Weidinger Sebastian Johannes, Bsc

Analyzing Long-Document Transformer Models For Scientific Abstractive Summarization

Master's Thesis
to achieve the university degree of
Master of Science
Master's degree programme: Software Engineering and Management

submitted to

Graz University of Technology

Supervisor
Assoc.Prof. Dipl.-Ing. Dr.techn. Christian Gütl

Co-Supervisor
Dipl.-Ing Sarah Frank
Institute of Interactive Systems and Data Science, Graz University of Technology & CERN

Institute of Interactive Systems and Data Science
Head: Univ.-Prof. Dipl.-Ing. Dr.techn. Frank Kappe

Graz, August 2024

Weidinger Sebastian Johannes, Bsc

Analyse von Langdokument-Transformermodellen für Wissenschaftliche Abstrakte Zusammenfassungen

Masterarbeit
zur Erlangung des akademischen Grades eines
Diplom-Ingenieur
Masterstudium: Software Engineering and Management

eingereicht an der

Technische Universität Graz

Betreuer
Assoc.Prof. Dipl.-Ing. Dr.techn. Christian Gütl

Mitbetreuer
Dipl.-Ing Sarah Frank
Institute of Interactive Systems and Data Science, Graz University of Technology & CERN

Institute of Interactive Systems and Data Science
Vorstand: Univ.-Prof. Dipl.-Ing. Dr.techn. Frank Kappe

Graz, August 2024

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date	Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Dissertation identisch.

Datum

Unterschrift

Acknowledgments

I would like to take this opportunity to thank everyone who supported me during my master's thesis.

My special thanks go to my supervisor, Christian Gütl, for his tireless support, valuable advice, and motivating manner. Without his help, this work would not have been possible. I would also like to thank my co-supervisor, Sarah Frank, for her constructive suggestions and constant support.

A heartfelt thank you goes to my family and friends, who were always there for me and supported me in every way. In particular, I would like to thank my girlfriend, whose patience and understanding helped me immensely during stressful times.

Finally, I would like to thank the CoDiS Lab for providing me with the opportunity to write my master's thesis in an inspiring and supportive environment.

Thank you all!

Abstract

Automatic Text Summarization (ATS) is a growing field in Natural Language Processing (NLP) due to rapidly expanding text data such as scientific publications. Manual text summarization is a labor-intensive and time-consuming process that demands significant effort and resources. Consequently, ATS has developed within the field of NLP, recognized as one of its most demanding tasks.

This work focuses on summarizing scientific articles, which present several significant challenges. Firstly, the use of specialized terminology and jargon can make it difficult for summarization algorithms to accurately interpret and condense the content. Secondly, the often extensive length of scientific texts requires the summarization system to effectively handle and process large volumes of information while maintaining coherence and relevance. Additionally, scientific articles typically contain complex structures, including numerous citations, figures, and references, which further complicates the summarization process. These challenges necessitate advanced techniques and approaches to ensure the production of high-quality, meaningful summaries.

To produce summaries that resemble human-written texts, we employ state-of-the-art transformer models for abstractive summarization. To address the transformer's disadvantage of quadratic computational cost with increasing input sequence length, we compare an efficient, low-resource, long-document transformer approach called SLED with models having smaller input sizes, investigating the advantages of such a method. Additionally, we examine important aspects such as length or facet controllability and sentence traceability. For this purpose, we used existing and newly scraped attribute-oriented scientific datasets written by human experts. We trained the SLED models and evaluated them in terms of performance and text quality.

We found that for longer and more comprehensive summaries, it is beneficial to take the whole document into account, increasing by up to 1.98 in ROUGE1 on certain summary lengths. Additionally, the informativeness and quality of the text generated by relatively small LLMs are comparable to those of models with a larger number of parameters for this specific task (86.64 versus 90.00 overall score).

Kurzfassung

Automatische Textzusammenfassung (ATS) ist ein wachsendes Gebiet in der Verarbeitung natürlicher Sprache (NLP), das durch die schnell wachsende Menge an Textdaten, wie wissenschaftliche Veröffentlichungen, vorangetrieben wird. Manuelle Textzusammenfassung ist ein arbeitsintensiver und zeitaufwändiger Prozess, der erhebliche Anstrengungen und Ressourcen erfordert. Folglich hat sich ATS innerhalb des NLP-Bereichs entwickelt und wird als eine der anspruchsvollsten Aufgaben anerkannt.

