	High

Dimension Overview for Segmentation Task

Table 9.1: Dimensions relevant for planning and structuring the segmentation task. The selected dimensions are highlighted with a gray background, see Table 5.1 for the original table and Section 5.2 for explanations regarding the parameters.

Reasoning for Selected Dimensions

In the following, the selected dimensions from Table 9.1 will be briefly justified. Due to the small nature of the task, the choices for budget and headcount require no explanation. Time frame was determined as mid-term, since no direct deadline was set beforehand, but progress in this research and an intended submission at future conferences were reasons for purposefully working on this case analysis.[202] The skill dimensions are self-reported estimates, while the data dimensions depend on the small dataset, with varying quality due to the automatic transcriptions by Whisper [Radford et al., 2023] that necessitated some preprocessing. The data comprises unstructured text, which is why low structuredness was chosen.

[202] The time frame for this task was therefore set according to the submission deadline of the *NLP4Sustain* Workshop at KONVENS 2025, see Jegan and Henrich [2025b].

Model count is obvious, due to the pre-conceived restriction on two main models, i.e., one traditional and one modern, and a comprehensive evaluation with both quantitative and qualitative aspects was desired. Powerful workstations were available for the experimental implementation, while customers in a strict sense were not intended due to the pure research nature of this study, similarly to availability, portability and adaptability, which are all set to none due to the small-scale fixed extent of the task. Explainability and reproducibility were similarly rated as medium, since the qualitative evaluation depends on explainable results. Risk factors are low due to the dataset not being published anywhere, which would be a critical aspect if this were the case since the video transcripts are not intended for public use. A moderately high degree of documentation was desired, at least for the traditional model, to enable quick and seamless implementation.

Soft Skills for Segmentation Task

Only after multiple discussions with the authors from Lehmann and Landes [2024] could the dataset be analyzed and further preprocessed in such a way that a purposeful segmentation approach could be envisioned. The continued exchange was required since the main goal of the initial project was not simply text segmentation, but a more complex pipeline with keyword extraction and topic detection, which serve as a basis for an ontology and recommender system, see above in Section 9.1 and in Lehmann and Landes [2024]. Thus, the input data was structured as to serve the later steps in the pipeline and therefore required a deeper examination and exchange with the original authors, further underscoring the needs for communication, see the discussion on soft skills in Section 5.3. While the research domains from the data providers and the author of this thesis are similar, even then particular focus areas of the concerned parties differed, emphasizing the need for exchange. In this concrete case, interdisciplinary work was not required due to diverse domains, but rather due to different expertise and goals of the two different studies, which nevertheless shows the need for communication between projects. Other dimensions listed above during the discussion on soft skills in Section 5.3 would be necessary if a larger team were implementing this use case. In that case, personnel and other skills would need to be considered and adapted to this application, such as programming knowledge regarding NLP problems or a basic understanding of the data domain from the course the input data is transcribed from, i.e., software engineering.

Identification of Concrete Models

After studying aspects of the lessons taken from product development and other interdisciplinary processes introduced in Chapter 5, the concrete implementation was initialized. In the case of LLMs, a wide variety

of general-purpose LLMs were considered. In the end, GPT-4o [Hurst et al., 2024] provided by OpenAI, was chosen due to its state-of-the-art performance compared to other available LLMs [Islam and Moushi, 2025], at least at the time of conducting this study in March of 2025. The availability of the OpenAI platform[203] regarding ease of use further determined this selection. In the case of traditional models, due to the niche of text segmentation as a discipline, the breadth of available models was moderate, at least for the goal of this use case of text segmentation. An overview of existing techniques was already presented above in Section 3.4, but in addition to the presented techniques there, other approaches based on clustering [Lamprier et al., 2007a] and topic-modeling [Misra et al., 2011] were noticed in the literature review of existing approaches. Through specifying the literature review on approaches that would serve the goal of producing thematically cohesive units, compared to other segmentation approaches with goals of broader clustering, and a focus of accompanying libraries for the respective approach, the decision for TextTiling was reached.

TextTiling

TextTiling introduced by Hearst is intended to produce coherent segments by characterizing texts through subtopic shifts, which are based on "lexical co-occurrence and distribution" [Hearst, 1997, *p.* 33]. In contrast to previous approaches, multi-paragraph segments are intended to capture coherent text units that would present difficulties for other approaches that are purely size-based or statistical, see Section 3.4. The approach is structured into three parts, see Hearst [1997, *p.* 48ff.]: (1) Tokenization of the source text into units of text that are roughly the dimension of sentences, but can include multiple sentences or only parts of one sentence due to widely differing sentence sizes. These units of text are termed pseudosentences. (2) Determining the score for each pseudosentence or number of pseudosentences by comparing adjacent blocks of text.[204] The score is calculated by measuring "how many words the adjacent blocks have in common" [Hearst, 1997, *p.* 49]. (3) Identifying the boundaries between pseudosentences and therefore the subtopic shifts, when they occur and consequently determining the segment border.[205]

[203] See `https://platform.openai.com/docs/overview/`.

[204] Another method of determining the score is presented in Hearst [1997, *p.* 50], by using vocabulary introduction. The implementation in this use case applies the block comparison method described above.

[205] Another advantage regarding TextTiling is the already available implementation as part of the NLTK Python library. See Bird et al. [2009] and `https://www.nltk.org/_modules/nltk/tokenize/texttiling.html`.

Implementation

The actual implementation was straightforward. GPT-4o was applied using the OpenAI API platform, and sequentially, all transcripts were sent to the endpoint provided by the API. The prompt was kept as simple and as concise as possible and the following template was used, in the original German version:

> Die folgenden Absätze stammen aus dem Transkript einer Vorlesung
> zum Thema Software Engineering:
> {transcript}
> Aufgabe:
> Erstelle Segmente basierend auf dem gegebenen Text.
> Hinweise zum Ausgabeformat:
> Gib ausschließlich die Segmente zurück, mit einer leeren Zeile
> als Trennung zwischen den Segmenten.

A translated prompt into English would read:

> The following paragraphs are taken from the transcript of a lecture
> on software engineering:
> {transcript}
> Task:
> Create segments based on the given text.
> Notes on the output format:
> Return only the segments, with a blank newline as a separator
> between the segments.

The instruction at the end of the prompt was added as additional task description, to prevent the problems from the initial implementation using GPT-3.5 in Lehmann and Landes [2024], which did not result in the desired segments, see above in Section 9.1. The TextTiling approach was also conducted sequentially, i.e., all transcripts were processed using the TextTiling implementation within NLTK. Two parameters from TextTiling can be adjusted. w represents the size of pseudosentences, see above in this section and in Hearst [1997, *p.* 48f]. k denotes the block size, i.e., the number of pseudosentences, that are compared with each other to adjacent groups of other pseudosentences to detect segment boundaries [Hearst, 1997, *p.* 49f.]. 121 parameter configurations for w and k were executed, see below in Footnote 207.

For both models, TextTiling and GPT-4o, comparable scripts in terms of complexity and length were produced. One slight drawback was noticed in terms of the documentation for TextTiling. While the original paper Hearst [1997] was detailed and comprehensible, the NLTK documentation was sparse and lacked additional information such as examples.[206] Still, due to the simple and transparent nature of the TextTiling approach the

[206] See `https://www.nltk.org/_modules/nltk/tokenize/texttiling.html`.

underlying code was still understandable, which presents one practical advantage of traditional methods, see above in Sections 8.3 and 8.4.

Having detailed the scenario and its realization, the next section comprises different factors regarding the evaluation of the use case.

9.3 Evaluation

Scope of Evaluation

The evaluation was not only conducted to assess the quality of the produced segments. While the quality of the produced output is naturally an important aspect of every NLP method, as has been mentioned during the high-level discussion of methods and approaches in general above in Section 4.2, other facets have been noted throughout this thesis as well, which will also be analyzed in this section. First, a quantitative and statistical evaluation is presented in Section 9.3.1. A qualitative evaluation will be displayed in Section 9.3.2, before efficiency in terms of runtime and energy consumption will be analyzed in Section 9.3.3 and other relevant factors. Section 9.3.4 will conclude this chapter.

9.3.1 Quantitative Evaluation and Outliers

Basic Segment Size

The first quantitative metric simply calculates the number of produced segments per transcript. All 172 transcripts were processed separately, which is why an average across the number of segments per transcript was the logical first step. The results for GPT-4o and all 121 TextTiling parameter configurations were collected and surveyed.[207] To analyze a select number of TextTiling results more closely, three parameter configurations were chosen, one of them representing the default parameters given in Hearst [1997] as $w = 20$ and $k = 10$, and two adjacent parameter sets, see above in Section 9.2 regarding details on w and k. The results are printed in Table 9.2.[208]

Further Dimensions of Segment Sizes

In order to compare the sizes of the produced segments against each other in another manner, the average number of sentences and tokens were calculated for each segment. The selection of the three parameter configurations regarding the TextTiling parameters of w and k align

[207] In total, 121 parameter configurations were tested, with the following values $[1, 2, 3, 5, 8, 10, 15, 20, 30, 40, 50]$ being selected for both w and k.

[208] Due to size constraints, not all metrics are printed in this thesis. The full statistics are available at the GitHub page of the corresponding publication Jegan and Henrich [2025b] at `https://github.com/uniba-mi/text-segmentation/blob/main/data/results-statistics.xlsx`.

Model	**No. of Segments** Average (SD)	**No. of Sent.** per Segment Average (SD)	**No. of Tokens** per Segment Average (SD)
GPT-4o	8.95 (5.48)	13.73 (29.07)	176.74 (399.30)
TextTiling			
$w = 20; k = 5$	8.67 (7.01)	8.85 (2.88)	118.66 (21.16)
$w = 20; k = 10$	8.59 (6.97)	8.92 (2.71)	120.21 (20.15)
$w = 30; k = 10$	5.94 (4.53)	12.73 (4.70)	169.78 (36.14)

Table 9.2: Quantitative results of the text segmentation task. *Model* represents the applied approach, with w denoting the size of pseudosentences and k the block size as variables for TextTiling, *No. of Segments* denoting the average number of produced segments across all transcripts, *No. of Sent.* the average number of sentences per segment and *No. of Tokens* likewise for number of tokens per segment, once again averaged across all transcripts. The standard deviation is given in brackets. Adapted from Jegan and Henrich [2025b].

closely to the three quantitative metrics achieved by GPT-4o, see Table 9.2. The number of segments, for $w = 20$, and for number of sentences and tokens for $w = 30$, roughly correspond to the results from GPT-4o and thus offer a further comparative aspect regarding the differences through parametrization of TextTiling and the results produced by GPT-4o. A closer look at the produced segments revealed noticeable and, after a brief analysis, less comprehensible results generated by GPT-4o for the dimensions sentences and tokens. When statistically evaluating the output of the four models, it was observed that outliers across these statistical measures were most prominently produced by GPT-4o. To further evaluate this aspect of the text segmentation procedure, the standard deviation was calculated for all three measures, and based on these scores, further and more qualitative analysis of the standard deviation was conducted.

Analysis of Outliers

The standard deviation of number of sentences and number of tokens revealed drastically higher values for GPT-4o than for the three TextTiling models, by roughly one order of magnitude, see Table 9.2. Due to this unexpected occurrence, which was not noted in the initial qualitative spot-checks of the results, a deeper analysis was conducted. When surveying the segments of all transcripts in respect to the transcripts themselves, a markedly higher number of outliers was noticed for GPT-4o. Concretely,

GPT-4o produced extremely long segments for dozens of transcripts or in one case no segments, i.e., the transcript was not segmented at all.[209] A manual review of the transcript, for which GPT-4o produced no segments at all, revealed no obvious characteristic, which caused this failed segmentation, or special property of that text, which differentiates this text from other transcripts. The reasoning behind this failed segmentation and the other outliers in terms of extremely long segments are uncertain, instead further underlining the unreliability behind the output generated by modern LLMs, especially for more uncommon use cases such as text segmentation. The values for TextTiling present a more consistent picture, able to produce segment sizes depending on the parameter selection, since lower values of w and k for these three configurations result in lower numbers of sentences and tokens, and vice versa for higher w and k values.

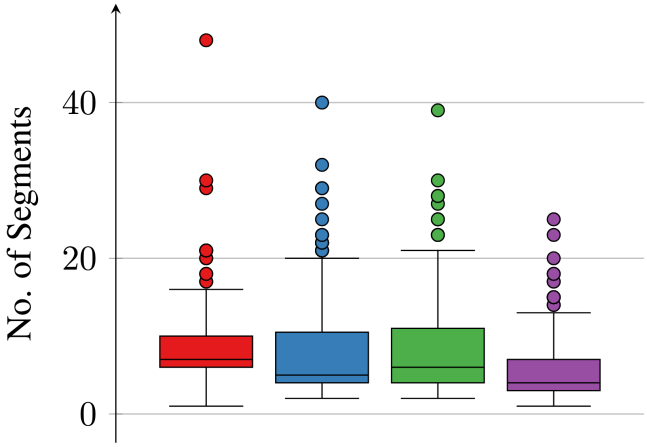


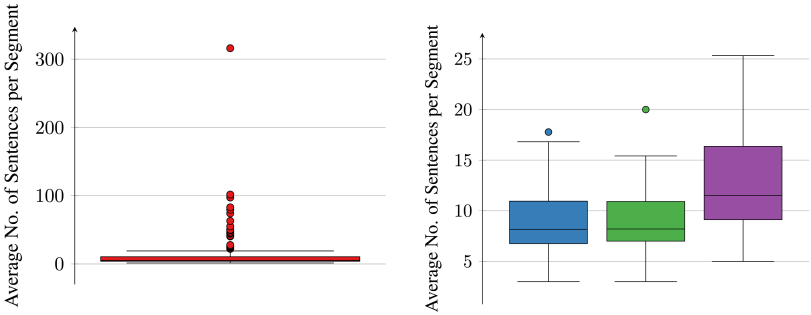
Figure 9.3: Number of segments for each transcript. All 172 transcripts are presented in each box plot, with the number of segments per transcript displayed on the y-axis. Red corresponds to GPT-4o, and the other colors to TextTiling models with the following parameters respectively: Blue to $w = 20$; $k = 5$, green to $w = 20$; $k = 10$ and violet to $w = 30$; $k = 10$.

Number of Segments

The number of segments per transcript is shown in Figure 9.3. While the box plots appear similar, a few observations from the underlying data

[209] For a comprehensive quantitative analysis, see the complete statistics in `https://github.com/uniba-mi/text-segmentation/blob/main/data/results-statistics.csv`, which collects all quantitative and statistical results of the GPT-4o model as well as all TextTiling models.

are relevant here. From Table 9.2, the highest number of segments per transcript on average are produced by GPT-4o, however by only a slight amount, which is in alignment with the highest median produced for all four models in Figure 9.3. The higher average by the blue model ($w = 20$; $k = 5$) from Table 9.2 compared to the green model ($w = 20$; $k = 10$) is grounded in a different distribution of number of segments, especially for segment values in the lowest and largest quartiles, respectively, which is why the median for the former model is calculated as five and the latter model amounts to six, see the box plots in Figure 9.3. The drastically different standard deviation regarding GPT-4o is not visible in this broader visualization, since the number of segments does not differ that much for the four observed models. In contrast, the analysis of number of sentences and tokens presents a different picture, see Figure 9.4 and Figure 9.5 respectively.



(a) GPT-4o

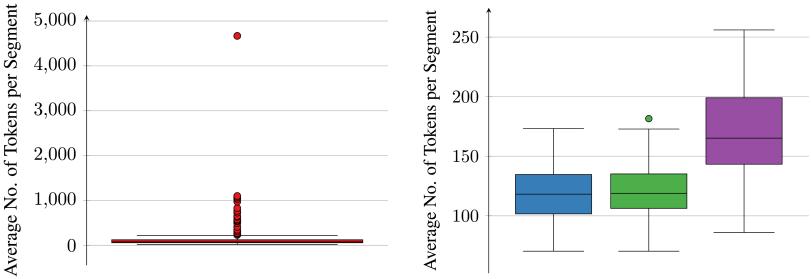
(b) TextTiling: Blue - $w = 20$; $k = 5$, green - $w = 20$; $k = 10$, violet - $w = 30$; $k = 10$

Figure 9.4: Average number of sentences for each segment averaged across the transcripts. All 172 transcripts are presented in each box plot, with the average number of sentences per transcript displayed on the y-axis.

Number of Sentences

The analysis of sentences and tokens necessitates a different processing. For each segment, sentence tokenizers and word tokenizers are applied respectively and the results are averaged across one transcript to get a representation of the number of sentences and tokens per segment per transcript. The sentences are presented in Figure 9.4 and require a separate visualization for GPT-4o due to the extreme outliers, which enable an explanation for the drastically higher standard deviation from Table 9.2. For the outliers at the top end of the scale in Figure 9.4a, extremely long segments with nearly one hundred sentences per segment or in one case

over three hundred sentences for one transcript without any segments, see above regarding this outlier earlier in this section, were generated by GPT-4o. Thus, the segmentation failed in these cases, and compared to TextTiling, see Figure 9.4b, GPT-4o produced uneven, unreliable and undesired outputs. As for TextTiling, higher values of w tend to produce longer segments in terms of average number of sentences per segment, see Figure 9.4b, and also for average number of tokens per segment, see below in Figure 9.5b.



(a) GPT-4o

(b) TextTiling: Blue - $w = 20$; $k = 5$, green - $w = 20$; $k = 10$, violet - $w = 30$; $k = 10$

Figure 9.5: Average number of tokens for each segment averaged across the transcripts. All 172 transcripts are presented in each box plot, with the average number of tokens per transcript displayed on the y-axis.

Number of Tokens & Quantitative Conclusion

The number of tokens confirms the findings from the analysis of sentences, with extreme outliers only appearing for GPT-4o, see Figure 9.5a, and TextTiling once again presenting uniform results, albeit with slightly higher number of tokens for the parameter configuration of $w = 30$; $k = 10$, see Figure 9.5b. Thus, as mentioned above for the analysis of the number of sentences, increasing the parameter w leads to longer sentences overall. This result matches the intuitive interpretation of this parameter, since for longer pseudosentence or token-sequence sizes, which are set through w, longer resulting segments seem natural. Therefore, a parametrization leads to direct adjustments of the output, which can be quantified since the average number of sentences and tokens per segment was not only calculated for the three parameter configurations from above in Figures 9.3, 9.4b and 9.5b, but for all tested configurations, see Footnote 207. The overall trends for parameter w noted here are confirmed

for larger sample sizes starting from $w = 1$ up until $w = 50$.[210] The statistics presented above capture the size of the produced segments in different dimensions, but do not necessarily correspond to the quality in terms of coherence and semantic relatedness within the segments and regarding segment boundaries. Therefore, a brief qualitative discussion is required.

9.3.2 Qualitative Evaluation

Lack of References

As mentioned above during the discussion of lacking metrics in Section 6.2 and use cases such as marginalia generation and others in Section 7.2, NLP methods for datasets without a gold standard or references within the data are inherently difficult to evaluate. The limitations of reference-based evaluations should however not be discounted, see also the discussion of this phenomenon in Section 6.2. Segment boundaries, especially for spoken language and thus texts that are structurally less organized than written texts, are hard to clearly distinguish, for humans and even more so for text processing systems.[211]

Qualitative Evaluation

Instead of a reference-based evaluation, spot-checks for the four models mentioned above were conducted in this study. While the overall quality of the results by GPT-4o and all TextTiling approaches remain comparable, a few observations stand out, which have already been referenced in Jegan and Henrich [2025b, *p.* 277f.]. One such observation includes a slight advantage of GPT-4o in producing semantically more coherent and related segments. This factor can be attributed to, e.g., more segments using rhetorical questions as the first sentence, which is useful as an introductory or starting part of a segment. TextTiling segments did not exhibit this characteristic to the same extent. The strength of GPT-4o to be able to find deeper semantic links and relations between text portions by their immensely more complex underlying language model is one point in favor of the LLM since the traditional model TextTiling with its drastically simpler algorithm is not able to capture such deeper relationships. Furthermore, despite the outliers and the standard deviation values noted above in Table 9.2, the qualitative spot-checks revealed a slightly more uniform type of segments for GPT-4o when compared to the results from

[210] A full statistical analysis is available at `https://github.com/uniba-mi/text-segmentation/blob/main/data/results-statistics.xlsx`.

[211] For a more detailed discussion on the evaluation of text segmentation systems and problems therein, see, e.g., Ghinassi et al. [2024] for a recent discussion or Lamprier et al. [2007b] or Pevzner and Hearst [2002] for older but more focused studies in this research domain.

TextTiling. In contrast, the extreme outliers in terms of size represent disadvantages for GPT-4o, that even became obvious in this qualitative survey. Also, the parametrization of TextTiling and being able to produce segments with different sizes just by adapting parameters and quickly producing output with dramatically lower costs are further points in favor of TextTiling, and will be discussed below regarding efficiency.

9.3.3 Efficiency

Methodology of Runtime Analysis

When analyzing this use case regarding efficiency, different factors need to be evaluated. Of these, the time spent processing the segmentation of the dataset will be studied first. As mentioned above in Section 4.2 and more detailed in Section 8.1.1 regarding the advantages of traditional approaches, the runtime is certainly relevant when comparing different NLP methods intended for a task. The following values all relate to executing the segmentation for the whole dataset once in total, i.e., all 172 transcripts are processed sequentially one time.[212] The values were retrieved for measuring the time processing the segmentation tasks only, thus no preprocessing or other data-cleaning steps are included, and were measured within the Python scripts. The computation was intentionally not optimized in any way, e.g., through parallelization, multi-threading or other measures, in order to keep the processing of TextTiling and the sequential processing done via the OpenAI API comparable. For the latter, the calls to the API were sent sequentially as well. The values are displayed in Table 9.3.[213]

Results of Runtime Analysis

As measured, processing the transcripts through TextTiling when compared to GPT-4o is more efficient by a factor of at least $500\times$ for the measured parameter configuration from Table 9.3. As mentioned, optimization was intentionally omitted. However, due to the lack of transparency and lack of documentation for the computations done by OpenAI to execute the requests received through their API for GPT-4o, it is unclear, what overhead in terms of preprocessing, infrastructure or other pipeline

[212] Since the capability of different hardware configurations naturally has an impact on the processing speed, three different machines were tested. The primary machine was the same system already applied above in Section 8.1, **S1** from Section 8.1.1, which was used to compute the values given in Table 9.3. Other machines were tested as well, ranging from a desktop PC, **S2** from Section 8.1.1, to a more computationally capable server using an AMD EPYC 7443 24-Core processor with 1,000GB of RAM. All tested machines were able to process all transcripts once in between 9 and 16 seconds for the TextTiling parameters $w = 20$ and $k = 10$.

[213] The quoted metrics have also been presented in a less detailed manner in Jegan and Henrich [2025b, *p.* 277].

	GPT-4o	TextTiling		
		$w = 20$ $k = 5$	$w = 20$ $k = 10$	$w = 30$ $k = 10$
Time	6,127.6	9.12	10.72	7.15

Table 9.3: Runtime analysis of the segmentation task. *Time* represents the total time of one execution run, processing all 172 transcripts once, in seconds.

operations are applied that should not count towards the actual segmentation processing done by GPT-4o. Since no other way of more accurately measuring the computation done with GPT-4o is available, this estimate needs to suffice for a broad comparison between the methods. Even so, the runtimes observed in Table 9.3 are a noteworthy and drastic difference between the two approaches.

Further TextTiling Analysis

As for TextTiling, while only three parameter configurations and their quantitative and qualitative results have been analyzed in detail, the minimum and maximum parameter configurations have also been tested. In case of $w = 1$; $k = 1$, the runtime increased by a large margin, which is natural, since the parameter w concerns the size of pseudosentences, which are used to construct blocks that are compared to one another. Lower values for the size of pseudosentences result in larger sets for comparison as well as more comparisons that need to be computed, which both increase the runtime of the segmentation procedure. Thus, the time processing all 172 transcripts with parameters $w = 1$; $k = 1$ took 128.21 seconds.[214] In contrast, the maximum values of $w = 50$; $k = 50$ naturally resulted in drastically lower runtimes, with 5.48 seconds required to process all 172 transcripts once. The fluctuations in Table 9.3 regarding the three parameter configurations of TextTiling can be explained by the same phenomenon as described for the configurations $w = 1$; $k = 1$ in contrast to $w = 50$; $k = 50$, however in a less severe manner due to smaller differences[215] for w and k. The parameter k defines the number of blocks of pseudosentences that are compared to each other, see Section 9.2. Larger

[214] See `https://github.com/uniba-mi/text-segmentation/blob/main/data/results-statistics.xlsx` for all measurements regarding efficiency tracking of TextTiling. All measurements for TextTiling were repeated four times, corresponding to the second to fifth tab in the spreadsheet. Hence, averaged values are presented in this section and in Tables 9.3 and 9.4.

[215] Here $w = 20$; $k = 5$, $w = 20$; $k = 10$ and $w = 30$; $k = 10$ respectively.

values of k therefore lead to longer runtimes for configurations with the same w, however contributing less heavily to the runtime when compared to w overall.[216]

Energy Consumption

Apart from runtime analysis, other dimensions were evaluated, such as the energy consumed by the different approaches. As mentioned above in Section 4.2 and already discussed during the benefits of traditional methods in Section 8.1.2, the energy expended for processing text is another dimension critical to today's NLP landscape, especially in times of LLMs and processing large datasets. The lack of clear standards and practices, briefly presented in Section 4.2, does not absolve potential NLP practitioners from including an analysis of the energy consumption for a prospective method, due to a variety of reasons, such as rising energy demands by many industries and the AI field in particular. The actual measurement of the consumed energy for a specific process is not trivial since different modes of power tracking can be envisioned. Jay et al. [2023] provide a survey on both hardware-based and software-based power tracking and compare the two modes. For this study on text segmentation, the same procedure was applied that has been used and described in Section 8.1.2 for measuring the energy consumption of the use cases presented there.

Limitations

Whereas the power tracking of the TextTiling-based approaches can be measured in detail, the power consumption of the segmentation through GPT-4o is harder to quantify. The developer API platform provided by OpenAI, similarly to the observations from above regarding the runtime analysis, does not provide any information on the energy expended by the requests sent to the API. Thus, a detailed calculation akin to the analysis conducted for TextTiling was not possible, and estimates must suffice. As also noted in Jegan and Henrich [2025b], different studies with varying estimates for the energy consumption of LLMs are available.

Estimations for ChatGPT

One early and widely cited study by De Vries [2023, *p.* 2192] quotes 3 Wh for a single interaction with an LLM, in their case estimating this value for the ChatGPT version available in 2023, then based on GPT-3.5. The same study by De Vries [2023] compared this estimation to the energy consumption of a typical Google search request as a baseline. Such a Google search request, approximated for searches without the recent AI processes, requires one tenth of the 3 Wh estimated of an LLM request, i.e., roughly 0.3 Wh for the search [De Vries, 2023, *p.* 2].[217] While the indexed data has certainly increased manifold, efficiency improvements

[216] See the concrete values in the spreadsheet linked in Footnote 214.

[217] The quoted value of 0.3 Wh from De Vries [2023, *p.* 2] is cited from `https://googleblog.blogspot.com/2009/01/powering-google-search.html`, dating back to 2009.

in processing search queries on software as well as hardware side would nowadays presumably lead to energy consumption that is in the same ballpark as the earlier value. This interpretation is naturally only applicable to search queries without any AI overhead, in contrast to modern search systems, e.g., the AI overviews applied by Google as of 2024, see below in Section 10.1.

Recent Studies

The amount of expended energy has a stronger point of emphasis recently due to the continued interest in LLM development, which is why other studies have been conducted. Newer studies attribute lower power consumption to LLMs, in one such case[218] estimating one typical query to ChatGPT using GPT-4o consuming 0.3 Wh, thus just a tenth of the previously reported value from De Vries [2023] and equal to a typical Google query, see Footnote 217. Other studies such as Stojkovic et al. [2024] or Jegham et al. [2025] focus on different methodologies and also other dimensions such as water consumption and carbon footprint. Unfortunately, the lack of transparency of LLM providers prohibits the accurate quantification of the energy expended for single requests.

Measuring Energy Consumption

In the end, both estimates, the upper bound of 3 Wh per request and the lower bound of 0.3 Wh per request, see above, were used to approximate the actual energy consumption. To capture the processing of all transcripts done by GPT-4o, these values were multiplied by 172 since each transcript was segmented by using a separate request. In contrast to the ranges presented for GPT-4o, concrete values were calculated for the TextTiling approaches by using the Code Carbon software-based power tracking, see Section 8.1.2 for more details on this power tracking software. The captured values can be seen in Table 9.4.[219]

Energy Consumption Results

When comparing the results of the analysis on energy consumption, differences within several orders of magnitude were noted, even when using the more conservative values from the lower bounds of the ranges given for GPT-4o. While estimates were used for GPT-4o due to the unavailability of concrete values, the stark differences between the two approaches become apparent and present a strong argument in favor of using the traditional model. Similar to the analysis above regarding the runtime, the minimum and maximum parameter configurations for Text-

[218] See the study by `https://epoch.ai/gradient-updates/how-much-energy-does-chatgpt-use/`.

[219] The values were calculated using the same system as presented above for the runtime analysis, detailed in Footnote 212.

	GPT-4o	TextTiling		
		$w = 20$ $k = 5$	$w = 20$ $k = 10$	$w = 30$ $k = 10$
Energy	51.6 - 516	0.061	0.072	0.048

Table 9.4: Energy consumption of the segmentation task. *Energy* represents the consumed energy of one execution run, processing all 172 transcripts once, in Wh. The range regarding GPT-4o is caused by a lack of transparency of the provided API, which is why estimates need to be used here that can vary widely, see above in this section for more details.

Tiling were tested as well.[220] On the one hand, applying the minimum configuration $w = 1$; $k = 1$, 0.85 Wh were measured for processing all 172 transcripts once. The energy consumption for this configuration is drastically higher when compared to the measurements of the TextTiling experiments from Table 9.4 due to the increased processing time, see also above during the discussion of the runtime. On the other hand, when using the maximum configuration, $w = 50$; $k = 50$, 0.037 Wh were observed. Both values confirm the findings of the runtime analysis, and match the intuitive interpretation from above, with lower values regarding the parameters resulting in more computations necessary for the segmentation and thus a higher power draw, and vice versa for higher values. The fluctuations between the three parameter configurations from Table 9.4 follow the same logic as for the runtime from Table 9.3.

Interpretation of Energy Results

Regardless, the values noted for GPT-4o are still considerably higher even for the worst-performing TextTiling configuration in this evaluation. Thus, the advantage of the traditional model TextTiling in contrast to GPT-4o is confirmed once more in regard to the energy consumption. Also, the impact of the ongoing climate crisis is an additional factor, which should be considered. The use of the traditional technique in just this small use case displays the stark differences in terms of energy costs between the two models and the ability to conserve much energy with comparable qualitative performance. The energy savings for other, potentially much larger, applications are immense and should be pursued.

[220] Once again, as for the details of the runtime analysis earlier in this section, see the second to fifth tab of the spreadsheet for all measurements in `https://github.com/uniba-mi/text-segmentation/blob/main/data/results-statistics.xlsx`.

Now that efficiency has been analyzed in detail, more dimensions were noticed during the realization and evaluation of this use case, which will be collected in the next section.

9.3.4 Other Dimensions

Budget

One further dimension of evaluating the approaches concerns the monetary expenses necessary to produce the segments. The required software for TextTiling is completely open source and therefore no dedicated expense is necessary, except a machine capable of running the requisite packages. The latter aspect, however, would also be necessary to perform the calls towards the APIs in order to send the requests to the LLMs. For the LLM, additional costs arose, due to the expenses required in using the OpenAI API. This attribute of modern LLMs, the costs in using the models, was already mentioned above during the first definition attempt in Section 2.5, during the discussion of analogies to product development and engineering realm in Section 5.1 and the selection criteria in the same chapter in Section 5.2, which provide a basis for discussing research question 4. The expenses vary depending on the provider and the applied model, but the values shown here present a representative snapshot of applying an LLM for similar text processing tasks. All 172 transcripts were processed once, in sequential order, thus 175 requests were sent to the OpenAI API, including three requests for testing purposes. In total, 248,615 tokens were transmitted to the API, applying the GPT-4o model, resulting in costs of $2.93, or 1.67 cents per transcript.[221] The costs for this small scenario are manageable, but for larger datasets or a permanently available service, expenses can quickly increase.

Locally Executed LLMs as Alternative

Another option to both keep the flexibility of LLMs and prevent ongoing costs shall be mentioned as an alternative, i.e., the use of locally run LLMs, which were briefly introduced in Section 2.4. A few experiments were conducted using models trained with open weights such as LLaMA 3.2[222] and Microsoft's Phi[223], both advertised as capable models without requiring powerful hardware resources. In brief evaluations using the same prompts as for GPT-4o, the segmentation was not successful, i.e., the generated output did not match the input transcripts with additional segment

[221] These values were observed in March of 2025. As of September 2025, the pricing had already changed, amounting to $2.5 per 1 million input tokens. The volatile nature of LLMs that are dependent on their providers was already discussed in Section 2.4.

[222] See `https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/`.

[223] See `https://azure.microsoft.com/en-us/products/phi/`.

markers. This problem thus matches the issues described for the earlier GPT version from the original paper by Lehmann and Landes [2024] for GPT-3.5. The inability of the smaller local models in following the instructions in the prompt follows these observations from GPT-3.5, potentially due to the smaller parameter size when compared to GPT-4o. Additional experiments would be the next logical step for similar models, potentially with the addition of a desired schema to the prompt, which is intended to guide the LLM to generate output structured according to the given schema.[224]

Lessons Learned

After presenting the evaluation and the results of the use case on text segmentation, a few observations deserve to be mentioned, as a lessons learned or retrospective, which should be part of the development lifecycle, as mentioned above in Section 5.1. One learning concerns the quickness of achieving a working method. Both TextTiling and GPT-4o systems were executable within a similar time frame and with comparable amounts of code. While the documentation of the applied TextTiling library could be improved, the simple nature of its method mitigated this weakness and enabled a quick implementation. Furthermore, the advantage regarding both time and energy efficiency for TextTiling in comparison to GPT-4o was even more drastic in reality than expected, with several orders of magnitude for both metrics, see Tables 9.3 and 9.4. This result in terms of efficiency, combined with the qualitatively comparable performance of both systems, confirms research question ❷ as well as provides a concrete example for benefits of using a traditional technique, see research question ❸. Lastly, the parametrization of TextTiling, which displays one advantage of a traditional model regarding customizability for the desired use case was manifested during the quantitative evaluation, see above in Table 9.2 and the analysis of outliers from above. In the end, the structured and purposeful nature of approaching this scenario proved successful and displayed strengths regarding both approaches but also revealed some differences for both methods.

[224] The functionality of structured outputs of locally run LLMs promises this functionality through the Ollama framework, see `https://ollama.com/`.

10 Applicability of Approaches

Following the analysis of the "new" use case of text segmentation in Chapter 9, several classes of traditional approaches will be contextualized regarding their applicability and relevancy in the current NLP landscape: Extractive approaches, not necessarily only for summarization purposes, for the first class in Section 10.1, rule-based techniques and decision trees for the second class in Section 10.2, and SVMs as the last class in Section 10.3. A penultimate Section 10.4 will present other types of traditional approaches that did not fit in the previous three classes. Lastly, the final Section 10.5 will provide a conclusion and attempt recommendations for different use cases from another perspective, with lessons from other domains in mind, mainly Chapter 5, to capture research question 4 regarding the selection process of choosing an appropriate NLP method.

Two different perspectives will be applied in this chapter. First, the perspective of techniques instead of use cases will be the focal point in Sections 10.1 to 10.4 regarding research question 3, showcasing potential benefits of using traditional models. Second, the use case-centric perspective, which was detailed in Chapters 7 and 9 as well, will be applied in Section 10.5 regarding research question 4. The applicability of rule-based approaches for information extraction and SVMs for classification purposes, as discussed earlier in Sections 7.3.1 and 7.3.2 regarding Pulver [2023] and Raab [2023] respectively, was already presented as part of two talks, see Jegan [2023, 2026]. Both talks discussed the trade-off between traditional models and neural networks, with the former talk [Jegan, 2023] providing an initial spark for the main research topics of this thesis and the latter [Jegan, 2026] signifying a closing statement in January of 2026.

10.1 Extractive Approaches

Terminology

The first type of approaches discussed in this chapter comprises extractive techniques, which naturally bring summarization approaches to mind. The defining attribute for extractive summarization techniques, especially

in contrast to generative techniques and as already detailed in Section 3.6, is the selection process of the text that is identified for the summaries, which necessarily needs to appear identically in the original texts. This process is by definition antithetic to abstractive scenarios, which are able to generate new text that is not required to appear in the source document. While the natural interpretation of extractive techniques concerns text summarization, other NLP tasks can also be approached using these extractive techniques akin to extractive summarization, see, e.g., a keyword extraction task using TextRank [Zhang et al., 2020b]. For this thesis, extractive techniques are considered traditional approaches due to efficiency, explainability and reproducibility characteristics. However, hybrid approaches combining extractive and abstractive techniques as discussed briefly in Section 8.5 and especially in Aumiller et al. [2022], or purely neural network-based extractive approaches, see, e.g., Nallapati et al. [2017] or applied in Lang [2025], exist, but this thesis will primarily focus on less complex systems.

Differentiation to Other Techniques

Also, a distinction is required for the techniques presented in the next Section 10.2 for rule-based techniques and decision trees. The techniques typically applied in extractive summarization, e.g., LexRank [Erkan and Radev, 2004] or TextRank [Mihalcea and Tarau, 2004], are not purely rule-based extraction techniques. They instead require a text representation phase followed by a ranking phase, based on some sort of similarity measure between textual units such as sentences. This procedure differs from rule-based approaches or decision trees, which are either strictly sequentially checking rules or applying machine learning-based decision tree learning methods. Furthermore, the techniques from the subsequent Section 10.2 are mainly intended for classification and information extraction purposes and thus produce a class or tag as result of the processing, which differs from the output of extractive approaches that are typically larger in scale, consisting of several sentences.[225]

Snippets

One of the primary use cases of LLMs is summarization tasks, seen, e.g., in research [Zhang et al., 2024b] as well as the inclusion of sum-

[225] Typical summarization approaches are envisioned in this comparison. Keyword extraction approaches, as earlier mentioned in this section, naturally produce smaller-scale output, but the distinction between classifying tasks and text production tasks still remains the same. See the discussion of these types of NLP tasks above in Section 6.2 regarding different evaluation methods and metrics.

maries in the SERP of a regular web search request.[226] As mentioned already in Section 7.2.2, the goal of summaries can vary widely, which holds true for summaries in the context of the SERP. Before the addition of AI-generated overviews, short summaries – here called snippets – consisting of two to three short sentences were included in the SERP besides the URL and the title of the corresponding webpage.[227] Snippets tend to be rather short and indicative, i.e., they do not go into detail regarding the source text. They should also present an overview to users, so that they can decide if the retrieved website in the SERP is relevant to their search intent [Dal Cin, 2025, *p.* 10]. Consequently, snippets can depend on the search requests submitted by the user. In contrast, summaries are independent of the search query in this scenario. Furthermore, many other summary types have differing goals, oftentimes requiring an informative summary, i.e., the collection of the main points of the source text.[228]

Summarization Use Cases

In this thesis, three summarization use cases were reviewed, Gottwald [2023], Harms [2024] and Dal Cin [2025]. While the goal of the intended summaries differed across these use cases, the strong performance of modern, transformer-based approaches was noted across all three studies. While the first use case [Gottwald, 2023] was conducted before the rise of chat-based LLMs, the other two applied Mistral-based LLMs: Mixtral, as used in Harms [2024], and NeMo, as used in Dal Cin [2025], reach the overall best performance for both tasks as representatives of LLMs, for the former on the summarization of German school websites and for the latter on five distinct multilingual summarization tasks. The class-leading performance of LLMs, as noted in Zhang et al. [2024b], is thus confirmed for summarization tasks in these studies.

Role of Traditional Models for Summarization

While the general results of summarization evaluations support the use of LLMs for summarization tasks overall, specific use cases or features would still provide arguments for the use of traditional extractive models such as LexRank. First, efficiency in terms of processing time was noted in both Gottwald [2023, *p.* 43] and Harms [2024, *p.* 24] as clearly favoring LexRank in comparison to all neural network-based systems. Second, explainability [Dal Cin, 2025, *p.* 33f.] and reproducibility [Dal Cin, 2025, *p.* 45ff.] are further attributes, which would favor approaches such as

[226] The integration of summaries in the SERP as conducted by Google using their AI overviews as of May 2024, see `https://support.google.com/websearch/answer/14901683/`, was met with a critical reception due to the opt-out nature of this update with unintuitive ways of deactivating this feature, see, e.g., `https://www.tomshardware.com/how-to/block-google-ai-overviews/`.

[227] See `https://developers.google.com/search/docs/appearance/snippet/`.

[228] See the classification by Jones [1999], also discussed in Section 7.2.2.

LexRank. The extractive approach LexRank can be reproduced easily and is also explainable through its transparent processes like the graph representation of the source texts and its ranking of sentences based on centrality scoring, see Erkan and Radev [2004, *p.* 459ff.]. A final attribute concerns the restrictive nature of the produced output, which can only consist of texts selected from the source document. This characteristic therefore prevents hallucinations from being generated. In the end, if solely the highest quality based on evaluation metrics such as ROUGE is desired for a summary, flawed as such metrics may be, see Sections 6.2 and 7.2, LLMs are the clear recommendation for an intended summarization system. However, if other criteria such as efficiency or preventing hallucinations are of importance, traditional methods would still be worth investigating.

Keyphrase Prediction Techniques

The generation of marginalia, as presented above in Section 7.2.1 from Lang [2025], applied extractive techniques with both traditional techniques, e.g., TF-IDF and PositionRank [Florescu and Caragea, 2017], and neural network-assisted extractive systems such as SIFRank+ [Sun et al., 2020]. Neither traditional nor modern extractive techniques produced results that matched the two best generative methods, mT5 [Xue et al., 2021] and GPT-4o [Hurst et al., 2024], in terms of quantitative evaluation using the f1-score, the MRR or the MoverScore, which was also confirmed through a qualitative survey, see Lang [2025, *p.* 35ff.] and Lang et al. [2025, *p.* 232ff.]. However, TF-IDF was noted as performing strongly, unexpectedly so, since it was purely intended as a baseline. Efficiency benefits are additionally apparent when using this traditional technique, especially in comparison to mT5 and GPT-4o, but also compared to the neural network-assisted extractive technique SIFRank+. As such, TF-IDF outperformed several dedicated keyphrase extraction methods regarding exact matches and even produced the strongest results across all tested systems including mT5 and GPT-4o for approximate matches of present marginalia between reference and generated marginalia [Lang, 2025, *p.* 38].[229]

The lessons from this subsection and the following Sections 10.2 to 10.4 will be collected in an overarching summary in Section 10.5. As noted earlier in this section, summarization techniques produce new text as their output, which differs from the techniques applied in the next Section 10.2. There, classification and information extraction use cases will be detailed from the perspective of two related techniques.

[229] The difference between present marginalia, sourced exactly from the original text, in contrast to absent marginalia was already detailed above in Section 7.2.1.

10.2 Rule-based Techniques and Decision Trees

Terminology

Classification and extraction tasks, which include both NER and PoS tagging, are the predominant tasks associated with rule-based techniques and decision trees, see, e.g., Jiang [2012] and Aggarwal and Zhai [2012a]. Systems using rules or implementing decision trees can be seen as either derivatives of each other or further advancements with decision trees based on rules, depending on their definition or the perspective of the user or researcher. The former interpretation is applicable, if the different potential options modeled through the rules are simply visualized as a tree, which would constitute a decision tree. In a stricter sense, the traditional decision tree model would be problematic in this interpretation, if multiple rules appear to be true at the same time. An applicable policy would need to be employed for such cases, e.g., a determination of a certain sequential order, in which the rules are checked [Jiang, 2012, *p.* 16f.]. The latter interpretation is the one more commonly found in the literature and research, i.e., decision trees as based on rules or classifications modeled as a machine learning task [Orphanos et al., 1999]. The goal of the decision tree is to construct a model for extracting information or classifying certain parts of a text.

Rule-Based vs. Decision Trees

Apart from the terminological differences between the two techniques, a structural or procedural distinction has been noted for classification approaches, see Aggarwal and Zhai [2012a, *p.* 180f.] and Quinlan [1986]. The hierarchical nature of decision trees, in contrast to rule-based approaches, can serve as an advantage or present a restriction, depending on the perspective and use case. As mentioned above, rules allow the representation of overlaps or inconsistencies for cases, for which decision trees would typically require a clear boundary between a single decision or class [Aggarwal and Zhai, 2012a, *p.* 181]. Furthermore, in the more common interpretation of decision trees generated through machine learning, interpretability is not as immediate and direct as for rule-based approaches, but still markedly more approachable than with modern LLMs, see Soon et al. [2001] and Du et al. [2019]. On the other hand, the manual testing and creation of rules within a rule-based system, which are the primary way of generating decisions, extractions or classifications, requires further effort and expertise through the user.[230] Another advantage of decision

[230] The possibility of automatically learning a set of rules should be mentioned here as well. Different types of learning rules are common in research, through either top-down or bottom-up approaches, see, e.g., Jiang [2012, *p.* 17] for more details.

trees revolves around the ability of assessing the importance of features, see Kazemitabar et al. [2017] for more details.

Information Extraction Use Case

The information extraction task presented in this thesis, first introduced in Section 7.3.1, showcases two distinct manifestations, one successful application and another less so, both presented in Pulver [2023]. The first, and more successful implementation, relates to the applicability of rule-based approaches for a still relevant use case, i.e., the identification of legal forms of German companies from their names, as detailed above in Section 7.3.1 and discussed in Jegan [2024]. Due to the limited number of different spellings, abbreviations and other variants of the legal forms, the creation of applicable rules was possible with reasonable effort, while providing comparable performance to the other tested techniques. The other, and less successful approach, demonstrated poorer performance of decision trees for the second use case, which requires the identification of the type of industry or trade based on the company name. Additionally, decreased efficiency regarding processing speed was also noted for decision trees, especially for larger datasets and in contrast to the two leading techniques, FastText and Naive Bayes classifiers, see Pulver [2023, *p.* 44ff. and *p.* 63] and Section 7.4.3.

Entity Matching Use Case

A second use case including decision trees addresses the entity matching problem first presented in Section 7.4.2, see Hebeis [2025]. Decision trees were applied on this classification task in several configurations and in comparison to other traditional methods, see below in Sections 10.3 and 10.4, and modern BERT-based techniques. Both qualitative and quantitative evaluations showcase the comparable performance of models based on decision trees in this use case, with both traditional and modern methods producing similar results quantitatively in terms of the f1-scores and demonstrating their effectiveness for this task as well [Hebeis, 2025, *p.* 42ff. and *p.* 54ff.]. Differences can be noted for efficiency and project prerequisites. Regarding the former, the traditional models based on decision trees and others, see Sections 10.3 and 10.4, are demonstrably more efficient in terms of computational resources as well as processing time [Hebeis, 2025, *p.* 55]. For the latter, i.e., the prerequisites of the applied techniques, BERT-based classifiers as applied in Hebeis [2025, *p.* 38f.] do not require a pre-defined schema mapping between metadata fields that were used for the entity matching purposes, in contrast to the other methods including decision trees. Thus, additional preparatory effort is necessary for traditional methods, with advantages regarding efficiency and explainability, as noted in Section 7.4.2 for decision trees in this use case.