Diese Arbeit konzentriert sich auf die Zusammenfassung wissenschaftlicher Artikel, die mehrere bedeutende Herausforderungen darstellen. Erstens kann die Verwendung spezialisierter Terminologie und Fachjargon es den Zusammenfassungsalgorithmen erschweren, den Inhalt genau zu interpretieren und zu verdichten. Zweitens erfordert die oft beträchtliche Länge wissenschaftlicher Texte, dass das Zusammenfassungssystem große Informationsmengen effektiv verarbeitet und dabei Kohärenz und Relevanz beibehält. Darüber hinaus enthalten wissenschaftliche Artikel typischerweise komplexe Strukturen, einschließlich zahlreicher Zitate, Abbildungen und Referenzen, was den Zusammenfassungsprozess weiter verkompliziert. Diese Herausforderungen erfordern fortschrittliche Techniken und Ansätze, um qualitativ hochwertige und aussagekräftige Zusammenfassungen zu erstellen.

Um Zusammenfassungen zu erstellen, die menschenverfassten Texten ähneln, verwenden wir hochmoderne Transformermodelle für die abstrakte Zusammenfassung. Um den Nachteil des quadratischen Rechenaufwands von Transformern bei zunehmender Eingabesequenzlänge zu adressieren, vergleichen wir einen effizienten, ressourcensparenden Ansatz für Langdokumenten-Transformer namens SLED mit Modellen, die kleinere Eingabegrößen haben, und untersuchen die Vorteile einer solchen Methode. Darüber hinaus untersuchen wir wichtige Aspekte wie Längen- oder Facettensteuerbarkeit und Nachverfolgbarkeit von Sätzen. Zu diesem Zweck verwendeten wir bestehende und neu gesammelte attributorientierte wissenschaftliche Datensätze, die von menschlichen Experten verfasst wurden. Wir trainierten die SLED-Modelle und bewerteten sie hinsichtlich ihrer Leistung und Textqualität.

Wir stellten fest, dass es für längere und umfassendere Zusammenfassungen vorteilhaft ist, das gesamte Dokument zu berücksichtigen, wobei der ROUGE1-Wert bei bestimmten Zusammenfassungslängen um bis zu

1,98 steigt. Zudem sind die Informationsgehalt und Qualität der von relativ kleinen Sprachmodellen generierten Texte vergleichbar mit denen von Modellen mit einer größeren Anzahl von Parametern für diese spezielle Aufgabe (86,64 gegenüber 90,00 Gesamtscore).

Contents

List of Figures

List of Tables

1. Introduction

This chapter addresses the motivation behind this work, as well as outlining the contributions and structure of the thesis.

1.1. Motivation

The volume of text data is rapidly growing in many domains, such as digital libraries, the Web, social networks, and newswire services. According to Bornmann et al. (2021) this trend is also followed by the scientific literature, which has an overall annual growth rate of around 4.10 % based on the number of publications. Therefore, the increase is exponential with a doubling time of 17.3 years, which can be observed in the last decades. To cope with such a huge amount of textual content, methods and tools for compressing and summarising texts are inevitable. Manual text summarization is tedious and time-consuming work that requires a lot of effort and resources. For this reason, Automatic Text Summarization (ATS) emerged from Natural Language Processing (NLP), which is considered one of the most challenging tasks.

Recently, a lot of research has been done on long document abstractive models for text summarization (Koh et al., 2022). Benchmarks such as SCROLLS (Shaham et al., 2022) were created to evaluate the models' capabilities in different long-document tasks. However, larger high-quality datasets with human-written summaries of scientific articles have not yet been provided. In addition, very few datasets include attributes such as aspect or length for the task of controllable summarization. Furthermore, research is also interested in efficient processing of long input texts. According to Tay et al. (2022), there are many approaches to address this problem. However, only a few comparisons to larger, more sophisticated models were made with respect to their text quality and performance.