In the use case on entity matching, other techniques including SVMs, random forest and Naive Bayes classifiers were applied. Since the explainability and reproducibility differ in contrast to decision trees, and not consistently within these three other techniques, they will be treated in the next sections.

10.3 Support Vector Machines

Analogy to Previous Section

Similarly to rule-based techniques and decision trees, SVMs are predominately applied for classification [Cortes, 1995] and regression [Vapnik et al., 1996] tasks, the former including information extraction use cases in NLP. The information extraction task recognizing the type of industry from company names, see Pulver [2023] and already described in this chapter in Section 10.2, was also tested with SVMs, which displayed improved performance compared to decision trees [Pulver, 2023, *p.* 45ff.], both in terms of quantitative results for measured accuracy as well as efficiency in terms of runtime. However, even SVMs still trail behind the two leading techniques in this task, FastText and Naive Bayes, for details on the latter technique see below in Section 10.4. In the entity matching use case, see Hebeis [2025] and above in Section 10.2, SVMs were also applied, but did not demonstrate class-leading performance within the selection of the tested traditional models.

SVMs for Classification

A more successful application of SVMs was noted in Raab [2023], as introduced in Section 7.3.2, for the classification of texts from two domain datasets, i.e., the legal and insurance domains. SVMs were used as a baseline in comparison to few-shot approaches, which apply more complex neural network-based techniques such as SetFit [Tunstall et al., 2022]. The few-shot neural network-based techniques were able to surpass the SVM baseline for the legal domain [Raab, 2023, *p.* 36ff.]. For the insurance domain, however, the baseline provided by SVMs achieved the highest scores, even surpassing the modern models including SetFit and ProtoBERT [Tänzer et al., 2022] when evaluated using a typical training and test split [Raab, 2023, *p.* 40ff.]. This leading performance of the traditional model SVM was also presented in Jegan [2024, 2026] and captures the unexpected competitiveness of older techniques, especially when approaching a new task without preexisting training and testing datasets. The insurance domain was of even higher importance compared to the legal domain due to the main background of the study from Raab [2023], which was conceptualized in cooperation with an industry partner work-

ing directly with insurance companies. The legal dataset, used commonly for research purposes and introduced by Glaser et al. [2018], was applied in Raab [2023] to evaluate a second data domain, while the primary focus was on the insurance dataset.

Due to the showcase presented in Raab [2023] with its impressive performance in comparison to modern transformer-based approaches, SVMs were presented separately from the following models. Due to their overlapping goals within the respective use cases and similar application scenarios, Naive Bayes and random forests will all be discussed in the next section.

10.4 Others

Information Extraction Techniques

Both information extraction use cases from Pulver [2023], see the discussion of rule-based approaches and decision trees from this chapter in Section 10.2, were also approached using another technique, i.e., based on Naive Bayes classification procedures. The task of extracting the legal form from company names was processed with higher accuracy by the rule-based techniques in comparison to Naive Bayes, while a staggered hybrid approach of first using rules and only applying Naive Bayes for the remaining cases resulted in the highest scores [Pulver, 2023, *p.* 41ff.]. The other task concerning the identification of the type of industry based on company names presented higher scores for Naive Bayes in contrast to decision trees, both in terms of accuracy and efficiency [Pulver, 2023, *p.* 45ff.]. Additionally, the more modern FastText classification also proved as successful as Naive Bayes, with slight advantages for the modern approach in accuracy while also being slightly less efficient [Pulver, 2023, *p.* 62f.]. Both techniques, Naive Bayes and FastText, provide a reasonable balance of cost and performance, from a runtime perspective, see Section 7.4.3.

Entity Matching Techniques

While several distinct traditional models were tested in Hebeis [2025] to approach the entity matching problem, classification based on random forest algorithms produced the highest f1-scores in several different configurations, for small and large dataset variants [Hebeis, 2025, *p.* 42]. Introduced above in Section 2.2, a key feature of these techniques, their randomness, presents a challenge regarding reproducibility in contrast to many other traditional models. Due to the random seed applied during initialization, reproducible results are not intended for random forest

algorithms by default.[231] The additional efficiency advantages mentioned above for decision trees in Section 10.2, which were noted for the entity matching use case in contrast to the BERT-based approaches, are also applicable for the random forest classification [Hebeis, 2025, *p.* 44].

Text Segmentation Techniques

As detailed in Chapter 9, text segmentation was examined for a concrete use case with the intention of observing the differences between applying a traditional method and modern LLMs, see Jegan and Henrich [2025b]. TextTiling [Hearst, 1997], based on the lexical co-occurrence of words and subtopic shifts, which is calculated through lexical similarity alone [Hearst, 1997, *p.* 49ff.], was applied as the traditional technique for this text segmentation problem. This procedure is notably simple, without using additional data such as thesauri or ontologies [Hearst, 1997, *p.* 59] and therefore highly efficient with comparable performance to modern LLMs, see the discussion on runtime and expended energy in Section 9.3.

TF-IDF Techniques

TF-IDF representations have not only been used as intermediate steps in processing pipelines to vector spaces enabling similarity computations [Jurafsky and Martin, 2024, *p.* 116f.], but have also been shown to perform well on one of the NLP tasks collected in this thesis.[232] The task of marginalia generation, modeled as a keyphrase prediction problem from Lang [2025] and introduced in Section 7.2.1, was evaluated using several extractive and generative systems. As efficient and traditional extractive model, TF-IDF was first only used as a baseline, but during evaluation recognized as the leading extractive model with comparable performance to the neural network-based extractive model SIFRank+ [Lang et al., 2025, *p.* 232]. However, all extractive models including TF-IDF still trail behind the transformer-based methods in this keyphrase prediction task, in this case mT5 and GPT-4o.

After summarizing the performance of traditional methods through experiments and use cases discussed in this thesis, which can naturally only present a snapshot of applications in NLP, the next section will present traditional models from a different point of view. The perspective of the past four sections Sections 10.1 to 10.4 used the techniques themselves as the origin of the discussion, i.e., which techniques were applied successfully in the analyzed use cases. Now, to summarize for prospective users and developers of NLP problems, recommendations will be presented in

[231] Whereas the random behavior is part of the default configurations for random forest implementations, reproducible states can be achieved through parametrization, see, e.g., `https://scikit-learn.org/stable/common_pitfalls.html#randomness/`.

[232] These vector spaces can be interpreted as precursors to early embedding representations such as word2vec, with short dense vectors as main difference to the earlier sparse vector spaces, see Section 2.3 and Jurafsky and Martin [2024, *p.* 117ff.].

the final Section 10.5 of this chapter, from the perspective of the use cases and applicable techniques in their respective cases.

10.5 Recommendations for Techniques & Use Cases

The recommendations presented in this section will be limited to the tasks discussed in this thesis due to the extensive breadth of NLP problems and applications in current research and practice. Thus, the list presented in Section 3.1 will be discussed in that order. In doing so, references to the related work and historical development of the NLP tasks from Chapter 3 and the SLR in Chapter 4 will be presented as well as lessons learned from other domains such as engineering and design theory in Chapter 5.

Recommendations for Information & Relation Extraction

Information and relation extraction has mostly shifted towards the use of transformer-based LLMs, see Section 3.2, however, three approaches from each task were retrieved as part of the SLR from Jegan and Henrich [2025a, *p.* 7.] that still use traditional models as either comparison baselines [Liu, 2023; Mazumder et al., 2023] or, more importantly, as core method of the paper, e.g., in Osman et al. [2023], Sultana et al. [2022] and Xiang and Yangfei [2023]. The latter three papers comprise rule-based methods as well as SVMs and KNN classifiers. As part of this thesis, the rule-based approach from Pulver [2023] was the leading performer in the first subtask, which represents a clearly defined use case with a finite number of forms for the type of information that should be extracted, in this case the legal form of a company [Pulver, 2023, *p.* 41ff.]. When dealing with a similarly small-scale problem, i.e., allowing the definition of rules based on a manageable number of different forms, rule-based approaches or decision trees have a high potential to be as successful as modern and more inefficient systems. Even the slightly more complex second subtask, identifying the type of business from company names, was approached using Naive Bayes classifiers nearly as successful as more modern classifiers such as FastText [Pulver, 2023, *p.* 44ff.], with additional efficiency improvements for the traditional model [Pulver, 2023, *p.* 62f.]. When surveying the domains from Chapter 5, explainability as well as efficiency are the prime factors when dealing with information and relation extraction approaches, enabled through rule-based systems, decision trees or Naive Bayes classifiers.

Text classification has also shifted towards the use of LLMs, see Section 3.3. The SLR nevertheless retrieved eleven publications with ref-

Recommendations for Text Classification

erences to traditional models. Apart from using traditional models as baselines, linear n-gram classifiers [Chen et al., 2023a] and decision tree classifiers [Lin, 2023; Zhang et al., 2023a] were used as the main technique of the respective publication. As part of the classification task on insurance and legal texts from Raab [2023], SVMs were originally only intended as a baseline, but achieved better results than more complex few-shot systems based on neural networks for the insurance dataset [Raab, 2023, *p.* 40ff.]. Due to the special domain of the insurance dataset, without freely available large datasets for training purposes, the used dataset remained rather small. Therefore, few-shot approaches were applied, but could not surpass the performance of the SVM classifier, which shows the applicability of traditional classifiers for such constrained tasks, in which only little data is available. The entity matching task from Hebeis [2025], which can be interpreted as a classification task, applied random forests, XGBoost and other traditional machine learning systems on the matching task. The performances of random forests and XGBoost surpassed the competing BERT-based system, especially for smaller dataset sizes, and never trailing behind the BERT-based system while also staying much more efficient in terms of runtime [Hebeis, 2025, *p.* 42ff.]. Thus, traditional machine learning-based classification systems such as SVMs or random forests should still be tested, especially if no large-scale training datasets with labels are available. The lessons from Chapter 5, from the perspective of classification, are once again mostly related to the efficiency factor as well as the ease of use and documentation of the widely researched text classification NLP task, see Section 8.3. Finally, combined approaches, e.g., stacking, see Section 8.5 and Wolpert [1992], or boosting, see Section 2.2.2 and Freund et al. [1996], are prominent representatives of hybrid classification systems, allowing the application of several different techniques to potentially improve upon the performance of single classifiers, however at the expense of additional runtime and energy consumption.

Recommendations for Text Segmentation

Text segmentation as one of the smaller research areas analyzed in this thesis has not received the same attention from the NLP research and practice as text classification or text summarization use cases. Nevertheless, even for text segmentation, the impact of LLMs has been noticed, see, e.g., Gong et al. [2022] or Alkhalil et al. [2025]. While text segmentation was not a part of the SLR in Jegan and Henrich [2025a], research as part of the case analysis from Chapter 9 has shown traditional approaches that are still in use today. As part of chunking strategies, fixed-size segmentation [Chalkidis et al., 2022; Dai et al., 2019] is still present in research. The statistical approach TextTiling by Hearst [1997] was identified as a tradi-

tional technique and was directly compared to the LLM GPT-4o. Despite the difficulty in evaluating text segmentation approaches, see Section 3.4 and Section 9.3, comparable qualitative performance was found for TextTiling in comparison to the LLM, accompanied by efficiency advantages by several orders of magnitude for the traditional model. In reviewing Chapter 5, efficiency as well as adaptability of TextTiling and similar techniques are the key characteristics of traditional models. Adaptability can be achieved through parametrization, see Section 9.3, since the parameters directly influence the quantitative output, i.e., the size of segments, and can also provide explainable and reproducible results.

Recommendations for Text Simplification

Text simplification, similarly to the next NLP task of text summarization, and in contrast to the previous mentioned tasks in this section, produces a new text based on the source document. While both can be abstractive in nature, extractive techniques can only be applied for text summarization, see below in this section. Text simplification procedures, as discussed in Section 3.5, can target different properties of a text, ranging from lexical, syntactic to semantic adjustments, however a purely extractive approach cannot be applied. The SLR from Jegan and Henrich [2025a], see Section 4.1, was only able to retrieve one publication on text simplification using traditional techniques, and in that case from North et al. [2023], lexical complexity was measured using traditional approaches, so no full simplification system was presented there. However, in the updated study for publications from 2024, after the SLR from Jegan and Henrich [2025a] was concluded, a text simplification system using rule-based approaches for syntactic simplifications was retrieved, see Keerthan Kumar et al. [2024]. Still, even this publication is limited through its focus on only one simplification dimension. Traditional simplification procedures therefore are only able to target one of the simplification characteristics: Lexical, syntactic or semantic. In contrast, modern systems use a breadth of simplification processes, as analyzed through the German simplification approach introduced in Section 7.1.1 from Fruth [2022] and Fruth et al. [2024]. There, reinforcement learning was used to target parameters such as fluency, salience and simplicity all at once through an unsupervised simplification task on paragraph-level German texts. As a baseline a pivot model was introduced in the simplification approach from Fruth [2022], with a pipeline consisting of translating the text into English, simplifying the English texts using the original technique from Laban et al. [2021] and translating the simplified versions back into English, which provided competitive results, see Sections 7.3.2 and 8.5. In a similar vein, simplification can also be approached as a monolingual

translation task, from the complex source text to the simplified version, see Al-Thanyyan and Azmi [2021]. In the end, text simplification scenarios benefit from the abstractive capabilities of transformer-based approaches much more directly than the previously mentioned tasks from this chapter. Traditional models should therefore only be applied in a limited manner for simplification purposes, e.g., if only lexical simplifications are desired, since the generative nature of this task clearly favors LLM-based approaches.

Recommendations for Text Summarization

Text summarization, as one of the premier use cases of LLMs, has received much attention even before the rise of neural networks and the interest has not waned since. Nevertheless, extractive approaches, which have been identified in the SLR from Jegan and Henrich [2025a], see Section 4.1, were still retrieved in ten publications, mostly as part of baselines for comparing new techniques with known approaches. Whereas some publications present extractive techniques as part of the summarization pipeline, e.g., by pre-ranking with a BERT-based system followed by an extractive selection process in Licari et al. [2023], others only use the extractive techniques as baselines, e.g., in Atri et al. [2023] or Du et al. [2023], which trail behind the more complex modern generative text summarization techniques. In general, however, text summarization is primarily approached through LLMs in recent years, due to the prevalence of LLM-based systems for this NLP task and the human-like performance achieved by modern text summarization systems that surpass traditional systems, which, again, are mostly applied as baselines [Zhang et al., 2024b]. Therefore, the role of traditional models has diminished for text summarization purposes, but niche or specialized use cases for traditional and extractive text summarization systems still remain. One such example concerns use cases, in which the prevention of hallucinations is a hard requirement, e.g., in critical domains such as law or medicine.[233] The explainability, i.e., how the sentences of a summary were

[233] Unfortunately, hallucinations and fabrications generated by LLMs have already been noticed in judicial procedures, see, e.g., the study on this topic in Latif [2025] or in `https://arstechnica.com/tech-policy/2025/05/judge-initially-fooled-by-fake-ai-citations-nearly-put-them-in-a-ruling/`.

chosen and why they were selected,[234] and reproducibility are two factors from Chapter 5 that would be applicable to traditional models in addition to efficiency advantages for both runtime and energy consumption, see Section 8.1. Hybrid scenarios, as mentioned by Licari et al. [2023] with pre-ranking based on BERT-based models for extractive summaries, would fall short in efficiency studies, but would bridge the gap between limitations of extractive approaches and still retain the absence of hallucinations. Furthermore, for languages and domains with limited training data, text summarization approaches based on statistical or graph-based techniques, which do not require large-scale parallel training datasets, would provide a clear target for traditional models. There, smaller datasets would suffice for applying extractive techniques that are less complex and would provide an alternative to LLMs, which suffer from problems such as hallucinations and domain shift in the generated texts, if the provided task was not represented in the training data of the LLM [Afzal et al., 2023].

Recommendations for a specific use case depend on the task at hand and parameters such as available data, domain, language, available hardware and others. In general, perspectives such as efficiency, explainability and reproducibility, if and when they are required for a respective task, should be considered in a constructive and purposeful manner within the project plan as well as other dimensions, see Section 5.2, referencing research question 4. Lastly, NLP tasks in a near uniform manner have shifted in research and practice towards the use of LLMs, with text simplification and text summarization in many cases rightfully so due to markedly improved performance compared to prior systems, whereas other, more restricted use cases – as shown earlier in this chapter in Sections 10.2 to 10.4 such as information extraction, text classification or text segmentation, can still benefit greatly from the use of traditional models. Therefore, research question 3 can be confirmed for the techniques listed in this chapter, however only after considering the respective use case and an analysis of goals, circumstances and other features, which differ depending on each task.

[234] It has to be noted that explainability for larger language models is not as clear as for extractive approaches. However, one example, described in Section 7.4.2, regarding the training datasets of a language model showcases the link between model and training data. Since a dataset consisting of both text summarization and text simplification data was used in fine-tuning the mLongT5 model applied in [Dal Cin, 2025, *p.* 42] for a multilingual text summarization study, overall simpler sentences – due to the inclusion of text simplification data during training – were generated by this model for the experiments in German when compared to the other languages. Therefore, the impact of training data should be noted and considered when choosing a pre-trained or fine-tuned model.

11 Conclusion and Outlook

The role of text as a medium and communication device is ever growing in everyday life, business, research and as part of rising AI advances. NLP has been impacted in broad and sweeping ways by the introduction of LLMs, which, however, was also accompanied by several limitations and drawbacks. This thesis addresses the diminished role of older techniques, that have been sidelined by LLMs, and presents potential benefits of the continued use or even reemergence of such models, which have been referred to as *traditional models* in this thesis. Answers to the four research questions, see the definition of those research questions in Section 1.1, were given based on extensive literature study, innovative experimentation and contextualization of different perspectives regarding NLP applications.

Result of Research Question 1

Research question 1 – relating to the current usage of traditional models in NLP applications and research – was analyzed in Section 4.1 through an SLR [Jegan and Henrich, 2025a] for five NLP tasks: Information and relation extraction, text classification, text simplification and text summarization. For all tasks, usage of traditional models in recent NLP research could be observed for 2023 and 2024, confirming research question 1 despite the apparent dominance of LLMs.

Result of Research Question 2

The second research question 2 – relating to the competitiveness of traditional models in comparison to modern methods – has also been answered by the SLR, which classified the respective traditional techniques for each publication as to their role within the publication, i.e., if they were just part of a pipeline, part of the core method or used for comparison purposes. More than a third of the retrieved papers from 2023 across all observed tasks evaluated traditional techniques with competitive results compared to other, more modern techniques. By reviewing selected use cases in Chapter 7, additional application scenarios were identified that can and should still be processed by using traditional techniques, specifically information extraction, see Section 7.3.1, text classification, see Section 7.3.2, and entity matching, see Section 7.4.2. Lastly, a dedicated case analysis was conducted by particularly considering out-of-domain lessons from Chapter 5 for a text segmentation use case, see

Chapter 9 as well as Jegan and Henrich [2025b]. This study further confirms the ongoing relevancy of traditional methods for more constrained applications such as text segmentation.

Result of Research Question 3

In surveying the ongoing challenges of NLP in Chapter 6 different problematic dimensions of current methods in particular such as hallucinations, reproducibility, explainability and efficiency were identified. These challenges can nearly unanimously be approached in a more direct manner by using traditional techniques. The adaptation of methods for non-English use cases and lacking metrics for evaluation purposes, as defined in the same chapter in Sections 6.2 and 6.3 respectively, are, however, a limiting factor. These restrictions affect older and more modern techniques in the same manner. To further illustrate the listed benefits that were desired for research question 3 – relating to potential benefits of using traditional models – Chapter 8 presents a detailed analysis of efficiency. Both runtime in Section 8.1.1 and energy consumption in Section 8.1.2 were discussed, as well as reproducibility in Section 8.2, when applying traditional techniques. Other advantages are collected in Sections 8.3 to 8.5, i.e., explainability, documentation and the potential of hybrid systems. A further reference towards the beneficial use of traditional techniques is shown once more through the case analysis from Chapter 9. The collected benefits of using the traditional model in the case analysis on a text segmentation problem include efficiency in both runtime and energy consumption, adaptability as well as budget while also achieving qualitatively competitive results with the tested LLM, see the evaluation in Section 9.3.

Result of Research Question 4

Research question 4 – relating to the systematic approach that can potentially benefit the selection of methods for NLP tasks – was approached by surveying the domains of engineering and design theory, discussed in Chapter 5. The lessons from this chapter provide the basis for a systematic guidance towards choosing appropriate software for NLP problems, e.g., by applying lessons from layered development and construction processes, see Figure 5.3, or by applying the selection criteria for a given NLP task from Table 5.1. The collected lessons and recommendations were applied in Section 9.2 through realizing the text segmentation task. Lastly, the discussion regarding applicability of different approaches from Chapter 10 was conducted with the systematic approach from the domains outside of NLP in mind. First, from the perspective of the techniques themselves, Sections 10.1 to 10.4 show for which use cases these techniques can be applied. To name an example, extractive or hybrid approaches can be used for summaries, in which hallucinations must not

occur, and when efficiency or explainability are a key factor in processing the source text. Second, recommendations for techniques from the use cases' point of view are presented in Section 10.5. These recommendations include, e.g., the use of rule-based approaches for clearly defined information extraction use cases, SVMs or Naive Bayes classifiers for classification tasks with limited training data or a reconsideration of older, statistical approaches for niche use cases such as text segmentation, see also Jegan [2026].

Synopsis of Research Questions

In answering the four research questions, traditional techniques were presented through various perspectives. The occurrence of traditional techniques was observed in today's NLP research as well as their role in current NLP publications, with competitive performance in certain use cases and results that are trailing behind in others. Furthermore, benefits of using traditional techniques such as efficiency, reproducibility, explainability, adaptability and reduced resource requirements in contrast to modern methods were collected. An out-of-domain point of view was applied on NLP problems in general through a systematic approach, which led to discussing the applicability of traditional models for specific NLP tasks and on a separate use case, i.e., text segmentation.

Reach of LLMs

The outlook for future NLP developments, especially considering the advances through LLMs since the release of the first publicly available version of ChatGPT in late 2022, is uncertain. Predictions range from the arrival of an Artificial General Intelligence, in contrast to more limited current AI, all the way to a bursting economic bubble akin to the dot-com bubble of the late 1990s.[235] It is certain that the prevalence of LLMs is currently, as of 2025, ever increasing, which has led to new observations and perhaps unintended side effects. The integration of summaries on the SERP of search engines is one such case. First, decreased visits have been noted for the websites listed in the search results, since the included AI-provided summaries already present the answer for the respective query to the user in many cases.[236] For site providers and news outlets, decreased visits and clicks lead to fewer impressions and less ad revenue. This development, in turn, could result in less money spent on ads within search providers – creating an interconnected problem for both search providers and website providers. Second, the integrated summary can contain hal-

[235] The potential of a speculative AI bubble has been noted in various articles across different domains apart from NLP, such as philosophy [Floridi, 2024], social sciences [Bohner and Vertesi, 2025] and as part of a position paper of the German Gesellschaft für Informatik [Steller, 2025].

[236] See, e.g., `https://arstechnica.com/ai/2025/07/research-shows-google-ai-overviews-reduce-website-clicks-by-almost-half/`.

lucinations and factual errors. Even though references are included and linked beside the generated summary, the stated facts are, depending on the user, taken at face-value.[237]

Timeliness of this Thesis

The advances in NLP, mainly driven by the development of LLMs in recent years, are certainly impressive in many tasks. The current NLP landscape is a fast-moving and developing research field with advances in quick succession and, in particular for LLMs, characterized by the release of newer, larger and computationally more demanding models in an often monthly manner by the large AI providers such as Anthropic, Google or OpenAI.[238] Therefore, a critical aspect of the contribution of this thesis is the timeliness of the analysis presented here. More precisely, ongoing developments underscore the timeliness and importance of a critical analysis that addresses when the use of LLMs is warranted and when traditional approaches are adequate. An assessment of LLMs for current NLP tasks requires a discussion of their limitations.

Limitations of Current Systems

One such limitation comprises accurately rating the performance through the use of benchmarks, which are susceptible to benchmark data contamination, see Section 7.1.2. In general, the problem of accurately evaluating models through a purely quantitative manner and through simple scores is an ongoing issue, discussed in detail in Sections 6.2 and 7.2. Related to the issue of evaluating models are explainability and reproducibility issues, collected in Chapters 6 and 7, which are somewhat limited for modern methods in contrast to traditional techniques, see the benefits of the latter in Chapter 8. Despite these shortcomings, all attention of current NLP use and research is placed on LLMs, which is why alternatives, e.g., in the form of traditional techniques, are often ignored. This thesis helps to overcome this bias with a fair scientific comparison and discussion, giving traditional models an appropriate assessment including their applicability in certain use cases. As a plea for future work, research of efficient and more restrained models should continue. The appropriate, considerate and purposeful use of modern systems, which naturally are represented by LLMs in the research field of Natural Language Processing, will continue to be of importance, not only as shown in this thesis but also by others, see, e.g., Bjarnason [2024] or Steller [2025].

[237] See, e.g., `https://www.thetimes.com/uk/technology-uk/article/google-ai-overviews-aio-wrong-vs32029z6/`.

[238] Although efforts to produce more efficient LLMs have increased in recent years, the trend of larger models is still ongoing, and even though inference might be somewhat more efficient, the training process continues to be more extensive and thus demanding ever more time and energy [Emer et al., 2025; Schwartz et al., 2020; Stojkovic et al., 2024].

A simple recommendation or classification into better and worse systems, as with many aspects of life in general into black and white, is impossible. It always depends on the concrete task at hand, the available data and its quality, resource constraints and other facets. Many of these factors were discussed as part of the analysis of out-of-domain perspectives from engineering and design theory in Chapter 5 and especially regarding different selection criteria of NLP tasks in Section 5.2. As a brief concluding distinction, generative tasks like simplification or summarization clearly benefit from the use of modern transformer-based models, as shown in Chapter 7 and succinctly in Section 10.5. Other use cases, such as information extraction, entity matching, text classification or text segmentation, as shown in Chapters 9 and 10, however, still can be applied successfully with traditional techniques in many cases. These techniques offer additional advantages such as efficiency, explainability and reproducibility, particularly in contrast to modern LLMs, see Section 10.5. Although, as mentioned, these recommendations should always be contextualized properly within the respective use case.

Traditional vs. Modern Systems

Several directions for future research can be proposed. The case-based nature of this thesis with a focus on traditional methods lends itself to different follow-up studies. The different perspectives in Chapter 10 could be seen as a starting point: Focusing on the techniques or on the tasks. From the first perspective, traditional models could be applied within the text classification realm in resource-constrained application scenarios, see Section 10.5. For instance, SVM, random forest or decision tree classifiers could be used for different types of tasks that need to identify some parts of a text – similar to the information extraction use case presented in Section 7.3.1 – or to simply classify the input data – see Section 7.3.2. Within this approach, a tiered system could be employed, with traditional models as a starting point. If a certain threshold, e.g., a classification confidence score, is reached, the approach would terminate and return the result. If the threshold is not met, more advanced neural network-based methods or LLMs could be applied. Further steps such as voting between different systems, like stacking algorithms, see Wolpert [1992], or hybrid systems with case-based processing can be imagined. The second perspective – from the point of view of NLP tasks – could include a more detailed analysis of niche use cases such as text segmentation, as presented in Chapter 9, or other less researched tasks, e.g., text simplification. As the SLR from Section 4.1 showed, studies in text simplification are scarce, and hybrid approaches, similar to the ones presented in Section 8.5, are a promising direction for future research there. One such approach could

Outlook

include traditional models as an initial step for a focused text simplification, purely on a lexical or syntactical dimension, before a generative approach would produce a simplified text based on the original input data and the results provided by the traditional methods. In general, other studies into rule-based approaches, which can be a worthwhile consideration for narrow tasks, or evaluation parameters such as efficiency, explainability or reproducibility, especially after the emergence of LLMs, also merit further research.

In Closing

Finally, it should be stated that this thesis should not be seen as an indictment of LLMs. Instead, it should be interpreted as a call of consideration for other, perhaps forgotten techniques – namely traditional techniques – that may offer an alternative or even provide benefits when compared to modern systems. The invention of LLMs, while impressive in many tasks and touching many aspects of our daily life, is not as all-encompassing as the invention of writing by humankind, as articulated in the introductory quote by Carl Sagan. The development of language, both spoken and written, is a process that has been ongoing for thousands of years and is still evolving today. Likewise, AI, as a new communication partner, must be constantly re-examined and further developed. In surveying the impact of LLMs, a change was noticed in how people work with written text, how text is processed and what tools are available to even laypeople. The thorough and considerate use of these tools, especially for the NLP community as developers and even ambassadors of them, is an ongoing challenge and will certainly be relevant for years to come. Efficiency, explainability and reproducibility are important features of NLP techniques in general, and traditional models exhibit them near unanimously. The results of this thesis demonstrate that traditional models remain relevant for certain use cases and that a careful consideration of each NLP task and its context is – and will be – essential.

List of Abbreviations

ACL	Association for Computational Linguistics
ACM	Association for Computing Machinery
AI	Artificial Intelligence
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
BGB	Bürgerliches Gesetzbuch
BLEU	Bilingual Evaluation Understudy
BOL	Beginning of Life
BoW	Bag of Words
CLI	Command-Line Interface
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DUC	Document Understanding Conference
EDU	Elementary Discourse Unit
EOL	End of Life
FKGL	Flesch-Kincaid-Grade-Level
FLOPS	Floating-Point Operations Per Second
FRE	Flesch-Reading-Ease
FSG	Finite State Grammar
GAN	Generative Adversarial Network
GND	Gemeinsame Normdatei
GNN	Graph Neural Network
GPU	Graphics Processing Unit
GPT	Generative Pre-trained Transformer
GTC	General Terms and Conditions

HMM	Hidden Markov Model
HTML	Hypertext Markup Language
KNN	K-Nearest Neighbors
LIME	Local Interpretable Model-agnostic Explanations
LLM	Large Language Model
LSA	Latent Semantic Analysis
LSTM	Long Short-Term Memory
METEOR	Metric for Evaluation of Translation with Explicit Ordering
MOL	Middle of Life
MRR	Mean Reciprocal Rank
MTEB	Massive Text Embedding Benchmark
NDCG	Normalized Discounted Cumulative Gain
NER	Named-Entity Recognition
NLG	Natural Language Generation
NLP	Natural Language Processing
NLTK	Natural Language ToolKit
PaaS	Platform as a Service
PEP	Produktentstehungsprozess
PoS	Part of Speech
RAG	Retrieval-Augmented Generation
RAM	Random Access Memory
RLHF	Reinforcement Learning from Human Feedback
RNN	Recurrent Neural Networks
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
SARI	System output Against References and against the Input sentence
SCST	Self-Critical Sequence Training
SERP	Search Engine Result Page
SLR	Structured Literature Review
SVM	Support Vector Machine
TAC	Text Analysis Conference

TF-IDF	Term Frequency-Inverse Document Frequency
TPS	Tokens Per Second
URL	Uniform Resource Locator
VRAM	Video Random Access Memory
XAI	eXplainable Artificial Intelligence
XML	eXtensible Markup Language

List of Figures

List of Tables

Bibliography

[1] Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*. (Cited on page 64.)

[2] Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1985). A Learning Algorithm for Boltzmann Machines. *Cognitive Science*, 9(1):147–169. (Cited on page 104.)

[3] Afzal, A., Vladika, J., Braun, D., and Matthes, F. (2023). Challenges in Domain-Specific Abstractive Summarization and How to Overcome Them. In *15th International Conference on Agents and Artificial Intelligence, ICAART 2023*, pages 682–689. SCITEPRESS. (Cited on page 232.)

[4] Aggarwal, C. C. and Zhai, C. (2012a). A Survey of Text Classification Algorithms. *Mining Text Data*, pages 163–222. (Cited on pages 27, 34, 35, and 223.)

[5] Aggarwal, C. C. and Zhai, C. (2012b). A Survey of Text Clustering Algorithms. *Mining Text Data*, pages 77–128. (Cited on page 27.)

[6] Aiyappa, R., An, J., Kwak, H., and Ahn, Y.-y. (2023). Can We Trust the Evaluation on ChatGPT? In *The 61st Annual Meeting of the Association for Computational Linguistics*. (Cited on page 64.)

[7] Akter, M., Bansal, N., and Karmaker, S. K. (2022). Revisiting Automatic Evaluation of Extractive Summarization Task: Can We Do Better Than ROUGE? In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1547–1560. (Cited on page 47.)

[8] Al-Jarrah, O. Y., Yoo, P. D., Muhaidat, S., Karagiannidis, G. K., and Taha, K. (2015). Efficient Machine Learning for Big Data: A Review. *Big Data Research*, 2(3):87–93. (Cited on page 61.)

[9] Al-Thanyyan, S. S. and Azmi, A. M. (2021). Automated Text Simplification: A Survey. *ACM Computing Surveys (CSUR)*, 54(2):1–36. (Cited on pages 39, 40, 42, 43, 57, and 231.)

[10] Alahmari, S. S., Goldgof, D. B., Mouton, P. R., and Hall, L. O. (2020). Challenges for the Repeatability of Deep Learning Models. *IEEE Access*, 8:211860–211868. (Cited on page 186.)

[11] Alkhalil, B. F., Zhuang, Y., Mursi, K. T., and Aseeri, A. O. (2025). Enhancing Trust Factor Identification in E-Commerce: The Role of Text Segmentation and Factor Extraction With Transformer Models. In *2025 IEEE 4th International Conference on AI in Cybersecurity (ICAIC)*, pages 1–8. IEEE. (Cited on pages 38 and 229.)

[12] Ananiadou, S., McNaught, J., Thompson, P., Rehm, G., and Uszkoreit, H. (2012). *The English Language in the Digital Age*. Springer. (Cited on page 101.)

[13] Anschütz, M., Mosca, E., and Groh, G. (2024). Simpler Becomes Harder: Do LLMs Exhibit a Coherent Behavior on Simplified Corpora? In *Proceedings of the Workshop on DeTermIt! Evaluating Text Difficulty in a Multilingual Context@ LREC-COLING 2024*, pages 185–195. (Cited on pages 43, 56, 57, and 59.)

[14] Aroyehun, S. T., Angel, J., Alvarez, D. A. P., and Gelbukh, A. (2018). Complex Word Identification: Convolutional Neural Network vs. Feature Engineering. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 322–327. (Cited on pages 42, 56, 57, and 59.)

[15] Artetxe, M., Aldabe, I., Agerri, R., Perez-De-Viñaspre, O., and Soroa, A. (2022). Does Corpus Quality Really Matter for Low-Resource Languages? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7383–7390. (Cited on page 101.)

[16] Arvan, M., Pina, L., and Parde, N. (2022). Reproducibility in Computational Linguistics: Is Source Code Enough? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2350–2361. (Cited on page 19.)

[17] Asano, H., Mizumoto, T., and Inui, K. (2017). Reference-Based Metrics Can Be Replaced With Reference-Less Metrics in Evaluating Grammatical Error Correction Systems. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 343–348. (Cited on page 98.)

[18] Atri, Y. K., Goyal, V., and Chakraborty, T. (2023). Fusing Multimodal Signals on Hyper-Complex Space for Extreme Abstractive Text Summarization (TL; DR) of Scientific Contents. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3724–3736. (Cited on pages 54 and 231.)

[19] Aumiller, D., Chouhan, A., and Gertz, M. (2022). EUR-Lex-Sum: A Multi- and Cross-Lingual Dataset for Long-Form Summarization in the Legal Domain. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7626–7639. (Cited on pages 45,

101, 192, 195, and 220.)

[20] Aumiller, D., Fan, J., and Gertz, M. (2023). On the State of German (Abstractive) Text Summarization. In König-Ries, B., Scherzinger, S., Lehner, W., and Vossen, G., editors, *BTW 2023*, volume P-331 of *LNI*, pages 195–220. Gesellschaft für Informatik e.V. (Cited on pages 19, 91, 98, 132, and 138.)

[21] Aumiller, D. and Gertz, M. (2022). Klexikon: A German Dataset for Joint Summarization and Simplification. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 2693–2701. (Cited on pages 153 and 158.)

[22] Bach, N. and Badaskar, S. (2007). A Review of Relation Extraction. *Literature Review for Language and Statistics II*, 2:1–15. (Cited on page 33.)

[23] Backus, J. W. (1959). The Syntax and the Semantics of the Proposed International Algebraic Language of the Zurich ACM-GAMM Conference. In *ICIP Proceedings*, pages 125–132. (Cited on pages 7 and 21.)

[24] Baez, A. and Saggion, H. (2023). LSLlama: Fine-Tuned LLaMA for Lexical Simplification. In *Proceedings of the Second Workshop on Text Simplification, Accessibility and Readability*, pages 102–108. (Cited on pages 43, 56, 57, and 59.)

[25] Bahdanau, D., Chorowski, J., Serdyuk, D., Brakel, P., and Bengio, Y. (2016). End-to-End Attention-Based Large Vocabulary Speech Recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4945–4949. IEEE. (Cited on page 15.)

[26] Bai, Z., Wang, P., Xiao, T., He, T., Han, Z., Zhang, Z., and Shou, M. Z. (2024). Hallucination of Multimodal Large Language Models: A Survey. *arXiv preprint arXiv:2404.18930*. (Cited on page 111.)

[27] Baker, S. and Kanade, T. (2000). Hallucinating Faces. In *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)*, pages 83–88. IEEE. (Cited on page 93.)

[28] Banerjee, S. and Lavie, A. (2005). METEOR: An Automatic Metric for MT Evaluation With Improved Correlation With Human Judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72. (Cited on page 47.)

[29] Bao, G. and Zhang, Y. (2023). A General Contextualized Rewriting Framework for Text Summarization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:1624–1635. (Cited on pages 54, 56, 57, and 59.)

[30] Baroni, M., Matiasek, J., Trost, H., et al. (2002). Predicting the Components of German Nominal Compounds. In *ECAI*, volume 2002, page

15th. (Cited on page 141.)

[31] Barzilay, R. and Lapata, M. (2008). Modeling Local Coherence: An Entity-Based Approach. *Computational Linguistics*, 34(1):1–34. (Cited on page 99.)

[32] Bauer, E. and Kohavi, R. (1999). An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine Learning*, 36(1):105–139. (Cited on page 191.)

[33] Bayes, T. (1763). An Essay Towards Solving a Problem in the Doctrine of Chances. *Philosophical Transactions of the Royal Society of London*, 53:370–418. (Cited on page 10.)

[34] Beeferman, D., Berger, A., and Lafferty, J. (1999). Statistical Models for Text Segmentation. *Machine Learning*, 34:177–210. (Cited on pages 38, 56, and 58.)

[35] Belz, A., Agarwal, S., Shimorina, A., and Reiter, E. (2021). A Systematic Review of Reproducibility Research in Natural Language Processing. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 381–393. Association for Computational Linguistics. (Cited on page 104.)

[36] Bender, E. M. (2009). Linguistically Naïve != Language Independent: Why NLP Needs Linguistic Typology. In *Proceedings of the EACL 2009 Workshop on the Interaction Between Linguistics and Computational Linguistics: Virtuous, Vicious or Vacuous?*, pages 26–32. (Cited on page 100.)

[37] Bengio, Y. and Bengio, S. (1999). Modeling High-Dimensional Discrete Data With Multi-Layer Neural Networks. *Advances in Neural Information Processing Systems*, 12. (Cited on page 13.)

[38] Bengio, Y., Ducharme, R., and Vincent, P. (2000). A Neural Probabilistic Language Model. *Advances in Neural Information Processing Systems*, 13. (Cited on page 13.)

[39] Berger, A., Della Pietra, S. A., and Della Pietra, V. J. (1996). A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71. (Cited on page 11.)

[40] Biagioli, C., Francesconi, E., Passerini, A., Montemagni, S., and Soria, C. (2005). Automatic Semantics Extraction in Law Documents. In *Proceedings of the 10th International Conference on Artificial Intelligence and Law*, pages 133–140. (Cited on page 146.)

[41] Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing With Python: Analyzing Text With the Natural Language Toolkit.* " O'Reilly Media, Inc.". (Cited on pages 187 and 204.)

[42] Bishop, C. M. and Nasrabadi, N. M. (2006). *Pattern Recognition and Machine Learning*, volume 4. Springer. (Cited on pages 11 and 36.)

[43] Bjarnason, B. (2024). *The Intelligence Illusion: A Practical Guide to the Business Risks of Generative AI*. (Cited on page 236.)

[44] Blackwell, R. E., Barry, J., and Cohn, A. G. (2024). Towards Reproducible LLM Evaluation: Quantifying Uncertainty in LLM Benchmark Scores. *arXiv preprint arXiv:2410.03492*. (Cited on page 156.)

[45] Bohner, J. and Vertesi, J. (2025). Towards a Socioeconomics of Hype: Hype Dynamics and Symbolic Boundary Work within the Speculative AI Bubble. *Social Science Computer Review*, page 08944393251361935. (Cited on page 235.)

[46] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching Word Vectors With Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146. (Cited on pages 15, 142, and 166.)

[47] Bougouin, A., Barreaux, S., Romary, L., Boudin, F., and Daille, B. (2016). TermITH-Eval: A French Standard-Based Resource for Keyphrase Extraction Evaluation. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC)*, pages 1924–1927. (Cited on page 152.)

[48] Brandow, R., Mitze, K., and Rau, L. F. (1995). Automatic Condensation of Electronic Publications by Sentence Selection. *Information Processing & Management*, 31(5):675–685. (Cited on page 45.)

[49] Breiman, L. (1996). Bagging Predictors. *Machine Learning*, 24:123–140. (Cited on pages 9, 10, 35, 56, and 59.)

[50] Breiman, L. (2001). Random Forests. *Machine Learning*, 45:5–32. (Cited on pages 10, 107, and 191.)

[51] Brin, S. and Page, L. (1998). The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117. (Cited on pages 26 and 45.)

[52] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language Models Are Few-Shot Learners. *Advances in Neural Information Processing Systems*, 33:1877–1901. (Cited on pages 17, 21, and 62.)

[53] Campos, R., Mangaravite, V., Pasquali, A., Jorge, A. M., Nunes, C., and Jatowt, A. (2018). YAKE! Collection-Independent Automatic Keyword Extractor. In *European Conference on Information Retrieval*, pages 806–810. Springer. (Cited on page 122.)

[54] Cao, Q., Balasubramanian, A., and Balasubramanian, N. (2020). Towards Accurate and Reliable Energy Measurement of NLP Models. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 141–148. (Cited on page 62.)

[55] Cartuyvels, R., Spinks, G., and Moens, M.-F. (2021). Discrete and Continuous Representations and Processing in Deep Learning: Looking Forward. *AI Open*, 2:143–159. (Cited on page 6.)

[56] Chakraborty, N., Ornik, M., and Driggs-Campbell, K. (2025). Hallucination Detection in Foundation Models for Decision-Making: A Flexible Definition and Review of the State of the Art. *ACM Computing Surveys*. (Cited on page 116.)

[57] Chalkidis, I., Dai, X., Fergadiotis, M., Malakasiotis, P., and Elliott, D. (2022). An Exploration of Hierarchical Attention Transformers for Efficient Long Document Classification. *arXiv preprint arXiv:2210.05529*. (Cited on pages 38 and 229.)

[58] Chandrasekar, R., Doran, C., and Bangalore, S. (1996). Motivations and Methods for Text Simplification. In *COLING 1996 Volume 2: The 16th International Conference on Computational Linguistics*. (Cited on page 41.)

[59] Chen, C., Yin, Y., Shang, L., Jiang, X., Qin, Y., Wang, F., Wang, Z., Chen, X., Liu, Z., and Liu, Q. (2022). Bert2BERT: Towards Reusable Pretrained Language Models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2134–2148. (Cited on pages 149, 164, 170, and 171.)

[60] Chen, H., Sun, Y., Zhang, M., and Zhang, M. (2023a). Automatic Noise Generation and Reduction for Text Classification. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:139–150. (Cited on pages 53 and 229.)

[61] Chen, L., Zaharia, M., and Zou, J. (2023b). How Is ChatGPT's Behavior Changing Over Time? *arXiv preprint arXiv:2307.09009*. (Cited on page 64.)

[62] Chen, S. (2024). Research on Nvidia Investment Strategies and Analysis. *Highlights in Business, Economics and Management*, 24:2234–2240. (Cited on page 72.)

[63] Cheng, J. (2016). Long Short-Term Memory-Networks for Machine Reading. *arXiv preprint arXiv:1601.06733*. (Cited on page 16.)

[64] Chiticariu, L., Krishnamurthy, R., Li, Y., Reiss, F., and Vaithyanathan, S. (2010). Domain Adaptation of Rule-Based Annotators for Named-Entity Recognition Tasks. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1002–1012. (Cited on page 8.)

[65] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation.

arXiv preprint arXiv:1406.1078. (Cited on page 15.)

[66] Chomsky, N. (1956). Three Models for the Description of Language. *IRE Transactions on Information Theory*, 2(3):113–124. (Cited on pages 7, 21, 24, and 25.)

[67] Chomsky, N. (1976). *Syntactic Structures*. Mouton de Gruyter. (Cited on page 25.)

[68] Chuang, W. T. and Yang, J. (2000). Extracting Sentence Segments for Text Summarization: A Machine Learning Approach. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 152–159. (Cited on pages 56, 57, and 59.)

[69] Chung, H. W., Garcia, X., Roberts, A., Tay, Y., Firat, O., Narang, S., and Constant, N. (2023). UniMax: Fairer and More Effective Language Sampling for Large-Scale Multilingual Pretraining. In *The Eleventh International Conference on Learning Representations*. (Cited on page 100.)

[70] Cohan, A., Dernoncourt, F., Kim, D. S., Bui, T., Kim, S., Chang, W., and Goharian, N. (2018). A Discourse-Aware Attention Model for Abstractive Summarization of Long Documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621. (Cited on page 152.)

[71] Cohen, K. B., Xia, J., Zweigenbaum, P., Callahan, T. J., Hargraves, O., Goss, F., Ide, N., Névéol, A., Grouin, C., and Hunter, L. E. (2018). Three Dimensions of Reproducibility in Natural Language Processing. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, volume 2018, page 156. NIH Public Access. (Cited on page 63.)

[72] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural Language Processing (Almost) From Scratch. *Journal of Machine Learning Research*, 12:2493–2537. (Cited on page 14.)

[73] Conneau, A., Rinott, R., Lample, G., Williams, A., Bowman, S., Schwenk, H., and Stoyanov, V. (2018). XNLI: Evaluating Cross-Lingual Sentence Representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, page 2475. Association for Computational Linguistics. (Cited on page 102.)

[74] Cortes, C. (1995). Support-Vector Networks. *Machine Learning*. (Cited on pages 12, 21, and 225.)

[75] Cowie, J. and Lehnert, W. (1996). Information Extraction. *Communications of the ACM*, 39(1):80–91. (Cited on pages 30 and 32.)

[76] Cramer, J. S. (2002). The Origins of Logistic Regression. Technical report, Tinbergen Institute. (Cited on page 11.)

[77] Craswell, N. (2009). Mean Reciprocal Rank. (Cited on page 122.)

[78] Croft, W. B., Metzler, D., and Strohman, T. (2010). *Search Engines: Information Retrieval in Practice*, volume 520. Addison-Wesley Reading. (Cited on page 107.)

[79] Culotta, A. and Sorensen, J. (2004). Dependency Tree Kernels for Relation Extraction. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 423–429. (Cited on page 32.)

[80] Cunha, W., Viegas, F., França, C., Rosa, T., Rocha, L., and Gonçalves, M. A. (2023). A Comparative Survey of Instance Selection Methods Applied to Non-Neural and Transformer-Based Text Classification. *ACM Computing Surveys*, 55(13s):1–52. (Cited on pages 53, 56, 57, 58, and 60.)