1.2. Contribution and Research Question

In this work, we provide a newly sourced dataset of summaries written by experts (see Section 4.2). Data come from an open peer review website, described in Section 3.3.3. In addition, the summaries were divided into

several lengths. These length signals can be provided for controllable summarization training. Furthermore, multiple summaries were scraped for each article. Therefore, the dataset can be used for multi-target training and can contribute to better generalisation.

Furthermore, an efficient long-document transformer model called SLED (Ivgi et al., 2023), based on the fusion-in-decoder approach, is trained on the newly sourced scientific dataset. The model is then compared to various extractive and abstractive methods, which are evaluated for their performance and output quality. The size of the model and the associated computational effort are of significant interest. This leads to the formulation of the following research questions (further outlined in Chapter Evaluation).

- **[R1] Research Question 1:** *"Is there a benefit in using efficient long-document models over traditional small LLMs for the task of summarising scientific articles?"*

- **[R2] Research Question 2:** *"Can a similarity search approach that is based on abstractive summaries find and cover information more effectively than traditional extractive methods?"*

- **[R3] Research Question 3:** *"What are the qualitative differences between small LLMs and sophisticated models such as GPT?"*

By addressing these questions this work explores whether efficient long-document models offer advantages over traditional small-input LLMs for summarizing scientific articles, evaluates if abstractive summary-based similarity search outperforms extractive methods, and examines the qualitative differences between small LLMs and advanced models like GPT.

1.3. Thesis Structure

This thesis is organized into several chapters, each addressing a distinct aspect of the research.

Chapter 2 explores the theoretical foundations of automatic text summarization, including model types, text extraction techniques, datasets, and evaluation methods. Additionally, it reviews relevant literature in the field of automatic text summarization with a focus on transformer-based models.

Chapter 3 presents a conceptual architecture of the system's components and outlines the overall goal. It also details and explains the requirements in depth.

Chapter 4 explains the methods employed, including the system architecture and training progress. It also describes the structure of the generated dataset and the similarity search technique.

In Chapter 5 an efficient abstractive long-document transformer approach, called SLED (Ivgi et al., 2023), is analyzed and compared to models with a smaller input size. The results demonstrate the advantages of processing longer text inputs for summaries. Additionally, the influence of guiding signals, such as length and aspect, is studied and investigated. The output of the trained models is compared with sophisticated GPT models with a higher number of parameters (Singh et al., 2023). Readability, performance and text quality such as consistency, coherence, relevance, and fluency were evaluated. The tests show that smaller models can show comparable high performance in specific trained tasks. In addition, an approach to trace sentences in the original input document is investigated (see Section 3.3.4). The outcome of the method was included in the evaluation comparison. The suggested technique contributes to the trustability of the system. Furthermore, the method provides the possibility to provide high-quality extractive recaps in addition to the abstractive summaries.

Chapter 6 presents the lessons learned throughout this work, highlighting key insights gained, challenges encountered, and how these experiences have informed and improved the research process.

Chapter 7 summarizes the key findings of the research and discusses their implications. It also offers a forward-looking perspective on potential future directions and areas for further investigation.

2. Background and Related Work

Automatic text summarization (ATS) is a challenging task in the field of NLP. Much research has been done in the last decades. Many aspects have to be considered, such as the characteristics of the input and output text, the purpose, or the summarization type. Over the years, many approaches have been suggested, ranging from simple statistical methods to sophisticated deep learning techniques. However, some challenges, such as efficient processing of long input text and generating high-quality summaries, have not yet been completely overcome.

This chapter gives a broad overview of current developments in the field of efficient abstractive long-document summarization with respect to text characteristics, datasets, metrics, approaches, and issues.

2.1. Automatic Text Summarization

One of the first works on ATS was presented by Luhn (1958). The Luhn-described technique is used to summarise technical reports to simplify topic identification. The method could assign a significance value to each word and sentence. The high-scoring sentences were then extracted to generate an automatic summarization. Since these first approaches in the 1950s, researchers have tried to steadily improve their techniques to produce computer-generated abstracts that are indistinguishable from human-made summaries. The main focus all ATS systems have in common is that the summary should cover the essential information and topic, avoid redundancy, and be understandable and cohesive from a user's perspective. Although ATS systems are used in many different domains and applications, they can generally be divided into extractive, abstractive, and hybrid methods. In addition to these three main categories, a more fine-grained classification is possible.