[81] Dabre, R., Chu, C., and Kunchukuttan, A. (2020). A Survey of Multilingual Neural Machine Translation. *ACM Computing Surveys (CSUR)*, 53(5):1–38. (Cited on page 25.)

[82] Dadachev, B., Balinsky, A., and Balinsky, H. (2014). On Automatic Text Segmentation. In *Proceedings of the 2014 ACM Symposium on Document Engineering*, pages 73–80. (Cited on pages 37 and 39.)

[83] Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., and Salakhutdinov, R. (2019). Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics. (Cited on pages 38 and 229.)

[84] Dal Cin, G. (2025). *Multilingual Text Summarization Approaches - A Case Study on Generative and Extractive Methods*. Master's thesis, University of Bamberg. `https://doi.org/10.20378/irb-112942`. Supervisor: Jegan, R.; Reviewer: Henrich, A. (Cited on pages 6, 16, 24, 28, 56, 57, 100, 114, 118, 119, 129, 131, 134, 136, 137, 141, 144, 148, 151, 152, 153, 154, 156, 157, 158, 159, 170, 171, 173, 177, 179, 183, 185, 188, 221, and 232.)

[85] Dally, W. J., Keckler, S. W., and Kirk, D. B. (2021). Evolution of the Graphics Processing Unit (GPU). *IEEE Micro*, 41(6):42–51. (Cited on page 189.)

[86] de Maat, E., Krabben, K., and Winkels, R. (2010). Machine Learning Versus Knowledge Based Classification of Legal Texts. In *Legal Knowledge and Information Systems*, pages 87–96. IOS Press. (Cited on page 146.)

[87] De Vries, A. (2023). The Growing Energy Footprint of Artificial Intelligence. *Joule*, 7(10):2191–2194. (Cited on pages 214 and 215.)

[88] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41(6):391–407. (Cited on pages 12 and 13.)

[89] Deng, C., Zhao, Y., Tang, X., Gerstein, M., and Cohan, A. (2024). Investigating Data Contamination in Modern Benchmarks for Large Language Models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8698–8711. (Cited on page 116.)

[90] Derby, E., Schwaber, K., and Larsen, D. (2006). *Agile Retrospectives: Making Good Teams Great*. The pragmatic bookshelf. (Cited on pages 75 and 76.)

[91] Dernoncourt, F., Ghassemi, M., and Chang, W. (2018). A Repository of Corpora for Summarization. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. (Cited on pages 45, 70, and 152.)

[92] Devlin, J. (2018). BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*. (Cited on pages 16, 17, 21, 62, and 161.)

[93] Devlin, S. and Tait, J. (1998). The Use of a Psycholinguistic Database in the Simplification of Text for Aphasic Readers. *Linguistic Databases*, pages 161–173. (Cited on page 41.)

[94] Ding, N., Xu, G., Chen, Y., Wang, X., Han, X., Xie, P., Zheng, H.-T., and Liu, Z. (2021). Few-NERD: A Few-Shot Named Entity Recognition Dataset. In Zong, C., Xia, F., Li, W., and Navigli, R., editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3198–3213, Online. Association for Computational Linguistics. (Cited on pages 30, 56, and 60.)

[95] Dong, Z., Tang, T., Li, L., and Zhao, W. X. (2023). A Survey on Long Text Modeling With Transformers. *arXiv preprint arXiv:2302.14502*. (Cited on pages 38, 56, 58, and 60.)

[96] D'Silva, J. and Sharma, U. (2023). Impact of Similarity Measures in Graph-Based Automatic Text Summarization of Konkani Texts. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 22(2):1–16. (Cited on page 54.)

[97] Du, K., Lu, G., and Qin, K. (2023). An Extractive Text Summarization Based on Reinforcement Learning. In *Proceedings of the 2023 6th International Conference on Software Engineering and Information Management*, pages 19–25. (Cited on pages 55, 56, 60, and 231.)

[98] Du, M., Liu, N., and Hu, X. (2019). Techniques for Interpretable Machine Learning. *Communications of the ACM*, 63(1):68–77. (Cited on pages 106 and 223.)

[99] Dwivedi, R., Dave, D., Naik, H., Singhal, S., Omer, R., Patel, P., Qian, B., Wen, Z., Shah, T., Morgan, G., et al. (2023). Explainable AI (XAI): Core Ideas, Techniques, and Solutions. *ACM Computing Surveys*, 55(9):1–33. (Cited on page 163.)

[100] Eddine, M. K., Tixier, A., and Vazirgiannis, M. (2021). BARThez: A Skilled Pretrained French Sequence-to-Sequence Model. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9369–9390. (Cited on page 152.)

[101] Eisenstein, J. (2018). *Natural Language Processing*, volume 507. MIT Press. (Cited on pages 8, 11, 29, 35, 95, 96, and 102.)

[102] El-Kassas, W. S., Salama, C. R., Rafea, A. A., and Mohamed, H. K. (2021). Automatic Text Summarization: A Comprehensive Survey. *Expert Systems with Applications*, 165:113679. (Cited on pages 45, 46, and 47.)

[103] Elo, A. (2008). *The Rating of Chessplayers: Past and Present*. Ishi Press International. (Cited on page 123.)

[104] Elsworth, C., Huang, K., Patterson, D., Schneider, I., Sedivy, R., Goodman, S., Townsend, B., Ranganathan, P., Dean, J., Vahdat, A., et al. (2025). Measuring the Environmental Impact of Delivering AI at Google Scale. *arXiv preprint arXiv:2508.15734*. (Cited on page 183.)

[105] Emer, L., Mina, A., and Vandin, A. (2025). The Anatomy of Green AI Technologies: Structure, Evolution, and Impact. *arXiv preprint arXiv:2509.10109*. (Cited on pages 183 and 236.)

[106] Erkan, G. and Radev, D. R. (2004). LexRank: Graph-Based Lexical Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research*, 22:457–479. (Cited on pages 20, 22, 45, 56, 57, 59, 69, 70, 118, 148, 149, 164, 170, 171, 192, 220, and 222.)

[107] Etzioni, O., Banko, M., Soderland, S., and Weld, D. S. (2008). Open Information Extraction From the Web. *Communications of the ACM*, 51(12):68–74. (Cited on page 32.)

[108] Ezugwu, A. E., Ikotun, A. M., Oyelade, O. O., Abualigah, L., Agushaka, J. O., Eke, C. I., and Akinyelu, A. A. (2022). A Comprehensive Survey of Clustering Algorithms: State-of-the-Art Machine Learning

Applications, Taxonomy, Challenges, and Future Research Prospects. *Engineering Applications of Artificial Intelligence*, 110:104743. (Cited on page 27.)

[109] Fabbri, A. R., Kryściński, W., McCann, B., Xiong, C., Socher, R., and Radev, D. (2021). SummEval: Re-Evaluating Summarization Evaluation. *Transactions of the Association for Computational Linguistics*, 9:391–409. (Cited on page 137.)

[110] Falk, S., Ekchajzer, D., Pirson, T., Lees-Perasso, E., Wattiez, A., Biber-Freudenberger, L., Luccioni, S., and van Wynsberghe, A. (2025). More than Carbon: Cradle-to-Grave Environmental Impacts of GenAI Training on the Nvidia A100 GPU. *arXiv preprint arXiv:2509.00093*. (Cited on pages 109 and 110.)

[111] Fan, A., Bhosale, S., Schwenk, H., Ma, Z., El-Kishky, A., Goyal, S., Baines, M., Celebi, O., Wenzek, G., Chaudhary, V., et al. (2021). Beyond English-Centric Multilingual Machine Translation. *Journal of Machine Learning Research*, 22(107):1–48. (Cited on page 101.)

[112] Farmakiotou, D., Karkaletsis, V., Koutsias, J., Sigletos, G., Spyropoulos, C. D., and Stamatopoulos, P. (2000). Rule-Based Named Entity Recognition for Greek Financial Texts. In *Proceedings of the Workshop on Computational Lexicography and Multimedia Dictionaries (COMLEX 2000)*, pages 75–78. (Cited on page 9.)

[113] Faruqui, M. and Kumar, S. (2015). Multilingual Open Relation Extraction Using Cross-Lingual Projection. In Mihalcea, R., Chai, J., and Sarkar, A., editors, *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1351–1356, Denver, Colorado. Association for Computational Linguistics. (Cited on page 32.)

[114] Feldhusen, J., Grote, K.-H., et al. (2013). *Pahl/Beitz Konstruktionslehre*. Springer. (Cited on pages 67, 68, 69, 70, 71, 73, 74, 80, 86, 87, 88, 89, and 200.)

[115] Feng, X., Qin, B., and Liu, T. (2018). A Language-Independent Neural Network for Event Detection. *Science China Information Sciences*, 61:1–12. (Cited on pages 32, 56, 59, and 60.)

[116] Fields, J., Chovanec, K., and Madiraju, P. (2024). A Survey of Text Classification With Transformers: How Wide? How Large? How Long? How Accurate? How Expensive? How Safe? *IEEE Access*. (Cited on pages 36, 56, 57, and 59.)

[117] Flesch, R. (1948). A New Readability Yardstick. *Journal of Applied Psychology*, 32(3):221. (Cited on page 40.)

[118] Florescu, C. and Caragea, C. (2017). PositionRank: An Unsupervised Approach to Keyphrase Extraction From Scholarly Documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1115. (Cited on pages 122, 143, and 222.)

[119] Floridi, L. (2024). Why the AI Hype Is Another Tech Bubble. *Philosophy & Technology*, 37(4):128. (Cited on page 235.)

[120] Forman, G. (2003). An Extensive Empirical Study of Feature Selection Metrics for Text Classification. *Journal of Machine Learning Research*, 3(Mar):1289–1305. (Cited on page 35.)

[121] Fossey, E., Harvey, C., McDermott, F., and Davidson, L. (2002). Understanding and Evaluating Qualitative Research. *Australian & New Zealand Journal of Psychiatry*, 36(6):717–732. (Cited on page 95.)

[122] Francesconi, E. and Passerini, A. (2007). Automatic Classification of Provisions in Legislative Texts. *Artificial Intelligence and Law*, 15:1–17. (Cited on page 146.)

[123] Freiesleben, T. and König, G. (2023). Dear XAI Community, We Need to talk! Fundamental Misconceptions in Current XAI Research. In *World Conference on Explainable Artificial Intelligence*, pages 48–65. Springer. (Cited on page 190.)

[124] Freund, Y., Schapire, R. E., et al. (1996). Experiments With a New Boosting Algorithm. In *ICML*, volume 96, pages 148–156. Citeseer. (Cited on pages 9, 35, 56, 59, 191, 195, and 229.)

[125] Fruth, L. (2022). *An Approach Towards Unsupervised Text Simplification on Paragraph-Level for German Texts*. Master's thesis, University of Bamberg. `https://doi.org/10.20378/irb-112994`. Supervisor: Jegan, R.; Reviewer: Henrich, A. (Cited on pages 6, 24, 28, 43, 56, 60, 88, 112, 113, 129, 132, 133, 141, 144, 146, 147, 148, 154, 155, 156, 157, 159, 160, 192, 193, and 230.)

[126] Fruth, L., Jegan, R., and Henrich, A. (2024). An Approach Towards Unsupervised Text Simplification on Paragraph-Level for German Texts. In *Proceedings of the Workshop on DeTermIt! Evaluating Text Difficulty in a Multilingual Context @ LREC-COLING 2024*, pages 77–89. (Cited on pages 132, 155, and 230.)

[127] Gambhir, M. and Gupta, V. (2017). Recent Automatic Text Summarization Techniques: A Survey. *Artificial Intelligence Review*, 47(1):1–66. (Cited on pages 45 and 60.)

[128] Gatt, A. and Krahmer, E. (2018). Survey of the State of the Art in Natural Language Generation: Core Tasks, Applications and Evaluation. *Journal of Artificial Intelligence Research*, 61:65–170. (Cited on page 131.)

[129] Geroimenko, V. (2025). Key Challenges in Prompt Engineering. In *The Essential Guide to Prompt Engineering: Key Principles, Techniques, Challenges, and Security Risks*, pages 85–102. Springer. (Cited on page 156.)

[130] Ghinassi, I., Wang, L., Newell, C., and Purver, M. (2023). Lessons Learnt from Linear Text Segmentation: A Fair Comparison of Architectural and Sentence Encoding Strategies for Successful Segmentation. In *Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing*, pages 408–418. (Cited on pages 38, 56, 57, and 59.)

[131] Ghinassi, I., Wang, L., Newell, C., and Purver, M. (2024). Recent Trends in Linear Text Segmentation: A Survey. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3084–3095. (Cited on pages 56, 58, and 211.)

[132] Glaser, I., Moser, S., and Matthes, F. (2021). Summarization of German Court Rulings. In *Proceedings of the Natural Legal Language Processing Workshop 2021*, pages 180–189. (Cited on pages 148, 164, and 170.)

[133] Glaser, I., Scepankova, E., and Matthes, F. (2018). Classifying Semantic Types of Legal Sentences: Portability of Machine Learning Models. In *Legal Knowledge and Information Systems*, pages 61–70. IOS Press. (Cited on pages 144, 146, and 226.)

[134] Glavaš, G. and Štajner, S. (2013). Event-Centered Simplification of News Stories. In *Proceedings of the Student Research Workshop Associated With RANLP 2013*, pages 71–78. (Cited on page 41.)

[135] Goel, K., Rajani, N. F., Vig, J., Taschdjian, Z., Bansal, M., and Ré, C. (2021). Robustness Gym: Unifying the NLP Evaluation Landscape. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*, pages 42–55. (Cited on page 152.)

[136] Goldstein, J., Mittal, V. O., Carbonell, J. G., and Kantrowitz, M. (2000). Multi-Document Summarization by Sentence Extraction. In *NAACL-ANLP 2000 Workshop: Automatic Summarization*. (Cited on page 44.)

[137] Gong, H., Shen, Y., Yu, D., Chen, J., and Yu, D. (2020). Recurrent Chunking Mechanisms for Long-Text Machine Reading Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6751–6761. (Cited on page 38.)

[138] Gong, Z., Tong, S., Wu, H., Liu, Q., Tao, H., Huang, W., and Yu, R. (2022). Tipster: A Topic-Guided Language Model for Topic-Aware

Text Segmentation. In *International Conference on Database Systems for Advanced Applications*, pages 213–221. Springer. (Cited on pages 38 and 229.)

[139] Gottwald, P. (2023). *Analyse und Evaluation aktueller Textzusammenfassungstechniken für das Deutsche im Anwendungskontext von Domänenkorpora*. Bachelor's thesis, University of Bamberg. Supervisor: Jegan, R.; Reviewer: Henrich, A. (Cited on pages 16, 24, 28, 129, 137, 138, 144, 148, 149, 150, 157, 163, 164, 170, 171, 173, 174, 175, 178, 180, 184, 185, and 221.)

[140] Graves, A. (2013). Generating Sequences With Recurrent Neural Networks. *arXiv preprint arXiv:13.0850*. (Cited on page 15.)

[141] Grishman, R. (1997). Information Extraction: Techniques and Challenges. In *Information Extraction a Multidisciplinary Approach to an Emerging Information Technology: International Summer School, SCIE-97 Frascati, Italy, July 14–18, 1997*, pages 10–27. Springer. (Cited on pages 29 and 30.)

[142] Grishman, R. (2012). Information Extraction: Capabilities and Challenges. *International Winter School in Language and Speech Technologies WSLST*, 41. (Cited on page 30.)

[143] Grusky, M., Naaman, M., and Artzi, Y. (2018). Newsroom: A Dataset of 1.3 Million Summaries With Diverse Extractive Strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 708–719. (Cited on pages 118, 136, 152, 158, and 171.)

[144] Gulowaty, B. and Woźniak, M. (2021). Extracting Interpretable Decision Tree Ensemble from Random Forest. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE. (Cited on page 190.)

[145] Guo, H., Pasunuru, R., and Bansal, M. (2021). An Overview of Uncertainty Calibration for Text Classification and the Role of Distillation. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 289–306. (Cited on page 115.)

[146] Guo, M., Ainslie, J., Uthus, D. C., Ontanon, S., Ni, J., Sung, Y.-H., and Yang, Y. (2022). LongT5: Efficient Text-to-Text Transformer for Long Sequences. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 724–736. (Cited on pages 149, 164, and 171.)

[147] Gupta, P., Nigam, S., and Singh, R. (2024). Automatic Extractive Text Summarization Using Multiple Linguistic Features. *ACM Transac-*

tions on Asian and Low-Resource Language Information Processing. (Cited on pages 56, 57, 59, and 60.)

[148] HaCohen-Kerner, Y., Miller, D., and Yigal, Y. (2020). The Influence of Preprocessing on Text Classification Using a Bag-of-Words Representation. *PLOS One*, 15(5):e0232525. (Cited on page 35.)

[149] Hadi, M. U., Qureshi, R., Shah, A., Irfan, M., Zafar, A., Shaikh, M. B., Akhtar, N., Wu, J., Mirjalili, S., et al. (2023). A Survey on Large Language Models: Applications, Challenges, Limitations, and Practical Usage. *Authorea Preprints*. (Cited on pages 64 and 114.)

[150] Hambarde, K. A. and Proenca, H. (2023). Information Retrieval: Recent Advances and Beyond. *IEEE Access*, 11:76581–76604. (Cited on page 28.)

[151] Han, X., Zhao, W., Ding, N., Liu, Z., and Sun, M. (2022). PTR: Prompt Tuning With Rules for Text Classification. *AI Open*, 3:182–192. (Cited on page 36.)

[152] Harms, B. (2024). *Comparing Text Summarization Models and Training Datasets by Summarizing German School Website*. Bachelor's thesis, University of Bamberg. Supervisor: Jegan, R.; Reviewer: Henrich, A. (Cited on pages 16, 24, 28, 56, 58, 144, 148, 149, 150, 151, 157, 163, 164, 170, 171, 173, 175, 176, 178, 185, and 221.)

[153] Harris, Z. S. (1954). Distributional Structure. *WORD*, 10(2-3):146–162. (Cited on pages 6 and 21.)

[154] Hearst, M. A. (1992). Automatic Acquisition of Hyponyms From Large Text Corpora. In *COLING 1992 Volume 2: The 14th International Conference on Computational Linguistics*, pages 539–544. (Cited on page 31.)

[155] Hearst, M. A. (1997). TextTiling: Segmenting Text Into Multi-Paragraph Subtopic Passages. *Computational Linguistics*, 23(1):33–64. (Cited on pages 20, 37, 38, 39, 197, 204, 205, 206, 227, and 229.)

[156] Hebeis, M. (2025). *Entity Matching for Person Records in Authority Files - A Case Study-Based Comparison of Approaches*. Master's thesis, University of Bamberg. `https://doi.org/10.20378/irb-112660`. Supervisor: Fruth, L.; Reviewer: Henrich, A. (Cited on pages 12, 59, 157, 161, 162, 163, 165, 224, 225, 226, 227, and 229.)

[157] Held, W., Harris, C., Best, M., and Yang, D. (2023). A Material Lens on Coloniality in NLP. *arXiv preprint arXiv:2311.08391*. (Cited on page 101.)

[158] Henderson, P., Hu, J., Romoff, J., Brunskill, E., Jurafsky, D., and Pineau, J. (2020). Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning. *Journal of Machine Learning*

Research, 21(248):1–43. (Cited on page 62.)

[159] Henrich, A. (2008). *Information Retrieval 1 (Grundlagen, Modelle und Anwendungen)*. Bamberg: Lehrstuhl für Medieninformatik. (Cited on pages 27 and 97.)

[160] Henrich, A. and Wegmann, M. (2021). Search and Evaluation Methods for Class Level Information Retrieval: Extended Use and Evaluation of Methods Applied in Expertise Retrieval. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, pages 681–684. (Cited on page 10.)

[161] Hersh, W. (2024). Search Still Matters: Information Retrieval in the Era of Generative AI. *Journal of the American Medical Informatics Association*, 31(9):2159–2161. (Cited on page 27.)

[162] Hershcovich, D., Webersinke, N., Kraus, M., Bingler, J., and Leippold, M. (2022). Towards Climate Awareness in NLP Research. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2480–2494. (Cited on pages 62 and 183.)

[163] Ho, T. K. (1995). Random Decision Forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282. IEEE. (Cited on pages 10, 21, 63, and 184.)

[164] Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780. (Cited on pages 15, 21, and 170.)

[165] Hoda, R., Salleh, N., and Grundy, J. (2018). The Rise and Evolution of Agile Software Development. *IEEE Software*, 35(5):58–63. (Cited on page 71.)

[166] Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. (2020). The Curious Case of Neural Text Degeneration. In *8th International Conference on Learning Representations, ICLR*. (Cited on pages 105 and 155.)

[167] Hong, D., Wang, T., and Baek, S. (2023). Protorynet-Interpretable Text Classification via Prototype Trajectories. *Journal of Machine Learning Research*, 24(264):1–39. (Cited on pages 53, 56, and 57.)

[168] Honnibal, M., Montani, I., Van Landeghem, S., Boyd, A., et al. (2020). spaCy: Industrial-Strength Natural Language Processing in Python. (Cited on page 187.)

[169] Hovy, E. H., Lin, C.-Y., Zhou, L., and Fukumoto, J. (2006). Automated Summarization Evaluation With Basic Elements. In *LREC*, volume 6, pages 604–611. (Cited on page 47.)

[170] Hu, J., Ruder, S., Siddhant, A., Neubig, G., Firat, O., and Johnson, M. (2020). XTREME: A Massively Multilingual Multi-Task Benchmark for Evaluating Cross-Lingual Generalisation. In *International Confer-*

ence on Machine Learning, pages 4411–4421. PMLR. (Cited on page 101.)

[171] Hu, S., Ding, N., Wang, H., Liu, Z., Wang, J., Li, J., Wu, W., and Sun, M. (2022). Knowledgeable Prompt-Tuning: Incorporating Knowledge Into Prompt Verbalizer for Text Classification. In Muresan, S., Nakov, P., and Villavicencio, A., editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2225–2240, Dublin, Ireland. Association for Computational Linguistics. (Cited on page 36.)

[172] Hu, X., Chen, J., Meng, S., Wen, L., and Yu, P. S. (2023a). SelfLRE: Self-Refining Representation Learning for Low-Resource Relation Extraction. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2364–2368. (Cited on page 52.)

[173] Hu, X., Guo, Z., Teng, Z., King, I., and Yu, P. S. (2023b). Multimodal Relation Extraction With Cross-Modal Retrieval and Synthesis. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 303–311, Toronto, Canada. Association for Computational Linguistics. (Cited on page 32.)

[174] Huang, J. and Chang, K. C.-C. (2023). Towards Reasoning in Large Language Models: A Survey. In *61st Annual Meeting of the Association for Computational Linguistics, ACL 2023*, pages 1049–1065. Association for Computational Linguistics. (Cited on page 64.)

[175] Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., et al. (2025). A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *ACM Transactions on Information Systems*, 43(2):1–55. (Cited on pages 64, 93, 94, 114, 115, and 116.)

[176] Huang, Y., Feng, X., Feng, X., and Qin, B. (2021). The Factual Inconsistency Problem in Abstractive Text Summarization: A Survey. *arXiv preprint arXiv:2104.14839*. (Cited on page 93.)

[177] Hurst, A., Lerer, A., Goucher, A. P., Perelman, A., Ramesh, A., Clark, A., Ostrow, A., Welihinda, A., Hayes, A., Radford, A., et al. (2024). GPT-4o System Card. *arXiv preprint arXiv:2410.21276*. (Cited on pages 21, 122, 204, and 222.)

[178] Ioannidis, J. P. (2005). Why Most Published Research Findings Are False. *PLOS Medicine*, 2(8):e124. (Cited on page 104.)

[179] Islam, R. and Moushi, O. M. (2025). GPT-4o: The Cutting-Edge Advancement in Multimodal LLM. In *Intelligent Computing: Proceedings of the Computing Conference*, pages 47–60. Springer. (Cited on page 204.)

[180] Ispaylar, A. (2015). Selbstreflexion. In *Psychologie der Werte: Von Achtsamkeit bis Zivilcourage – Basiswissen aus Psychologie und Philosophie*, pages 177–186. Springer. (Cited on page 76.)

[181] Jackson, H. J. (2001). *Marginalia: Readers Writing in Books*. Yale University Press. (Cited on page 121.)

[182] Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Carney, A., et al. (2024). OpenAI o1 System Card. *arXiv preprint arXiv:2412.16720*. (Cited on page 115.)

[183] Jay, M., Ostapenco, V., Lefèvre, L., Trystram, D., Orgerie, A.-C., and Fichel, B. (2023). An Experimental Comparison of Software-Based Power Meters: Focus on CPU and GPU. In *2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 106–118. IEEE. (Cited on pages 109, 178, and 214.)

[184] Jegan, R. (2023). A Fundamental Shift in NLP? Traditional Models vs. Neural Networks and Large Language Models - Observations in Student Bachelor's and Master's Theses. In Cholotta, C. and Renker, C., editors, *META-LING 2023 - Methodological Exploration and Technological Advances in Linguistics*, page 11. (Cited on page 219.)

[185] Jegan, R. (2024). Observations in Students' Theses - A Critical Analysis of Use-Cases, Models and Problems in Natural Language Processing. In Karajgikar, J., Janco, A., and Otis, J., editors, *DH2024 Book of Abstracts*, pages 307–308. (Cited on pages 111, 112, 140, 145, 146, 224, and 225.)

[186] Jegan, R. (2026). LLMs vs. Traditional Models: A Task-Based Analysis of NLP Use Cases. In *KI in Bibliotheken weiterdenken – Datenqualität, Infrastruktur und Anwendungen für kleine und große Sprachmodelle*. Deutsche Nationalbibliothek. `https://wiki.dnb.de/spaces/FNMVE/pages/449892881`. (Cited on pages 219, 225, and 235.)

[187] Jegan, R. and Henrich, A. (2025a). A Structured Literature Review on Traditional Approaches in Current Natural Language Processing. `https://arxiv.org/abs/2505.12970`. (Cited on pages 12, 16, 28, 50, 51, 52, 53, 58, 59, 60, 65, 168, 192, 228, 229, 230, 231, and 233.)

[188] Jegan, R. and Henrich, A. (2025b). Contrasting Traditional Models and LLMs: An Evaluation Based on Text Segmentation. In *Proceedings of the 21st Conference on Natural Language Processing (KONVENS 2025): Workshops*, pages 274–281. (Cited on pages 6, 28, 39, 56, 57, 59, 139, 183, 197, 202, 206, 207, 211, 212, 214, 227, and 234.)

[189] Jegham, N., Abdelatti, M., Elmoubarki, L., and Hendawi, A. (2025). How Hungry Is AI? Benchmarking Energy, Water, and Carbon Footprint of LLM Inference. *arXiv preprint arXiv:2505.09598*. (Cited on

page 215.)

[190] Jelinek, F., Bahl, L., and Mercer, R. (1975). Design of a Linguistic Statistical Decoder for the Recognition of Continuous Speech. *IEEE Transactions on Information Theory*, 21(3):250–256. (Cited on page 26.)

[191] Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., and Fung, P. (2023). Survey of Hallucination in Natural Language Generation. *ACM Computing Surveys*, 55(12):1–38. (Cited on pages 49, 93, 94, and 118.)

[192] Jia, J., Liang, W., and Liang, Y. (2023). A Review of Hybrid and Ensemble in Deep Learning for Natural Language Processing. *arXiv preprint arXiv:2312.05589*. (Cited on page 10.)

[193] Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., Casas, D. d. l., Hanna, E. B., Bressand, F., et al. (2024). Mixtral of Experts. *arXiv preprint arXiv:2401.04088*. (Cited on pages 21, 150, 164, and 171.)

[194] Jiang, J. (2012). Information Extraction from Text. *Mining Text Data*, pages 11–41. (Cited on page 223.)

[195] Jiang, W., Synovic, N., Hyatt, M., Schorlemmer, T. R., Sethi, R., Lu, Y.-H., Thiruvathukal, G. K., and Davis, J. C. (2023). An Empirical Study of Pre-Trained Model Reuse in the Hugging Face Deep Learning Model Registry. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 2463–2475. IEEE. (Cited on page 65.)

[196] Jin, Z., Chauhan, G., Tse, B., Sachan, M., and Mihalcea, R. (2021). How Good Is NLP? A Sober Look at NLP Tasks through the Lens of Social Impact. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3099–3113. (Cited on page 183.)

[197] Joachims, T. (1998). Text Categorization With Support Vector Machines: Learning With Many Relevant Features. In *European Conference on Machine Learning*, pages 137–142. Springer. (Cited on page 12.)

[198] John, A. and Wilscy, M. (2013). Random Forest Classifier Based Multi-Document Summarization System. In *2013 IEEE Recent Advances in Intelligent Computational Systems (RAICS)*, pages 31–36. IEEE. (Cited on pages 44, 56, 57, and 59.)

[199] Jones, K. S. (1972). A Statistical Interpretation of Term Specificity and Its Application in Retrieval. *Journal of Documentation*, 28(1):11–21. (Cited on pages 7, 21, 44, and 122.)

[200] Jones, K. S. (1999). Automatic Summarizing: Factors and Directions. *Advances in Automatic Text Summarization*. (Cited on pages 48, 133, 134, 135, 138, 139, and 221.)

[201] Jones, K. S. (2007). Automatic Summarising: The State of the Art. *Information Processing & Management*, 43(6):1449–1481. (Cited on pages 45 and 47.)

[202] Joshi, M., Chen, D., Liu, Y., Weld, D. S., Zettlemoyer, L., and Levy, O. (2020). SpanBERT: Improving Pre-Training by Representing and Predicting Spans. *Transactions of the Association for Computational Linguistics*, 8:64–77. (Cited on page 33.)

[203] Jurafsky, D. and Martin, J. H. (2024). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 3rd edition. Online manuscript released January 12, 2025. (Cited on pages 6, 7, 11, 12, 13, 14, 15, 16, 18, 25, 26, 32, 36, 48, 49, 96, 97, 99, 102, 115, 170, 183, 191, and 227.)

[204] Kalchbrenner, N. and Blunsom, P. (2013). Recurrent Continuous Translation Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709. (Cited on page 15.)

[205] Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling Laws for Neural Language Models. *arXiv preprint arXiv:2001.08361*. (Cited on page 169.)

[206] Karlsson, F., Voutilainen, A., Heikkilae, J., and Anttila, A. (1995). *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text*, volume 4. Walter de Gruyter. (Cited on pages 8 and 20.)

[207] Kathikar, A., Nair, A., Lazarine, B., Sachdeva, A., and Samtani, S. (2023). Assessing the Vulnerabilities of the Open-Source Artificial Intelligence (AI) Landscape: A Large-Scale Analysis of the Hugging Face Platform. In *2023 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 1–6. IEEE. (Cited on page 65.)

[208] Kazemitabar, J., Amini, A., Bloniarz, A., and Talwalkar, A. S. (2017). Variable Importance Using Decision Trees. *Advances in Neural Information Processing Systems*, 30. (Cited on page 224.)

[209] Kedzie, C., Mckeown, K., and Daumé III, H. (2018). Content Selection in Deep Learning Models of Summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1818–1828. (Cited on page 45.)

[210] Keerthan Kumar, T., Adhith, A., Jayadeep, N., and Koolagudi, S. G. (2024). Syntactic Simplification of English Sentences Using Parse Trees. In *Proceedings of the 2024 Sixteenth International Conference on Contemporary Computing*, pages 490–496. (Cited on pages 56, 57, 59, 60, and 230.)

[211] Keraghel, I., Morbieu, S., and Nadif, M. (2024). Beyond Words: A Comparative Analysis of LLM Embeddings for Effective Clustering. In *International Symposium on Intelligent Data Analysis*, pages 205–216. Springer. (Cited on page 27.)

[212] Kerth, N. L. (2013). *Project Retrospectives: A Handbook for Team Reviews*. Addison-Wesley. (Cited on page 76.)

[213] Kew, T., Kostrzewa, M., and Ebling, S. (2023). 20 Minuten: A Multi-Task News Summarisation Dataset for German. In *Proceedings of the Swiss Text Analytics Conference*, pages 1–13. Association for Computational Linguistics. (Cited on page 152.)

[214] Kim, Y., Petrov, P., Petrushkov, P., Khadivi, S., and Ney, H. (2019). Pivot-based Transfer Learning for Neural Machine Translation between Non-English Languages. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 866–876. (Cited on page 191.)

[215] Kincaid, J. P., Fishburne, R., Rogers, R., and Chissom, B. (1975). Derivation of New Readability Formulas (Automated Reliability Index, Fog Count and Flesch Reading Ease Formula) for Navy Enlisted Personnel. *Naval Technical Training, US Naval Air Station: Millington, TN*. (Cited on page 40.)

[216] Klimt, B. and Yang, Y. (2004). The Enron Corpus: A New Dataset for Email Classification Research. In *European Conference on Machine Learning*, pages 217–226. Springer. (Cited on page 171.)

[217] Kocmi, T., Avramidis, E., Bawden, R., Bojar, O., Dvorkovich, A., Federmann, C., Fishel, M., Freitag, M., Gowda, T., Grundkiewicz, R., et al. (2024). Findings of the WMT24 General Machine Translation Shared Task: The LLM Era is Here but MT is Not Solved Yet. In *Proceedings of the Ninth Conference on Machine Translation*, pages 1–46. (Cited on page 26.)

[218] Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al. (2007). Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the ACL 2007 Demo and Poster Sessions*, pages 177–180. Association for Computational Linguistics. (Cited on page 42.)

[219] Koehn, P. and Knowles, R. (2017). Six Challenges for Neural Machine Translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39. (Cited on page 93.)

[220] Koh, H. Y., Ju, J., Liu, M., and Pan, S. (2022). An Empirical Survey on Long Document Summarization: Datasets, Models, and Metrics. *ACM*

Computing Surveys, 55(8):1–35. (Cited on page 38.)

[221] Koller, D. and Sahami, M. (1997). Hierarchically Classifying Documents Using Very Few Words. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 170–178. (Cited on page 20.)

[222] Krouska, A., Troussas, C., and Virvou, M. (2016). The Effect of Preprocessing Techniques on Twitter Sentiment Analysis. In *2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA)*, pages 1–5. IEEE. (Cited on page 87.)

[223] Kudo, T. and Matsumoto, Y. (2000). Use of Support Vector Learning for Chunk Identification. In *Fourth Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop*. (Cited on page 12.)

[224] Laban, P., Schnabel, T., Bennett, P., and Hearst, M. A. (2021). Keep It Simple: Unsupervised Simplification of Multi-Paragraph Text. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6365–6378. (Cited on pages 42, 112, 113, 114, 132, 133, 147, 148, 160, 193, and 230.)

[225] Ladhak, F., Durmus, E., Cardie, C., and Mckeown, K. (2020). WikiLingua: A New Benchmark Dataset for Cross-Lingual Abstractive Summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4034–4048. (Cited on pages 149 and 170.)

[226] Lam, K. N., Nguyen, T. N., and Kalita, J. (2022). Vietnamese Text Summarization Based on Elementary Discourse Units. In *Proceedings of the 2022 6th International Conference on Natural Language Processing and Information Retrieval*, pages 1–5. (Cited on pages 54, 56, 57, and 60.)

[227] Lamprier, S., Amghar, T., Levrat, B., and Saubion, F. (2007a). ClassStruggle: A Clustering Based Text Segmentation. In *Proceedings of the 2007 ACM Symposium on Applied Computing*, pages 600–604. (Cited on page 204.)

[228] Lamprier, S., Amghar, T., Levrat, B., and Saubion, F. (2007b). On Evaluation Methodologies for Text Segmentation Algorithms. In *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)*, volume 2, pages 19–26. IEEE. (Cited on page 211.)

[229] Lang, A. (2025). *Automatische Erzeugung von Marginalien*. Master's thesis, University of Bamberg. `https://doi.org/10.20378/irb-112598`. Supervisor: Jegan, R.; Reviewer: Henrich, A. (Cited on pages 59, 121, 122, 124, 125, 126, 140, 143, 220, 222, and 227.)

[230] Lang, A., Jegan, R., and Henrich, A. (2025). Automatic Creation of Marginalia. In *Proceedings of the 21st Conference on Natural Language Processing (KONVENS 2025): Long and Short Papers*, pages 228–240. (Cited on pages 121, 123, 124, 125, 127, 128, 222, and 227.)

[231] Lang, K. (1995). NewsWeeder: Learning to Filter Netnews. In *Machine Learning Proceedings 1995*, pages 331–339. Elsevier. (Cited on page 34.)

[232] Lapata, M., Barzilay, R., et al. (2005). Automatic Evaluation of Text Coherence: Models and Representations. In *IJCAI*, volume 5, pages 1085–1090. (Cited on page 99.)

[233] Lapuschkin, S., Binder, A., Montavon, G., Muller, K.-R., and Samek, W. (2016). Analyzing Classifiers: Fisher Vectors and Deep Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2912–2920. (Cited on page 107.)

[234] Latif, Y. A. (2025). Hallucinations in Large Language Models and Their Influence on Legal Reasoning: Examining the Risks of Ai-Generated Factual Inaccuracies in Judicial Processes. *Journal of Computational Intelligence, Machine Reasoning, and Decision-Making*, 10(2):10–20. (Cited on page 231.)

[235] Lau, J. H., Clark, A., and Lappin, S. (2017). Grammaticality, Acceptability, and Probability: A Probabilistic View of Linguistic Knowledge. *Cognitive Science*, 41(5):1202–1241. (Cited on page 132.)

[236] Lehmann, A. and Landes, D. (2024). Extracting Metadata From Learning Videos for Ontology-Based Recommender Systems Using Whisper & GPT. In *2024 IEEE Global Engineering Education Conference (EDUCON)*, pages 1–8. IEEE. (Cited on pages 56, 60, 197, 198, 199, 201, 203, 205, and 218.)

[237] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474. (Cited on page 18.)

[238] Li, B., Fang, G., Yang, Y., Wang, Q., Ye, W., Zhao, W., and Zhang, S. (2023). Evaluating ChatGPT's Information Extraction Capabilities: An Assessment of Performance, Explainability, Calibration, and Faithfulness. *CoRR*, abs/2304.11633. (Cited on page 31.)

[239] Li, J., Tang, T., Zhao, W. X., Nie, J.-Y., and Wen, J.-R. (2024). Pre-Trained Language Models for Text Generation: A Survey. *ACM Computing Surveys*, 56(9):1–39. (Cited on page 129.)

[240] Li, P., Yang, J., Islam, M. A., and Ren, S. (2025). Making AI Less "Thirsty": Uncovering and Addressing the Secret Water Footprint of AI Models. *Communications of the ACM*, 68(7):54–61. (Cited on page 109.)

[241] Li, Q., Peng, H., Li, J., Xia, C., Yang, R., Sun, L., Yu, P. S., and He, L. (2022). A Survey on Text Classification: From Traditional to Deep Learning. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(2):1–41. (Cited on pages 36, 59, and 60.)

[242] Li, X.-L. and Liu, B. (2014). Rule-Based Classification. In *Data Classification*, pages 149–184. Chapman and Hall/CRC. (Cited on pages 35, 56, and 59.)

[243] Li, Y., Li, J., Suhara, Y., Doan, A., and Tan, W.-C. (2020). Deep Entity Matching With Pre-Trained Language Models. *Proceedings of the VLDB Endowment*, 14(1):50–60. (Cited on page 161.)

[244] Liao, B., Herold, C., Khadivi, S., and Monz, C. (2024). IKUN for WMT24 General MT Task: LLMs Are Here for Multilingual Machine Translation. In *Proceedings of the Ninth Conference on Machine Translation*, pages 263–269. (Cited on page 26.)

[245] Liberatore, D., Kalimeri, K., Sever, D., and Mejova, Y. (2024). Quantitative Information Extraction From Humanitarian Documents. In *Proceedings of the 2024 International Conference on Information Technology for Social Good*, pages 240–248. (Cited on page 59.)

[246] Licari, D., Bushipaka, P., Marino, G., Comandé, G., and Cucinotta, T. (2023). Legal Holding Extraction From Italian Case Documents Using Italian-Legal-Bert Text Summarization. In *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law*, pages 148–156. (Cited on pages 51, 54, 192, 193, 231, and 232.)

[247] Lin, C.-Y. (2004a). Looking for a Few Good Metrics: Automatic Summarization Evaluation-How Many Samples Are Enough? In *NTCIR Conference on Evaluation of Information Access Technologies*. (Cited on page 47.)

[248] Lin, C.-Y. (2004b). ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pages 74–81. (Cited on pages 46, 47, and 91.)

[249] Lin, K. (2023). *Decision Trees for Text Classification in CS2*. ACM, New York, NY, USA. (Cited on pages 53 and 229.)

[250] Lin, Y.-H., Chen, C.-Y., Lee, J., Li, Z., Zhang, Y., Xia, M., Rijhwani, S., He, J., Zhang, Z., Ma, X., et al. (2019). Choosing Transfer Languages for Cross-Lingual Learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, volume 57. (Cited on page 101.)

[251] Lin, Z., Trivedi, S., and Sun, J. (2024). Generating With Confidence: Uncertainty Quantification for Black-Box Large Language Models. *Transactions on Machine Learning Research*. (Cited on page 104.)

[252] Linardatos, P., Papastefanopoulos, V., and Kotsiantis, S. (2020). Explainable AI: A Review of Machine Learning Interpretability Methods. *Entropy*, 23(1):18. (Cited on page 64.)

[253] Liu, L. (2023). Information Extraction Method of Topic Webpage Based on Multi-Angle Feature Learning. In *Proceedings of the 2023 4th International Conference on Computing, Networks and Internet of Things*, pages 92–96. (Cited on pages 51, 56, 57, 58, 60, and 228.)

[254] Liu, P., Qiu, X., and Huang, X. (2016). Recurrent Neural Network for Text Classification with Multi-Task Learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2873–2879. (Cited on pages 36, 56, and 57.)

[255] Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. (2023). Pre-Train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Computing Surveys*, 55(9):1–35. (Cited on page 18.)

[256] Liu, Y. (2019). RoBERTa: A Robustly Optimized Bert Pretraining Approach. *arXiv preprint arXiv:1907.11692*, 364. (Cited on pages 16, 36, 62, and 161.)

[257] Lopez, A. (2008). Statistical Machine Translation. *ACM Computing Surveys (CSUR)*, 40(3):1–49. (Cited on page 25.)

[258] Lu, S., Chen, L., Wang, W., Xu, C., Zhao, W., Guan, Z., and Lu, G. (2022). A Simple Semi-Supervised Joint Learning Framework for Few-Shot Text Classification. In *Proceedings of the 2022 5th International Conference on Artificial Intelligence and Pattern Recognition*, pages 14–21. (Cited on page 53.)

[259] Luc, D. T. (2008). Pareto Optimality. *Pareto Optimality, Game Theory and Equilibria*, pages 481–515. (Cited on page 108.)

[260] Luhn, H. P. (1957). A Statistical Approach to Mechanized Encoding and Searching of Literary Information. *IBM Journal of Research and Development*, 1(4):309–317. (Cited on pages 7, 21, 24, 44, and 122.)

[261] Lukasik, M., Dadachev, B., Papineni, K., and Simões, G. (2020). Text Segmentation by Cross Segment Attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4707–4716. (Cited on pages 37, 38, 39, 56, 57, 59, and 60.)

[262] Luo, C., Chen, Z., Jiang, X., and Yang, S. (2022). Gap Sentences Generation With TextRank for Chinese Text Summarization. In *Proceedings*

of the 2022 5th International Conference on Algorithms, Computing and Artificial Intelligence, pages 1–5. (Cited on page 55.)

[263] Lv, H., Liu, J., Wang, H., Wang, Y., Luo, J., and Liu, Y. (2023). Efficient Hybrid Generation Framework for Aspect-Based Sentiment Analysis. In Vlachos, A. and Augenstein, I., editors, *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1007–1018, Dubrovnik, Croatia. Association for Computational Linguistics. (Cited on page 30.)

[264] MacQueen, J. (1967). Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, volume 5, pages 281–298. University of California press. (Cited on page 27.)

[265] Maddela, M. and Xu, W. (2018). A Word-Complexity Lexicon and a Neural Readability Ranking Model for Lexical Simplification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. (Cited on page 40.)

[266] Madsen, A., Reddy, S., and Chandar, S. (2022). Post-hoc Interpretability for Neural NLP: A Survey. *ACM Computing Surveys*, 55(8):1–42. (Cited on pages 105, 106, and 157.)

[267] Maleki, N., Padmanabhan, B., and Dutta, K. (2024). AI Hallucinations: A Misnomer Worth Clarifying. In *2024 IEEE Conference on Artificial Intelligence (CAI)*, pages 133–138. IEEE. (Cited on pages 91, 93, and 94.)

[268] Mann, W. C. and Thompson, S. A. (1988). Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. *Text-interdisciplinary Journal for the Study of Discourse*, 8(3):243–281. (Cited on page 37.)

[269] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press. (Cited on page 37.)

[270] Manning, C. D. and Schutze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT press. (Cited on page 92.)

[271] Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., and McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60. (Cited on pages 143 and 187.)

[272] Maron, M. E. and Kuhns, J. L. (1960). On Relevance, Probabilistic Indexing and Information Retrieval. *Journal of the ACM (JACM)*,

7(3):216–244. (Cited on pages 11 and 21.)

[273] Maronikolakis, A. and Schütze, H. (2021). Multidomain Pretrained Language Models for Green NLP. In *Proceedings of the Second Workshop on Domain Adaptation for NLP*, pages 1–8. (Cited on page 183.)

[274] Martin, L., De La Clergerie, É. V., Sagot, B., and Bordes, A. (2020). Controllable Sentence Simplification. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4689–4698. (Cited on page 42.)

[275] Maslej-Krešňáková, V., Sarnovsky, M., Butka, P., and Machová, K. (2020). Comparison of Deep Learning Models and Various Text Pre-Processing Techniques for the Toxic Comments Classification. *Applied Sciences*, 10(23):8631. (Cited on page 87.)

[276] Matschat, D. (2022). *Automatic Generation of Misinformation - Natural Language Generators in the Application Scenario of Wikipedia*. Master's thesis, University of Bamberg. `https://doi.org/10.20378/irb-113231`. Supervisor: Jegan, R.; Reviewer: Henrich, A. (Cited on pages 129, 130, 154, 155, 157, 159, 163, 164, 170, 173, 178, and 184.)

[277] Maynez, J., Narayan, S., Bohnet, B., and McDonald, R. (2020). On Faithfulness and Factuality in Abstractive Summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919. (Cited on page 93.)

[278] Mazumder, D., Paik, J. H., and Basu, A. (2023). A Graph Neural Network Model for Concept Prerequisite Relation Extraction. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 1787–1796. (Cited on pages 50, 52, 56, 57, 58, and 228.)

[279] McCarthy, J. F. and Lehnert, W. G. (1995). Using Decision Trees for Coreference Resolution. *arXiv preprint cmp-lg/9505043*. (Cited on page 8.)

[280] Mehmood, F., Shahzadi, R., Ghafoor, H., Asim, M. N., Ghani, M. U., Mahmood, W., and Dengel, A. (2023). EnML: Multi-Label Ensemble Learning for Urdu Text Classification. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 22(9):1–31. (Cited on pages 53, 56, and 58.)

[281] Meister, C., Pimentel, T., Wiher, G., and Cotterell, R. (2023). Locally Typical Sampling. *Transactions of the Association for Computational Linguistics*, 11:102–121. (Cited on page 155.)

[282] Mihalcea, R. and Tarau, P. (2004). TextRank: Bringing Order Into Text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411. (Cited on pages 26, 45, 56, 57,

59, 70, 122, 148, 164, 170, 190, and 220.)