Automatic text summarization can be further divided into different aspects, such as the characteristics of the source text (input), the summary as text (output), and the summary usage (purpose) (Hovy and Lin, 1998, G. Sharma and Sharma, 2022).

2.1.1. Input: Characteristics of the Source Text

Characteristics of the input are the number of documents, specificity, as well as genre and scale (Hovy and Lin, 1998).

Number of documents. Single-document versus multi-document: In ATS systems, one or more documents are used as input. In the single-document summarization process, only one input document is handled, whereas in multi-document summarization, multiple topical related documents serve as input to the system (Hovy and Lin, 1998).

Multi-document input is usually more heterogeneous. However, redundancy is one of the main challenges in multi-document summarization, as information is often duplicated (G. Sharma and Sharma, 2022).

Specificity. Domain-specific versus general: If it is clear that the input documents are relevant to a single domain, domain-specific methods could be useful (G. Sharma and Sharma, 2022). Domain-specific summarization can focus on aspects such as content or input formats and derive additional information. Furthermore, less term ambiguity and atypical word and grammar usage can be expected. ATS systems for general domains cannot make such strict assumptions.

Genre and scale: Genres for text summarization can be categorised as summaries of news articles, special domains such as science or law, narrative documents, encyclopedias such as Wikipedia, social networks including blogs, and very short documents such as tweets (G. Sharma and Sharma, 2022). The scale depends on the length of the document. For example, books tend to have longer paragraphs and more chapters, whereas social media posts are brief and include only a few words and sentences. Some text summarization techniques are better applicable to certain genres and scales than others.

2.1.2. Output: Characteristics of the Summary

The final summary can be distinguished by derivation, coherence, partiality, and conventionality (Hovy and Lin, 1998).

Derivation. Extract versus abstract: An extract is an excerpt of the original text that varies from a single word or phrase to multiple sentences or paragraphs. An abstract is a shorter and rewritten version of the initial text. More details on those two summary forms are given in Sections Extractive Text Summarization 2.1.4 and Abstractive Text Summarization 2.1.5.

Coherence. Fluent versus disfluent: A fluent text is correct and clear in grammatical terms. The sentences are coherent and have a cohesive structure. In contrast, disfluent summaries are fragmented and incoherent. Summaries

with a higher degree of fluency are generally perceived as more readable.

Partiality. Neutral versus evaluative: Partiality describes how biased a system is. Neutral summaries are an objective reflection of the input text, whereas evaluative summaries incorporate the system's bias. Evaluation can happen explicitly by using statements of opinion or implicitly by including or omitting biased material.

Conventionality. Fixed versus floating: Fixed summaries follow certain conventions, such as formatting style, and are generated for a particular use or group of readers. Floating summaries vary in settings, readers, and purposes and do not have to conform to any conventions.

2.1.3. Purpose: Characteristics of Summary Use

The summary use depends on the audience, the form and the extensiveness (Hovy and Lin, 1998).

Audience. Generic versus query-oriented: In generic summaries, all major topics are equally important. A query-oriented summary is more user-focused. The user interacts with the system through requests. The final summary includes the desired topics or aspects either explicitly by highlighting or implicitly by excluding certain information.

Form. Indicative versus informative: Indicative summaries exclude the content of the text and provide only basic information about the subject and domain of the text. In contrast, informative summaries include the main content and give the reader a good overview of the essence of a text. Reading an informative summary subsequently allows the reader to outline the main parts of the input text.

Expansiveness. Background versus just-the-news: Depending on the reader's prior knowledge, summary articles present more or less background information. To fill in knowledge gaps, additional details about the place, time, actors, or circumstances are provided. If the assumption is made that the reader has enough background knowledge, only the new and recent topics are included. The former is called background, and the latter just-the-news summary.

2.1.4. Extractive Text Summarization

In extractive text summarization, as the name already implies, sentences and phrases of high importance are extracted from the whole document to form a summary. The resulting excerpt does not include newly formed sentences and consists purely of existing ones from the text. The first attempts at ATS systems in the 1950s were of an extractive nature. Luhn (1958) proposed a weighting of sentences based on word frequency, ignoring common words

with high occurrences. Baxendale (1958) made similar findings, but rather than focusing on individual words, it was suggested that greater importance is given to terms or phrases. The work of Baxendale (1958) also found that sentences of high significance are more likely to be placed at the beginning than at the end of a paragraph.