[283] Mikolov, T. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*, 3781. (Cited on pages 6, 14, and 92.)

[284] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed Representations of Words and Phrases and Their Compositionality. *Advances in Neural Information Processing Systems*, 26. (Cited on pages 13, 21, and 166.)

[285] Miller, G. A. (1995). WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41. (Cited on page 41.)

[286] Miller, G. A. and Chomsky, N. (1963). Finitary Models of Language Users. *Handbook of Mathematical Psychology*, 2. (Cited on page 25.)

[287] Miller, T. (2019). Explanation in Artificial Intelligence: Insights From the Social Sciences. *Artificial Intelligence*, 267:1–38. (Cited on page 105.)

[288] Mirończuk, M. M. and Protasiewicz, J. (2018). A Recent Overview of the State-of-the-Art Elements of Text Classification. *Expert Systems with Applications*, 106:36–54. (Cited on page 34.)

[289] Misra, H., Yvon, F., Cappé, O., and Jose, J. (2011). Text Segmentation: A Topic Modeling Perspective. *Information Processing & Management*, 47(4):528–544. (Cited on page 204.)

[290] Möglich, F. (2025). *Ansätze für einen frageorientierten Zugang zu Studienordnungen*. Master's thesis, University of Bamberg. `https://doi.org/10.20378/irb-112943`. Supervisor: Jegan, R.; Reviewer: Henrich, A. (Cited on pages 59, 114, 119, 120, 157, 160, 161, 163, and 165.)

[291] Mohandes, M., Deriche, M., and Aliyu, S. O. (2018). Classifiers Combination Techniques: A Comprehensive Review. *IEEE Access*, 6:19626–19639. (Cited on page 191.)

[292] Moriarty, J. (2011). *Qualitative Methods Overview*. National Institute for Health Research School for Social Care. (Cited on page 95.)

[293] Mudgal, S., Li, H., Rekatsinas, T., Doan, A., Park, Y., Krishnan, G., Deep, R., Arcaute, E., and Raghavendra, V. (2018). Deep Learning for Entity Matching: A Design Space Exploration. In *Proceedings of the 2018 International Conference on Management of Data*, pages 19–34. (Cited on page 165.)

[294] Muennighoff, N., Tazi, N., Magne, L., and Reimers, N. (2023). MTEB: Massive Text Embedding Benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037. (Cited on page 66.)

[295] Murphy, K. P. (2022). *Probabilistic Machine Learning: An Introduction*. MIT press. (Cited on page 161.)

[296] Nahar, M., Lee, E.-J., Park, J. W., and Lee, D. (2025). Catch Me if You Search: When Contextual Web Search Results Affect the Detection of Hallucinations. *arXiv preprint arXiv:2504.01153*. (Cited on page 117.)

[297] Nahm, U. Y. and Mooney, R. J. (2002). Text Mining With Information Extraction. In *Proceedings of the AAAI 2002 Spring Symposium on Mining Answers From Texts and Knowledge Bases*, pages 60–67. Stanford CA. (Cited on page 8.)

[298] Nallapati, R., Zhai, F., and Zhou, B. (2017). SummaRuNNer: A Recurrent Neural Network Based Sequence Model for Extractive Summarization of Documents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31. (Cited on page 220.)

[299] Narayan, S., Cohen, S., and Lapata, M. (2018). Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization. In *2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807. Association for Computational Linguistics. (Cited on pages 49 and 150.)

[300] Narayan, S. and Gardent, C. (2014). Hybrid Simplification Using Deep Semantics and Machine Translation. In *The 52nd Annual Meeting of the Association for Computational Linguistics*, pages 435–445. (Cited on page 42.)

[301] Narayanan, D., Shoeybi, M., Casper, J., LeGresley, P., Patwary, M., Korthikanti, V., Vainbrand, D., Kashinkunti, P., Bernauer, J., Catanzaro, B., et al. (2021). Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15. (Cited on page 17.)

[302] Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N., and Mian, A. (2025). A Comprehensive Overview of Large Language Models. *ACM Transactions on Intelligent Systems and Technology*, 16(5):1–72. (Cited on page 169.)

[303] Neto, M. P. and Paulovich, F. V. (2020). Explainable Matrix-Visualization for Global and Local Interpretability of Random Forest Classification Ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1427–1437. (Cited on page 107.)

[304] Ng, A. and Jordan, M. (2001). On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes. *Advances in Neural Information Processing Systems*, 14. (Cited on pages 11, 56, 57, and 59.)

[305] Nguyen, H.-L., Woon, Y.-K., and Ng, W.-K. (2015). A Survey on Data Stream Clustering and Classification. *Knowledge and Information Systems*, 45:535–569. (Cited on page 27.)

[306] Nguyen, T. H., Nguyen, H. H., Ahmadi, Z., Hoang, T.-A., and Doan, T.-N. (2021). On the Impact of Dataset Size: A Twitter Classification Case Study. In *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pages 210–217. (Cited on page 165.)

[307] North, K., Zampieri, M., and Shardlow, M. (2023). Lexical Complexity Prediction: An Overview. *ACM Computing Surveys*, 55(9):1–42. (Cited on pages 54, 56, and 230.)

[308] Núñez, H., Angulo, C., and Català, A. (2002). Rule Extraction From Support Vector Machines. In *ESANN*, pages 107–112. (Cited on pages 22 and 107.)

[309] Omar, K. and Al-Shaar, M. (2023). Method for Arabic Text Summarization Using Statistical Features and Word2vector Approach. In *Proceedings of the 2023 9th International Conference on Computer Technology Applications*, pages 258–262. (Cited on page 55.)

[310] Opitz, J. (2024). A Closer Look at Classification Evaluation Metrics and a Critical Reflection of Common Evaluation Practice. *Transactions of the Association for Computational Linguistics*, 12:820–836. (Cited on page 133.)

[311] Orphanos, G., Kalles, D., Papagelis, A., and Christodoulakis, D. (1999). Decision Trees and NLP: A Case Study in POS Tagging. In *Proceedings of Annual Conference on Artificial Intelligence (ACAI)*. (Cited on page 223.)

[312] Osman, T., Khalil, H., Miltan, M., Shaalan, K., and Alfrjani, R. (2023). Exploiting Functional Discourse Grammar to Enhance Complex Arabic Relation Extraction Using a Hybrid Semantic Knowledge Base-Machine Learning Approach. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 22(8):1–30. (Cited on pages 52 and 228.)

[313] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. (2022). Training Language Models to Follow Instructions with Human Feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744. (Cited on page 21.)

[314] Oyewole, G. J. and Thopil, G. A. (2023). Data Clustering: Application and Trends. *Artificial Intelligence Review*, 56(7):6439–6475. (Cited on page 28.)

[315] Paetsch, F., Eberlein, A., and Maurer, F. (2003). Requirements Engineering and Agile Software Development. In *WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.*, pages 308–313. IEEE. (Cited on pages 74 and 76.)

[316] Paetzold, G. and Specia, L. (2013). Text Simplification as Tree Transduction. In *Proceedings of the 9th Brazilian Symposium in Information and Human Language Technology*. (Cited on pages 42, 56, and 59.)

[317] Paetzold, G. and Specia, L. (2016). SemEval 2016 Task 11: Complex Word Identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 560–569. (Cited on pages 41, 56, and 57.)

[318] Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Infolab. (Cited on pages 26 and 45.)

[319] Pahl, G., Beitz, W., Feldhusen, J., and Grote, K.-H. (2007). Engineering Design: A Systematic Approach. *Engineering Design: A Systematic Approach*. (Cited on pages 67, 72, 77, 78, 79, and 201.)

[320] Pak, I. and Teh, P. L. (2017). Text Segmentation Techniques: A Critical Review. *Innovative Computing, Optimization and Its Applications: Modeling and Simulations*, pages 167–181. (Cited on pages 37 and 38.)

[321] Paliouras, G., Karkaletsis, V., Petasis, G., and Spyropoulos, C. D. (2000). Learning Decision Trees for Named-Entity Recognition and Classification. In *ECAI Workshop on Machine Learning for Information Extraction*. (Cited on page 9.)

[322] Pang, R. (2024). Research on Text Classification Applications Based on NLP Technology. In *Proceedings of the 2024 5th International Conference on Computing, Networks and Internet of Things*, pages 108–111. (Cited on page 60.)

[323] Pang, R. Y. and Gimpel, K. (2019). Unsupervised Evaluation Metrics and Learning Criteria for Non-Parallel Textual Transfer. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 138–147. (Cited on pages 99 and 100.)

[324] Papagiannopoulou, A. and Angeli, C. (2022). Social Media Text Summarisation Techniques and Approaches: A Literature Review. In *Proceedings of the 2022 5th International Conference on Algorithms, Computing and Artificial Intelligence*, pages 1–6. (Cited on page 55.)

[325] Papagiannopoulou, E. and Tsoumakas, G. (2020). A Review of Keyphrase Extraction. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(2):e1339. (Cited on pages 123 and 128.)

[326] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318. (Cited on page 40.)

[327] Parfenova, A. (2024). Automating the Information Extraction From Semi-Structured Interview Transcripts. In *Companion Proceedings of the ACM Web Conference 2024*, pages 983–986. (Cited on page 59.)

[328] Park, M. S. and Park, E. (2023). ACL TA-DA: A Dataset for Text Summarization and Generation. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*, pages 1233–1239. (Cited on page 54.)

[329] Patil, R. and Gudivada, V. (2024). A Review of Current Trends, Techniques, and Challenges in Large Language Models (LLMs). *Applied Sciences*, 14(5):2074. (Cited on pages 48 and 106.)

[330] Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L.-M., Rothchild, D., So, D., Texier, M., and Dean, J. (2021). Carbon Emissions and Large Neural Network Training. *arXiv preprint arXiv:2104.10350*. (Cited on pages 61 and 109.)

[331] Pava, J., Meinhardt, C., Zaman, H. B. U., Friedman, T., Truong, S. T., Zhang, D., Marivate, V., and Koyejo, S. (2025). Mind the (Language) Gap: Mapping the Challenges of LLM Development in Low-Resource Language Contexts. (Cited on pages 102 and 103.)

[332] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830. (Cited on page 188.)

[333] Peeperkorn, M., Kouwenhoven, T., Brown, D., and Jordanous, A. (2024). Is Temperature the Creativity Parameter of Large Language Models? *CoRR*. (Cited on pages 104 and 105.)

[334] Peer, E., Rothschild, D., Gordon, A., Evernden, Z., and Damer, E. (2022). Data Quality of Platforms and Panels for Online Behavioral Research. *Behavior Research Methods*, 54(4):1643–1662. (Cited on page 97.)

[335] Peng, B., Li, C., He, P., Galley, M., and Gao, J. (2023). Instruction Tuning With GPT-4. *arXiv preprint arXiv:2304.03277*. (Cited on page 156.)

[336] Pennington, J., Socher, R., and Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. (Cited on page 15.)

[337] Perez-Beltrachini, L. and Lapata, M. (2021). Models and Datasets for Cross-Lingual Summarisation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9408–9423. (Cited on pages 149 and 170.)

[338] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In Walker, M., Ji, H., and Stent, A., editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics. (Cited on pages 122 and 143.)

[339] Petkovic, D., Altman, R., Wong, M., and Vigil, A. (2018). Improving the Explainability of Random Forest Classifier – User Centered Approach. In *Pacific Symposium on Biocomputing*, volume 23, page 204. (Cited on pages 107 and 190.)

[340] Pevzner, L. and Hearst, M. A. (2002). A Critique and Improvement of an Evaluation Metric for Text Segmentation. *Computational Linguistics*, 28(1):19–36. (Cited on page 211.)

[341] Ponte, J. M. and Croft, W. B. (1997). Text Segmentation by Topic. In *International Conference on Theory and Practice of Digital Libraries*, pages 113–125. Springer. (Cited on pages 37 and 39.)

[342] Ponti, E., Reichart, R., Korhonen, A., and Vulic, I. (2018). Isomorphic Transfer of Syntactic Structures in Cross-Lingual NLP. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics. (Cited on page 101.)

[343] Porter, M. F. (1980). An Algorithm for Suffix Stripping. *Program*, 14(3):130–137. (Cited on page 141.)

[344] Porter, M. F. (2001). Snowball: A Language for Stemming Algorithms. (Cited on pages 6 and 141.)

[345] Prusa, J., Khoshgoftaar, T. M., and Seliya, N. (2015). The Effect of Dataset Size on Training Tweet Sentiment Classifiers. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 96–102. IEEE. (Cited on page 165.)

[346] Pulver, N. (2023). *Informationsgewinnung aus Unternehmensnamen mithilfe aktueller Text Mining Verfahren*. Master's thesis, University of Bamberg. Supervisor: Jegan, R.; Reviewer: Henrich, A. (Cited on pages 6, 8, 12, 16, 24, 28, 56, 57, 59, 140, 141, 142, 143, 163, 166, 167, 190, 192, 193, 194, 219, 224, 225, 226, and 228.)

[347] Purdy, C., Wang, X., He, L., and Riedl, M. (2018). Predicting Generated Story Quality with Quantitative Measures. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 14, pages 95–101. (Cited on page 99.)

[348] Qin, L., Chen, Q., Zhou, Y., Chen, Z., Li, Y., Liao, L., Li, M., Che, W., and Yu, P. S. (2025). A Survey of Multilingual Large Language Models. *Patterns*, 6(1). (Cited on page 102.)

[349] Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning*, 1:81–106. (Cited on pages 9, 21, and 223.)

[350] Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Elsevier. (Cited on page 9.)

[351] Quinlan, J. R. et al. (1996). Bagging, Boosting, and C4.5. In *AAAI/IAAI, Vol. 1*, pages 725–730. Citeseer. (Cited on page 10.)

[352] Raab, S. (2023). *Few Shot Text Classficiation for Domain Specific Languages*. Master's thesis, University of Bamberg. Supervisor: Jegan, R.; Reviewer: Henrich, A. (Cited on pages 6, 12, 24, 28, 88, 144, 145, 146, 188, 219, 225, 226, and 229.)

[353] Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., and Sutskever, I. (2023). Robust Speech Recognition via Large-Scale Weak Supervision. In *International Conference on Machine Learning*, pages 28492–28518. PMLR. (Cited on pages 197 and 202.)

[354] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving Language Understanding by Generative Pre-Training. Preprint on webpage at `https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf`. (Cited on page 62.)

[355] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language Models are Unsupervised Multitask Learners. *OpenAI Blog*, 1(8):9. (Cited on page 170.)

[356] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the Limits of Transfer Learning With a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):1–67. (Cited on pages 21, 30, 100, 149, 164, and 171.)

[357] Rahm, S. and Lunkenbein, M. (2014). Anbahnung von Reflexivität im Praktikum. Empirische Befunde zur Wirkung von Beobachtungsaufgaben im Grundschulpraktikum. *Schulpraktika in der Lehrerbildung: Theoretische Grundlagen, Konzeptionen, Prozesse und Effekte*. (Cited on page 76.)

[358] Ram, O., Levine, Y., Dalmedigos, I., Muhlgay, D., Shashua, A., Leyton-Brown, K., and Shoham, Y. (2023). In-Context Retrieval-Augmented Language Models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331. (Cited on page 18.)

[359] Ramshaw, L. A. and Marcus, M. P. (1999). Text Chunking Using Transformation-Based Learning. In *Natural Language Processing Using Very Large Corpora*, pages 157–176. Springer. (Cited on page 30.)

[360] Rathje, S., Mirea, D.-M., Sucholutsky, I., Marjieh, R., Robertson, C. E., and Van Bavel, J. J. (2024). GPT is an Effective Tool for Multilingual Psychological Text Analysis. *Proceedings of the National Academy of Sciences*, 121(34):e2308950121. (Cited on page 26.)

[361] Ravichander, A., Ghela, S., Wadden, D., and Choi, Y. (2025). HALoGEN: Fantastic LLM Hallucinations and Where to Find Them. *arXiv preprint arXiv:2501.08292*. (Cited on page 120.)

[362] Rennie, S. J., Marcheret, E., Mroueh, Y., Ross, J., and Goel, V. (2017). Self-Critical Sequence Training for Image Captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7008–7024. (Cited on page 114.)

[363] Resnik, P. and Lin, J. (2010). Evaluation of NLP Systems. *The Handbook of Computational Linguistics and Natural Language Processing*, pages 271–295. (Cited on pages 96, 97, 98, and 99.)

[364] Retkowski, F. and Waibel, A. (2024). From Text Segmentation to Smart Chaptering: A Novel Benchmark for Structuring Video Transcriptions. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 406–419. (Cited on page 38.)

[365] Riedl, M. and Biemann, C. (2012). TopicTiling: A Text Segmentation Algorithm Based on LDA. In *Proceedings of ACL 2012 Student Research Workshop*, pages 37–42. (Cited on page 38.)

[366] Roberts, A., Raffel, C., and Shazeer, N. (2020). How Much Knowledge Can You Pack Into the Parameters of a Language Model? *arXiv preprint arXiv:2002.08910*. (Cited on page 17.)

[367] Robertson, S., Zaragoza, H., et al. (2009). The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389. (Cited on page 27.)

[368] Royce, W. W. (1987). Managing the Development of Large Software Systems: Concepts and Techniques. In *Proceedings of the 9th International Conference on Software Engineering*, pages 328–338. (Cited on page 71.)

[369] Rudd, J., Stern, K., and Isensee, S. (1996). Low vs. High-Fidelity Prototyping Debate. *Interactions*, 3(1):76–85. (Cited on page 73.)

[370] Ruder, S., Vulić, I., and Søgaard, A. (2022). Square One Bias in NLP: Towards a Multi-Dimensional Exploration of the Research Manifold. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2340–2354. (Cited on pages 100, 101, and 102.)

[371] Rush, A. M., Chopra, S., and Weston, J. (2015). A Neural Attention Model for Abstractive Sentence Summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389. (Cited on page 46.)

[372] Saeed, A., Khan, H. U., Shankar, A., Imran, T., Khan, D., Kamran, M., and Khan, M. A. (2023). Topic Modeling Based Text Classification Regarding Islamophobia Using Word Embedding and Transformers Techniques. *ACM Transactions on Asian and Low-Resource Language Information Processing*. (Cited on pages 53, 56, and 57.)

[373] Sagan, C. (1995). *Cosmos*. Wings Books. (Cited on page 1.)

[374] Sahami, M., Dumais, S., Heckerman, D., and Horvitz, E. (1998). A Bayesian Approach to Filtering Junk E-Mail. In *Learning for Text Categorization: Papers From the 1998 Workshop*, volume 62, pages 98–105. Citeseer. (Cited on pages 34, 56, and 59.)

[375] Sahoo, P., Singh, A. K., Saha, S., Jain, V., Mondal, S., and Chadha, A. (2024). A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications. *arXiv preprint arXiv:2402.07927*. (Cited on page 156.)

[376] Sai, A. B., Mohankumar, A. K., and Khapra, M. M. (2022). A Survey of Evaluation Metrics Used for NLG Systems. *ACM Computing Surveys (CSUR)*, 55(2):1–39. (Cited on page 130.)

[377] Sainz, O., Campos, J., García-Ferrero, I., Etxaniz, J., de Lacalle, O. L., and Agirre, E. (2023). NLP Evaluation in Trouble: On the Need to Measure LLM Data Contamination for Each Benchmark. *Findings of the Association for Computational Linguistics: EMNLP 2023*. (Cited on page 116.)

[378] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. In *Proceedings of Thirty-Third Conference on Neural Information Processing Systems (NIPS2019)*. (Cited on page 161.)

[379] Saravanos, A. and Curinga, M. X. (2023). Simulating the Software Development Lifecycle: The Waterfall Model. *Applied System Innovation*, 6(6):108. (Cited on page 71.)

[380] Schütze, H., Hull, D. A., and Pedersen, J. O. (1995). A Comparison of Classifiers and Document Representations for the Routing Problem.

In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 229–237. (Cited on page 11.)

[381] Schwartz, R., Dodge, J., Smith, N. A., and Etzioni, O. (2020). Green AI. *Communications of the ACM*, 63(12):54–63. (Cited on pages 62, 183, and 236.)

[382] Scialom, T., Dray, P.-A., Lamprier, S., Piwowarski, B., Staiano, J., et al. (2020). MLSUM: The Multilingual Summarization Corpus. In *2020 Conference on Empirical Methods in Natural Language Processing: Proceedings of the Conference*, pages 8051–8067. Association for Computational Linguistics. (Cited on pages 149, 152, and 170.)

[383] Sehikh, I., Fohr, D., and Illina, I. (2017). Topic Segmentation in ASR Transcripts Using Bidirectional RNNs for Change Detection. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 512–518. IEEE. (Cited on pages 37, 56, 57, and 59.)

[384] Shakarian, P., Koyyalamudi, A., Ngu, N., and Mareedu, L. (2023). An Independent Evaluation of ChatGPT on Mathematical Word Problems (MWP). In *CEUR Workshop Proceedings*, volume 3433. CEUR-WS. (Cited on page 64.)

[385] Shapiro, D. (2020). Small Business Classification by Name: Addressing Gender and Geographic Origin Biases. *arXiv preprint arXiv:2012.10348*. (Cited on page 142.)

[386] Shardlow, M. (2013). A Comparison of Techniques to Automatically Identify Complex Words. In *Proceedings of the Student Research Workshop of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 103–109. (Cited on pages 40, 56, and 57.)

[387] Shardlow, M. (2014a). A Survey of Automated Text Simplification. *International Journal of Advanced Computer Science and Applications*, 4(1):58–70. (Cited on pages 39, 40, and 41.)

[388] Shardlow, M. (2014b). Out in the Open: Finding and Categorising Errors in the Lexical Simplification Pipeline. In *LREC*, pages 1583–1590. (Cited on page 40.)

[389] Shelmanov, A., Fadeeva, E., Tsvigun, A., Tsvigun, I., Xie, Z., Kiselev, I., Daheim, N., Zhang, C., Vazhentsev, A., Sachan, M., et al. (2025). A Head to Predict and a Head to Question: Pre-Trained Uncertainty Quantification Heads for Hallucination Detection in LLM Outputs. *arXiv preprint arXiv:2505.08200*. (Cited on page 120.)

[390] Shi, Y., Jiang, G., Qiu, T., and Yang, D. (2024). AgentRE: An Agent-Based Framework for Navigating Complex Information Landscapes in Relation Extraction. In *Proceedings of the 33rd ACM International Con-*

ference on Information and Knowledge Management, pages 2045–2055. (Cited on page 59.)

[391] Shi, Y., Wang, L., Tian, C., Wang, R., Pei, J., Hussian, A., and Bashir, A. K. (2023). A Chinese Short Text Classification Method for Tax Audit Reports Based on Word Importance and Syntactic Enhancement Bert. *ACM Transactions on Asian and Low-Resource Language Information Processing*. (Cited on page 53.)

[392] Siddharthan, A. (2011). Text Simplification Using Typed Dependencies: A Comparision of the Robustness of Different Generation Strategies. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 2–11. (Cited on page 41.)

[393] Sikosana, M., Ajao, O., and Maudsley-Barton, S. (2024). A Comparative Study of Hybrid Models in Health Misinformation Text Classification. In *Proceedings of the 4th International Workshop on Open Challenges in Online Social Networks*, pages 18–25. (Cited on pages 56, 57, 58, and 60.)

[394] Sindhu Meena, K. and Suriya, S. (2019). A Survey on Supervised and Unsupervised Learning Techniques. In *International Conference on Artificial Intelligence, Smart Grid and Smart City Applications*, pages 627–644. Springer. (Cited on page 99.)

[395] Singh, A., Fry, A., Perelman, A., Tart, A., Ganesh, A., El-Kishky, A., McLaughlin, A., Low, A., Ostrow, A., Ananthram, A., et al. (2025). OpenAI GPT-5 System Card. *arXiv preprint arXiv:2601.03267*. (Cited on page 21.)

[396] Smith, M. S. (2024). Challengers Are Coming for Nvidia's Crown: In AI's Game of Thrones, Don't Count Out the Upstarts. *IEEE Spectrum*, 61(10):40–44. (Cited on page 72.)

[397] Søgaard, A. (2022). Should We Ban English NLP for a Year? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5254–5260. (Cited on pages 100 and 103.)

[398] Sommerville, I. (2015). *Software Engineering*. Pearson International, 10th edition. (Cited on pages 67, 76, and 79.)

[399] Soon, W. M., Ng, H. T., and Lim, D. C. Y. (2001). A Machine Learning Approach to Coreference Resolution of Noun Phrases. *Computational Linguistics*, 27(4):521–544. (Cited on pages 10 and 223.)

[400] Spatharioti, S. E., Rothschild, D. M., Goldstein, D. G., and Hofman, J. M. (2023). Comparing Traditional and LLM-Based Search for Consumer Choice: A Randomized Experiment. *arXiv preprint arXiv:2307.03744*. (Cited on page 117.)

[401] Specia, L. (2010). Translating From Complex to Simplified Sentences. In *Computational Processing of the Portuguese Language: 9th International Conference*, pages 30–39. Springer. (Cited on page 42.)

[402] Specia, L., Jauhar, S. K., and Mihalcea, R. (2012). SemEval-2012 Task 1: English Lexical Simplification. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 347–355. (Cited on page 41.)

[403] Staudinger, M., Kusa, W., Piroi, F., Lipani, A., and Hanbury, A. (2024). A Reproducibility and Generalizability Study of Large Language Models for Query Generation. In *Proceedings of the 2024 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, pages 186–196. (Cited on page 63.)

[404] Steegh, E. and Sileno, G. (2023). No Labels? No Problem! Experiments With Active Learning Strategies for Multi-Class Classification in Imbalanced Low-Resource Settings. In *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law*, pages 277–286. (Cited on page 146.)

[405] Steller, J. J. (2025). KI × DH: Zum Umgang mit ML, Transformern und LLMs in der Digitalen Akademie. (Cited on pages 235 and 236.)

[406] Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F. (2020). Learning to Summarize With Human Feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021. (Cited on page 48.)

[407] Stojkovic, J., Choukse, E., Zhang, C., Goiri, I., and Torrellas, J. (2024). Towards Greener LLMs: Bringing Energy-Efficiency to the Forefront of LLM Inference. *arXiv preprint arXiv:2403.20306*. (Cited on pages 215 and 236.)

[408] Strati, F., Elvinger, P., Kerimoglu, T., and Klimovic, A. (2024). ML Training With Cloud GPU Shortages: Is Cross-Region the Answer? In *Proceedings of the 4th Workshop on Machine Learning and Systems*, pages 107–116. (Cited on page 72.)

[409] Strubell, E., Ganesh, A., and McCallum, A. (2020). Energy and Policy Considerations for Modern Deep Learning Research. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13693–13696. (Cited on page 61.)

[410] Sui, D., Zeng, X., Chen, Y., Liu, K., and Zhao, J. (2023). Joint Entity and Relation Extraction With Set Prediction Networks. *IEEE Transac-*

tions on Neural Networks and Learning Systems. (Cited on page 33.)

[411] Sui, P., Duede, E., Wu, S., and So, R. (2024). Confabulation: The Surprising Value of Large Language Model Hallucinations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14274–14284. (Cited on page 95.)

[412] Sultana, T., Harley, E. R., Adamson, G., and Malik, A. (2022). Extracting Source Information From News Articles: Information Extraction. In *Proceedings of the 2022 6th International Conference on Natural Language Processing and Information Retrieval*, pages 216–221. (Cited on pages 52, 56, 58, 59, 60, and 228.)

[413] Sun, Y., Qiu, H., Zheng, Y., Wang, Z., and Zhang, C. (2020). SIFRank: A New Baseline for Unsupervised Keyphrase Extraction Based on Pre-Trained Language Model. *IEEE Access*, 8:10896–10906. (Cited on pages 122 and 222.)

[414] Tang, D. (2023). Graphic Cards Supply Chain Problems During COVID-19. *Advances in Economics, Management and Political Sciences*, 12:77–83. (Cited on page 72.)

[415] Tänzer, M., Ruder, S., and Rei, M. (2022). Memorisation Versus Generalisation in Pre-Trained Language Models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7564–7578. (Cited on pages 145 and 225.)

[416] Taplin, J. (2017). *Move Fast and Break Things: How Facebook, Google, and Amazon Have Cornered Culture and What It Means for All of Us*. Pan Macmillan. (Cited on page 77.)

[417] Terzi, S., Bouras, A., Dutta, D., Garetti, M., and Kiritsis, D. (2010). Product Lifecycle Management – From Its History to Its New Role. *International Journal of Product Lifecycle Management*, 4(4):360–389. (Cited on page 73.)

[418] Thaler, S. L. (1995). "Virtual Input" Phenomena Within the Death of a Simple Pattern Associator. *Neural Networks*, 8(1):55–65. (Cited on page 93.)

[419] Thomas, E. and Vajjala, S. (2024). Keyphrase Generation: Lessons From a Reproducibility Study. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 9720–9731. (Cited on page 124.)

[420] Tiedemann, J. and Thottingal, S. (2020). OPUS-MT–Building Open Translation Services for the World. In *Annual Conference of the European Association for Machine Translation*, pages 479–480. European Associ-

ation for Machine Translation. (Cited on pages 148 and 193.)

[421] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023). LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971*. (Cited on pages 19, 21, and 43.)

[422] Treviso, M., Lee, J.-U., Ji, T., Aken, B. v., Cao, Q., Ciosici, M. R., Hassid, M., Heafield, K., Hooker, S., Raffel, C., et al. (2023). Efficient Methods for Natural Language Processing: A Survey. *Transactions of the Association for Computational Linguistics*, 11:826–860. (Cited on pages 108 and 109.)

[423] Tuggener, D. (2016). *Incremental Coreference Resolution for German*. PhD thesis, University of Zurich. (Cited on page 142.)

[424] Tuggener, L., Sager, P., Taoudi-Benchekroun, Y., Grewe, B. F., and Stadelmann, T. (2024). So You Want Your Private LLM at Home? A Survey and Benchmark of Methods for Efficient GPTs. In *2024 11th IEEE Swiss Conference on Data Science (SDS)*, pages 205–212. IEEE. (Cited on page 20.)

[425] Tunstall, L., Reimers, N., Jo, U. E. S., Bates, L., Korat, D., Wasserblat, M., and Pereg, O. (2022). Efficient Few-Shot Learning Without Prompts. *arXiv preprint arXiv:2209.11055*. (Cited on pages 145 and 225.)

[426] Uthus, D., Ontanon, S., Ainslie, J., and Guo, M. (2023). MLongT5: A Multilingual and Efficient Text-to-Text Transformer for Longer Sequences. In Bouamor, H., Pino, J., and Bali, K., editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9380–9386, Singapore. Association for Computational Linguistics. (Cited on pages 100, 118, 151, 157, and 171.)

[427] Vapnik, V. (1963). Pattern Recognition Using Generalized Portrait Method. *Automation and Remote Control*, 24(6):774–780. (Cited on page 12.)

[428] Vapnik, V., Golowich, S., and Smola, A. (1996). Support Vector Method for Function Approximation, Regression Estimation and Signal Processing. *Advances in Neural Information Processing Systems*, 9. (Cited on page 225.)

[429] Varab, D. and Schluter, N. (2021). MassiveSumm: A Very Large-scale, Very Multilingual, News Summarisation Dataset. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10150–10161. (Cited on page 138.)

[430] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention Is All You Need. In *Advances in Neural Information Processing Systems*, NIPS'17,

page 6000–6010, Red Hook, NY, USA. Curran Associates Inc. (Cited on pages 2, 15, 62, and 92.)

[431] Vázquez, R., Mickus, T., Zosa, E., Vahtola, T., Tiedemann, J., Sinha, A., Segonne, V., Sánchez-Vega, F., Raganato, A., Libovicky, J., et al. (2025). SemEval-2025 Task 3: Mu-SHROOM„ the Multilingual Shared Task on Hallucinations and Related Observable Overgeneration Mistakes. *arXiv preprint arXiv:2504.11975*. (Cited on page 120.)

[432] Viaene, S. (2013). Data Scientists Aren't Domain Experts. *IT Professional*, 15(6):12–17. (Cited on page 88.)

[433] Viswanathan, V., Gashteovski, K., Gashteovski, K., Lawrence, C., Wu, T., and Neubig, G. (2024). Large Language Models Enable Few-Shot Clustering. *Transactions of the Association for Computational Linguistics*, 12:321–333. (Cited on page 27.)

[434] Wadhwa, S., Amir, S., and Wallace, B. C. (2023). Revisiting Relation Extraction in the Era of Large Language Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, volume 2023, pages 15566–15589. NIH Public Access. (Cited on pages 33, 56, 57, and 59.)

[435] Wang, J., Yue, K., and Duan, L. (2023a). Models and Techniques for Domain Relation Extraction: A Survey. *Journal of Data Science and Intelligent Systems*, 1(2):65–82. (Cited on page 32.)

[436] Wang, S., Zhao, X., Li, B., Ge, B., and Tang, D. (2017). Integrating Extractive and Abstractive Models for Long Text Summarization. In *2017 IEEE International Congress on Big Data (BigData Congress)*, pages 305–312. IEEE. (Cited on page 46.)

[437] Wang, T., Wang, Z., Liu, W., and Shang, J. (2023b). WOT-Class: Weakly Supervised Open-World Text Classification. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 2666–2675. (Cited on pages 51, 53, 56, and 60.)

[438] Wang, X., Wang, Z., Gao, X., Zhang, F., Wu, Y., Xu, Z., Shi, T., Wang, Z., Li, S., Qian, Q., et al. (2024). Searching for Best Practices in Retrieval-Augmented Generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17716–17736, Miami, Florida, USA. Association for Computational Linguistics. (Cited on page 116.)

[439] Webson, A. and Pavlick, E. (2022). Do Prompt-Based Models Really Understand the Meaning of Their Prompts? In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2300–2344. (Cited on page 156.)

[440] Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., et al. (2022). Emergent Abilities of Large Language Models. *arXiv preprint arXiv:2206.07682*. (Cited on page 26.)

[441] Wei, Y. and Ding, Y. (2023). Application of Text Rank Algorithm Fused With LDA in Information Extraction Model. *IEEE Access*, 11:84301–84312. (Cited on pages 45, 56, 58, and 60.)

[442] Wieting, J., Berg-Kirkpatrick, T., Gimpel, K., and Neubig, G. (2019). Beyond BLEU: Training Neural Machine Translation With Semantic Similarity. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4344–4355. (Cited on page 98.)

[443] Wiggins, B. J. and Christopherson, C. D. (2019). The Replication Crisis in Psychology: An Overview for Theoretical and Philosophical Psychology. *Journal of Theoretical and Philosophical Psychology*, 39(4):202. (Cited on page 104.)

[444] Williams-Ceci, S., Macy, M. W., and Naaman, M. (2024). Misinformation Does Not Reduce Trust in Accurate Search Results, but Warning Banners May Backfire. *Scientific Reports*, 14(1):10977. (Cited on page 117.)

[445] Wilson, L. (2022). GPU Prices and Cryptocurrency Returns. *Applied Finance Letters*, 11:2–8. (Cited on page 72.)

[446] Wiseman, S., Shieber, S. M., and Rush, A. M. (2017). Challenges in Data-to-Document Generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263. (Cited on page 93.)

[447] Wolpert, D. H. (1992). Stacked Generalization. *Neural Networks*, 5(2):241–259. (Cited on pages 192, 229, and 237.)

[448] Wu, T., He, S., Liu, J., Sun, S., Liu, K., Han, Q.-L., and Tang, Y. (2023). A Brief Overview of ChatGPT: The History, Status Quo and Potential Future Development. *IEEE/CAA Journal of Automatica Sinica*, 10(5):1122–1136. (Cited on pages 17 and 92.)

[449] Xia, H., Tao, M., and Wang, Y. (2010). Sentiment Text Classification of Customers Reviews on the Web Based on SVM. In *2010 Sixth International Conference on Natural Computation*, volume 7, pages 3633–3637. IEEE. (Cited on pages 38, 56, 57, and 59.)

[450] Xiang, C. and Yangfei, Z. (2023). A Rule-Based Unstructured Information Extraction Model for Announcements of Listed Companies' Stock Increase or Decrease. In *Proceedings of the 2023 7th International Conference on Computing and Data Analysis*, pages 28–34. (Cited on pages 52, 56, 57, 59, and 228.)

[451] Xu, C., Guan, S., Greene, D., Kechadi, M., et al. (2024a). Benchmark Data Contamination of Large Language Models: A Survey. *arXiv preprint arXiv:2406.04244*. (Cited on page 116.)

[452] Xu, D., Chen, W., Peng, W., Zhang, C., Xu, T., Zhao, X., Wu, X., Zheng, Y., Wang, Y., and Chen, E. (2024b). Large Language Models for Generative Information Extraction: A Survey. *Frontiers of Computer Science*, 18(6):186357. (Cited on pages 31, 56, 57, and 59.)

[453] Xu, D. and Tian, Y. (2015). A Comprehensive Survey of Clustering Algorithms. *Annals of Data Science*, 2(2):165–193. (Cited on page 27.)

[454] Xu, W., Callison-Burch, C., and Napoles, C. (2015). Problems in Current Text Simplification Research: New Data Can Help. *Transactions of the Association for Computational Linguistics*, 3:283–297. (Cited on page 42.)

[455] Xu, W., Napoles, C., Pavlick, E., Chen, Q., and Callison-Burch, C. (2016). Optimizing Statistical Machine Translation for Text Simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415. (Cited on pages 40 and 41.)

[456] Xu, Y., Hu, L., Zhao, J., Qiu, Z., Xu, K., Ye, Y., and Gu, H. (2025). A Survey on Multilingual Large Language Models: Corpora, Alignment, and Bias. *Frontiers of Computer Science*, 19(11):1911362. (Cited on page 26.)

[457] Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., and Raffel, C. (2021). mT5: A Massively Multilingual Pre-Trained Text-to-Text Transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498. (Cited on pages 122, 149, 171, and 222.)

[458] Yan, W., Yu, H., Lu, W., and Yin, J. (2024). An Improved Text Similarity Calculation Method Combining TF-IDF and LexRank. In *2024 10th International Conference on Computer and Communications (ICCC)*, pages 498–502. IEEE. (Cited on pages 45, 56, and 59.)

[459] Yang, Y. (1999). An Evaluation of Statistical Approaches to Text Categorization. *Information Retrieval*, 1(1):69–90. (Cited on pages 34 and 35.)

[460] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding. *Advances in Neural Information Processing Systems*, 32. (Cited on page 36.)

[461] Yao, L., Pengzhou, Z., and Chi, Z. (2019a). Research on News Keyword Extraction Technology Based on TF-IDF and TextRank. In

2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS), pages 452–455. IEEE. (Cited on page 122.)

[462] Yao, Y., Duan, J., Xu, K., Cai, Y., Sun, Z., and Zhang, Y. (2024). A Survey on Large Language Lodel (LLM) Security and Privacy: The Good, the Bad, and the Zgly. *High-Confidence Computing*, 4(2):24. (Cited on page 115.)

[463] Yao, Y., Ye, D., Li, P., Han, X., Lin, Y., Liu, Z., Liu, Z., Huang, L., Zhou, J., and Sun, M. (2019b). DocRED: A Large-Scale Document-Level Relation Extraction Dataset. In Korhonen, A., Traum, D., and Màrquez, L., editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 764–777, Florence, Italy. Association for Computational Linguistics. (Cited on page 32.)

[464] Yin, W. and Zubiaga, A. (2021). Towards Generalisable Hate Speech Detection: A Review on Obstacles and Solutions. *PeerJ Computer Science*, 7:e598. (Cited on page 34.)

[465] Yong, W. Y., Jaiswal, R., and Perez Tellez, F. (2023). Explainability in NLP Model: Detection of COVID-19 Twitter Fake News. In *Proceedings of the 2023 Conference on Human Centered Artificial Intelligence: Education and Practice*, pages 29–35. (Cited on page 190.)

[466] Yuan, C., Xie, Q., and Ananiadou, S. (2023). Zero-Shot Temporal Relation Extraction With ChatGPT. In Demner-fushman, D., Ananiadou, S., and Cohen, K., editors, *The 22nd Workshop on Biomedical Natural Language Processing and BioNLP Shared Tasks*, pages 92–102, Toronto, Canada. Association for Computational Linguistics. (Cited on page 33.)

[467] Yuan, Y., Lv, S., Bao, Z., and Li, K. (2022). A Joint Model for Text Classification With BERT-BiLSTM and GCN. In *Proceedings of the 2022 5th International Conference on Artificial Intelligence and Pattern Recognition*, pages 180–186. (Cited on page 53.)

[468] Zamfirescu-Pereira, J. D., Wong, R. Y., Hartmann, B., and Yang, Q. (2023). Why Johnny Can't Prompt: How Non-Ai Experts Try (And Fail) to Design LLM Prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–21. (Cited on page 156.)

[469] Zesch, T. and Gurevych, I. (2009). Approximate Matching for Evaluating Keyphrase Extraction. In *Proceedings of the International Conference RANLP-2009*, pages 484–489. (Cited on page 124.)

[470] Zhai, F., Demberg, V., Shkadzko, P., Shi, W., and Sayeed, A. (2019). A Hybrid Model for Globally Coherent Story Generation. In *Proceedings of the Second Workshop on Storytelling*, pages 34–45. (Cited on page 99.)

[471] Zhang, D., Sensoy, M., Makrehchi, M., Taneva-Popova, B., Gui, L., and He, Y. (2023a). Uncertainty Quantification for Text Classific-

ation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3426–3429. (Cited on pages 53, 56, 59, and 229.)

[472] Zhang, H., Yu, P. S., and Zhang, J. (2024a). A Systematic Survey of Text Summarization: From Statistical Methods to Large Language Models. *ACM Computing Surveys*. (Cited on page 118.)

[473] Zhang, J., Zhao, Y., Saleh, M., and Liu, P. (2020a). PEGASUS: Pre-Training With Extracted Gap-Sentences for Abstractive Summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR. (Cited on pages 54, 149, and 171.)

[474] Zhang, M., Li, X., Yue, S., and Yang, L. (2020b). An Empirical Study of TextRank for Keyword Extraction. *IEEE Access*, 8:178849–178858. (Cited on page 220.)

[475] Zhang, Q., Chen, M., and Liu, L. (2017a). A Review on Entity Relation Extraction. In *2017 Second International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*, pages 178–183. IEEE. (Cited on pages 31 and 32.)

[476] Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. (2020c). BERTScore: Evaluating Text Generation With BERT. In *International Conference on Learning Representations*. (Cited on pages 131 and 132.)

[477] Zhang, T., Ladhak, F., Durmus, E., Liang, P., McKeown, K., and Hashimoto, T. B. (2024b). Benchmarking Large Language Models for News Summarization. *Transactions of the Association for Computational Linguistics*, 12:39–57. (Cited on pages 48, 49, 56, 57, 59, 220, 221, and 231.)

[478] Zhang, X. and Lapata, M. (2017). Sentence Simplification With Deep Reinforcement Learning. In *EMNLP 2017: Conference on Empirical Methods in Natural Language Processing*, pages 584–594. Association for Computational Linguistics. (Cited on page 42.)

[479] Zhang, X., Li, S., Hauer, B., Shi, N., and Kondrak, G. (2023b). Don't Trust ChatGPT When Your Question Is Not in English: A Study of Multilingual Abilities and Types of LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7915–7927. (Cited on pages 101 and 102.)

[480] Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-Level Convolutional Networks for Text Classification. *Advances in Neural Information Processing Systems*, 28. (Cited on page 34.)

[481] Zhang, Y., Zhong, V., Chen, D., Angeli, G., and Manning, C. D. (2017b). Position-Aware Attention and Supervised Data Improve Slot

Filling. In *Conference on Empirical Methods in Natural Language Processing*. (Cited on page 33.)

[482] Zhao, W., Peyrard, M., Liu, F., Gao, Y., Meyer, C. M., and Eger, S. (2019). MoverScore: Text Generation Evaluating With Contextualized Embeddings and Earth Mover Distance. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 563–578. (Cited on pages 122 and 125.)

[483] Zhao, Z. and Chen, P. (2022). To Adapt or to Fine-tune: A Case Study on Abstractive Summarization. In *Proceedings of the 21st Chinese National Conference on Computational Linguistics*, pages 824–835. (Cited on page 165.)

[484] Zheng, S., Huang, J., and Chang, K. (2024). Why Does ChatGPT Fall Short in Providing Truthful Answers? In *I Can't Believe It's Not Better Workshop: Failure Modes in the Age of Foundation Models*. (Cited on page 116.)

[485] Zhong, M., Liu, P., Chen, Y., Wang, D., Qiu, X., and Huang, X.-J. (2020). Extractive Summarization as Text Matching. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6197–6208. (Cited on page 46.)

[486] Zhou, G., Su, J., Zhang, J., and Zhang, M. (2005). Exploring Various Knowledge in Relation Extraction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, pages 427–434. (Cited on pages 32, 56, 57, and 59.)

[487] Zhou, K., Hwang, J., Ren, X., and Sap, M. (2024a). Relying on the Unreliable: The Impact of Language Models' Reluctance to Express Uncertainty. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3623–3643. (Cited on page 115.)

[488] Zhou, X., Chen, Z., Jin, X., and Wang, W. Y. (2021). HULK: An Energy Efficiency Benchmark Platform for Responsible Natural Language Processing. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 329–336. (Cited on page 62.)

[489] Zhou, Y., Keuper, M., and Fritz, M. (2024b). Balancing Diversity and Risk in LLM Sampling: How to Select Your Method and Parameter for Open-Ended Text Generation. *arXiv preprint arXiv:2408.13586*. (Cited on page 63.)

[490] Zhou, Y. and Xu, Y. (2023). Extractive Text Summarization Based on Comparative Learning. In *Proceedings of the 2023 3rd International Con-*

ference on Big Data, Artificial Intelligence and Risk Management, pages 28–32. (Cited on page 60.)

[491] Zhou, Z.-H. (2025). *Ensemble Methods: Foundations and Algorithms*. CRC press. (Cited on page 191.)

[492] Zhu, L., Zheng, C., Guan, W., Li, J., Yang, Y., and Shen, H. T. (2023). Multi-Modal Hashing for Efficient Multimedia Retrieval: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 36(1):239–260. (Cited on page 27.)

[493] Zhu, Z., Bernhard, D., and Gurevych, I. (2010). A Monolingual Tree-Based Translation Model for Sentence Simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, pages 1353–1361. (Cited on page 42.)

[494] Zini, J. E. and Awad, M. (2022). On the Explainability of Natural Language Processing Deep Models. *ACM Computing Surveys*, 55(5):1–31. (Cited on pages 105 and 106.)

University
of Bamberg
Press

The research field of Natural Language Processing (NLP) has experienced a major shift since the introduction of Large Language Models (LLMs). Current research as well as practice of text processing tools is focused mainly on the use and development of LLMs. However, limitations of modern methods have also emerged. Problems regarding the generated texts and the environmental impact of the large-scale use of LLMs are just two of many factors that should be critically analyzed, despite the prevalence of LLMs. These restrictions provide the main motivation for this thesis.

Traditional models as alternatives to LLMs will be discussed from different perspectives. The characterization of traditional models will be progressively developed as features of alternatives to LLMs will emerge in this thesis.

This process will be grounded in experiments, observations and evaluations. Several applications will be presented by surveying the state of the art as well as the current usage of traditional models. The concrete NLP applications comprise information and relation extraction, text classification, text segmentation, text simplification and text summarization. Through improved efficiency as well as enhanced explainability and reproducibility in comparison with neural network-based techniques, these showcases demonstrate the continued relevancy of traditional techniques in today's NLP landscape.

ISBN: 978-3-98989-131-9
9 783989 891319
www.uni-bamberg.de/ubp