Later, in the 1960s Edmundson (1969) built on the ideas of Luhn (1958) and Baxendale (1958). Emundson's work introduced four features to create a linear parameterized sentence weighting. In addition to word frequency and sentence positioning, the author proposed cue words and title words. Cue words are words that indicate pragmatic relevance; examples are "significant", "impossible", or "hardly". According to the author, sentences containing these hint words have a higher chance of being of importance. A cue dictionary was created from selected corpus words. In addition, title words are extracted from the title, subtitles, and headings of a document. The words in the titles and headings were shown to have high statistical significance and are highly descriptive. It is hypothesised that a document's author chooses appropriate headings, which summarises the heading's paragraph well. Sentences were weighted on the basis of the number of title words they contained. The total weight of a sentence was then calculated from the four characteristics, namely word frequency, sentence location, cue words, and title words. The sentences were then ranked according to their overall score, and the highest-ranking sentences were chosen for the summary.

In the 1990s, more sophisticated techniques were proposed. Brandow et al. (1995) based their work on the weighting scheme of the frequency times the term inverted document frequency or, in short, $tf * idf$ (Salton and McGill, 1983). Each word weighting depends on the frequency in the document and the overall occurrence in the entire corpus of documents. Words that appear relatively rarely in the corpus have higher weights, and words with high frequency within a document also increase weight. Therefore, unique words that convey meaningful information must be identified. Sentences are then selected for the presence of high-significant (with high $tf * idf$ value) words and other aspects, such as sentence location, words that indicate anaphora, desired summary length, and type of extract.

Mihalcea and Tarau (2004) proposed one of the most impactful works in ATS. The authors introduced a graph-based ranking algorithm called TextRank (Mihalcea and Tarau, 2004) that was inspired by Google PageRank (Page et al., 1999) and Kleinberg's HITS (Kleinberg, 1999) algorithm. Thus, the sentences form the vertices, and the similarities between the sentences represent the edges of the graph. Now, the basic idea of voting comes into play. A connection between a vertex and another symbolises a vote for that other vertex. The higher the number of votes for a vertex, the greater the importance of that vertex. Furthermore, the higher the importance of a

vertex, the greater the importance of the vote. In simple terms, the score of a vertex is measured by the number of voters and their respective score. After applying the ranking algorithm, the sentences are ordered. The highest-ranking sentences are chosen for the resulting summary. A simple and yet effective method that still serves as a baseline for more sophisticated techniques.

More recent work on ATS systems elaborates on embedding methods and deep learning techniques. For example, Kobayashi et al. (2015) and Kågebäck et al. (2014) were among the first to propose a document summarization based on word embeddings. The assumption is that an embedding can capture the meaning of a word. Through its mathematical description, it is also easier to calculate the similarity between two embeddings or a set of embeddings. Among deep learning methods, transformer-based models, such as BERT (Devlin et al., 2019) have become popular. For example, Liu and Lapata (2019) fine-tuned BERT to perform extractive and abstract summaization. But other deep learning models such as recurrent neural networks (RNN) or graph neural networks (GNN) are also commonly used.

2.1.5. Abstractive Text Summarization

Compared to the extraction summarization (ETS) task, abstractive summarization (ABS) is more challenging. Algorithms for ABS systems need to have semantic understanding of the input text. The systems then apply methods from the field of natural language generation (NLG), such as paraphrasing, synonym substitution, or sentence compression (M. Zhang et al., 2022). ABS is therefore conceptually more related to the process of human-generated summaries than ETS. Approaches can be broadly divided into three fields, namely structure-based, semantic-based, and deep-learning-based (Rane and Govilkar, 2019).

Structure-based approaches use structural and logical formats such as templates, trees, graphs, rules, lead, and ontologies. An example of early work based on tree structures is Barzilay and McKeown (2019). The authors used the method of sentence fusion to create the newly generated text. In their work, a dependency tree is built from the input text to represent the sentence structures and the dependencies between the words. The described algorithm then uses the dependency tree of each sentence and rephrases to fuse and align sentences.

Semantic-based methods exploit the semantic representation of the text and use the information obtained for the NLG systems. For example, Moawad and Aref (2012) used the representation of semantic sentences to enrich a semantic graph in their work. The authors then propose to reduce the graph to keep only important information. Finally, the algorithm uses the remaining facts to create new sentences that cover the main content

and semantics of the original text.

Deep learning-based models are the most sophisticated approaches and are based on concepts such as encoder-decoder, attention, RNN, or long-short-term memory (LSTM) models. For a long time, the development of ABS systems made little progress. With emerging technologies such as neural networks and deep learning techniques, improvements have been made in the area of ABS. Rush et al. (2015) was one of the first works to apply deep learning to an ABS task. The constructed model was based on the encoder-decoder architecture that is still used by many state-of-the-art techniques (Guo et al., 2022; Phang et al., 2022; Xiong et al., 2023).

2.1.6. Hybrid Text Summarization

Hybrid text summarization systems combine extractive and abstractive methods. First, sentences are selected using extractive techniques. Second, based on the chosen sentences, an abstractive approach is used for summary generation. Methods such as sentence compression, fusion, and generalisation are commonly employed in hybrid systems (G. Sharma and Sharma, 2022).

Compression. Sentence compression describes the process of reducing the length of the sentence while maintaining the salient information and meaning of the sentence. For example, Zajic et al. (2005) uses a syntactic trimmer to shorten sentences and provide multiple versions for each sentence. The idea is that the trimmer removes irrelevent syntactic elements and phrases. After that, the sentences are weighted according to a linear combination of six features.

Fusion. In sentence fusion and generalisation, merging and fusion techniques are applied to extracted sentences to generate an abstractive summary. Marsi and Krahmer (2005), for example, incorporates this concept in their summarization model. To align the sentence elements at the word, phrase, or substring level, a fusion tree is constructed based on the dependency tree for each sentence. Finally, natural language generation (NLG) algorithms use the fusion tree to automatically merge sentences and generate the summary.

Paraphrasing. In Napoles et al. (2011) the authors apply a paraphrastic approach. First, the model selects the salient sentences from the text to form an extractive summary. Then, a paraphrasing algorithm rewrites the chosen sentences abstractly. In this process, the system uses phrases that shorten the length of the sentence, resulting in a concise and compressed final summary. The authors call this form of sentence compression sentence tightening.

Recent works like Ghadimi and Beigy (2022) propose to use more sophisticated transformer models such as BART (Lewis et al., 2020) or T5 (Raffel et al., 2020) for the abstractive step. The hybrid system leverages these pre-trained language models for the paraphrasing task reaching state-of-the-art

results. These models combine sentence compression, fusion, and paraphrasing. The source input for the ABS step is still the extractive summary.

2.1.7. Long Document Summarization

Summarising long-form textual documents requires more effort compared to generic short text such as news articles. With increasing document length, human labour effort and knowledge requirements increase exponentially. As long texts such as scientific work are increasing, research in this field becomes inevitably. Otherwise, valuable information and findings are overlooked and do not contribute to scientific, social, and economic developments (Koh et al., 2022).

Fundamentals of Long Document Summarization

The summarization of short and long documents differs in three aspects, namely length of document, breath of content, and degree of coherence. Furthermore, long documents tend to be domain-specific, such as scientific papers, and contain more complex formulas and terminologies (Koh et al., 2022).

Length of document. Intuively, a document is considered "long" if an average human needs an excessive amount of time and effort to comprehend the entire text. This intuition cannot be applied in the context of machine learning. For state-of-the-art models, hardware and model limitations are the crucial factor. Their input can have only a size of 512 to 1,024 lexical tokens (Devlin et al., 2019; Lewis et al., 2020; Raffel et al., 2020). The average length of research papers in scientific datasets like ArXiv is around 10,700 tokens (Guo et al. (2022)). Therefore, the maximum size of the article is 10 to 20 times greater than the maximum input size of the regular models. Currently, innovative methods have been proposed to extend the input length to up to 16,384 tokens (Guo et al., 2022; Phang et al., 2022; Xiong et al., 2023).

Breadth of content. Generally, with increasing text length, non-redundant information will also grow. Often, the length of the reference summary increases with the length of the source text. However, the summary length is mostly restricted by the expectations of the average user for a reasonable length. It is empirically shown that the summaries become exponentially shorter in relation to the length growth of the source text (Koh et al., 2022). Therefore, it is inevitable that long text summaries lose important information that covers the central points of the original author.

Additionally, it was shown that users have heterogeneous preferences regarding the importance of content. Due to the relatively shorter summary length and the increasing breadth of content, this problem intensifies for

long text summarization (Koh et al., 2022). Controllable text summarization, described in Section 2.5, could be a solution to this problem.

Degree of coherence. Usually long documents consist of sections, each partially different in content. However, all sections are somehow subject to the main points of the document. Therefore, a summarization model cannot simply copy sentences from the different sections without influencing aspects such as the fluency, redundancy, and coherence of the resulting summary (Koh et al., 2022). Transformer models such as Guo et al. (2022), Phang et al. (2022), and Xiong et al. (2023) can overcome this issue, since they produce a more fluent and readable output.

Scientific Article Summarization

In scientific articles, the results of a scientific study, such as an experiment, survey, or interview, are reported. Typically, a study includes data collection, data analysis, and interpretation of results. There exists no uniform standard for the structure of a scientific article. However, many articles follow an unofficial generic structure, which helps to improve the communication and knowledge exchange between researchers. In general, an article consists of an abstract, an introduction, a related work, a methodology, an experimental results part, a discussion, and a conclusion section (Ibrahim Altmami and El Bachir Menai, 2022).

Abstract. The abstract is located at the beginning of the document. Generally, it has a length of around 150 to 250 words or less (Ibrahim Altmami and El Bachir Menai, 2022). It presents an informative summary of the key points and contribution of the survey without using any citation. In addition, the abstracts provide information in a concise way on applied techniques and methods, materials used, data sets, results, and conclusions.

Introduction. The majority of scientific articles have an introductory section, which is usually between 300 and 500 words or more (Ibrahim Altmami and El Bachir Menai, 2022). This section introduces the topic of the paper and provides essential background information for comprehension purposes.

Related work. The related work section presents former papers related to the subject. The purpose is to update the reader, explain the developments, and place the new survey in the context of the previous literature. The mentioned scientific articles are cited and referenced.

Methodology. This section contains detailed information about the process and the methods used. These details are crucial for other researchers, as they are interested in reimplementing the procedures and reproducing the outcome. It includes specifications on the tools used, preparations, preprocessing techniques, data origins, and post-processing methods.

Experimental results. In this section the results are shown in textual,

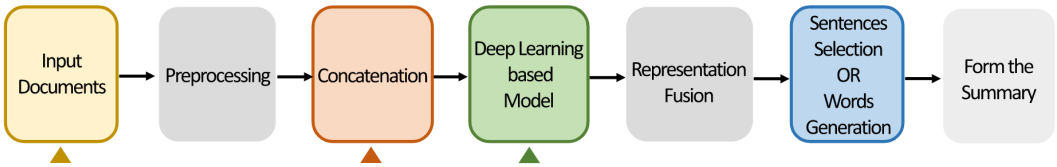


Figure 2.1.: Processing steps of a typical multi-document summarization system. Parts highlighted with a triangle are different from single-document frameworks. (Ma et al., 2022)

illustrative, or tabular form. In addition, outcome of statistical tests are described, often followed by an insightful discussion.

Conclusion. The final part of a survey is the conclusion. It summarises all the main findings and provides an outlook on possible future research.

References. The references of the cited papers are usually located at the end of the document. It is an ordered collection, also called a bibliography, of all the cited articles in the text.

2.1.8. Multi-document Summarization

Multi-document summarization (MDS) compared to single-document summarization (SDS) takes multiple documents into account and produces comprehensive summaries using relations and cross-references between texts (Ma et al., 2022). Usually, documents are written time separated and from different authors. The main challenge of MDS is to handle information from various sources, which is often diverse or redundant. Models must be highly capable of generating coherent, nonredundant, and consistent summaries from complex information. The methods used in MDS are often similar to those in SDS, from traditional algorithms to more sophisticated deep learning approaches. The MDS models are just more specialised in capturing cross-relations between documents.

Process Comparison to Single-document Summarization

According to Ma et al. (2022) the data processing in modern MDS systems differs slightly from that of SDS (see Figure 2.1). In addition to the larger input size and the number of input documents, the MDS models generally need different concatenation strategies. As already mentioned, models are used that exploit the cross-relation information and produce concise semantic-rich summaries. Usually, deep learning-based models are applied as they show better performance.

Apart from having highly diverse input documents with redundant or contradicting cross-relations, MDS also suffers from a lack of sufficient training data and suitable evaluation metrics. However, recently some multi-

document dataset mainly in the scientific domain have been introduced (see Section 2.7.3).

Depending on the task, the source types can be divided into three categories (Ma et al., 2022). First, when there are many short sources, such as in the task of summarising product reviews. Second, relatively few long documents, used, for example, to generate a summary from several news articles. Third, when there are one or few long documents in combination with multiple short documents. This is the case, for example, when there is a long scientific article and several short citation sentences.

Concatenation Methods

In general, there are two common concatenation techniques (Ma et al., 2022). First, flat concatenation, where the input documents are simply transformed into a flat long sequence similarly to SDS tasks. Second, hierarchical concatenation, which keeps the cross-relation information. Generally, deep learning models do not fully exploit the hierarchical relationship. The two common hierarchical concatenation techniques applied are either on the document level or on the word/sentence level within a document cluster. Usually, document-level methods contain subsequent steps before fusion and generation of the final summary. To generate the word or sentence structure, clustering or graph-based techniques are applied.

2.2. Deep Learning Networks

Deep neural networks (DNNs) are the backbone of deep learning (M. Zhang et al., 2022). Generally, these networks consist of multiple hidden layers and are, therefore, called multilayer perceptrons (MLPs). DNNs are defined primarily by their depth and width (Goodfellow et al., 2016). The depth defines the number of layers. Usually, there is one input layer, several hidden layers, and one output layer. The width describes the dimensionality of the hidden layers. Different types of DNNs are used for text summarization tasks, such as recurrent neural networks (RNNs), convolutional neural networks (CNNs), and graph neural networks (GNNs)(M. Zhang et al., 2022). RNN, CNN, and GNN are a particular form of a deep neural network. They differ mainly in their topology and layer connections (Goodfellow et al., 2016).

2.2.1. Recurrent Neural Networks

An RNN is specialised in processing sequential data (Goodfellow et al., 2016). In the following points the structure and properties are explained.

Structure

The network is fed a sequence of values $\boldsymbol{x}^{(t)}$, where t describes the time step ranging from 1 to τ. The values depend sequentially on each other. The prediction of the next time step is based on the current and the previous time step. The nodes of the hidden layer are connected to each other. The input of a hidden layer consists of the input layer and the output of the previous time step, namely the recurrent layer. In Figure 2.2 the described hidden layer is represented by a computational graph that unfolds over time. Each node illustrates the state for a specific time step t. The idea of unfolding represents the chain of events. In general, for an input value $\boldsymbol{x}^{(t)}$ the hidden state $\boldsymbol{h}^{(t)}$ is calculated by

$$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}), \tag{2.1}$$

where $\boldsymbol{x}^{(t-1)}$ is the previous hidden state and $\boldsymbol{\theta}$ is the parameter vector. A description of the parameters is shown in Figure 2.2. The input is parameterized by a weight matrix $\boldsymbol{U}$, the hidden to hidden layer by a weight matrix $\boldsymbol{W}$ and the output layer by a weight matrix $\boldsymbol{V}$. The network computes the output $\boldsymbol{o}$. The loss $\boldsymbol{L}$ is then calculated by comparing the predicted output $\hat{\boldsymbol{y}} = softmax(\boldsymbol{o})$ with the actual output $\boldsymbol{y}$.

Properties

RNNs also make use of the concept of parameter sharing (Goodfellow et al., 2016). It helps to generalise and to share statistical characteristics across different-sized sequences and positions in time, and in the case where the particular information appears at different positions within a sequence.

Due to these features, RNN can efficiently extract temporal or sequential semantic information from sequences (M. Zhang et al., 2022). RNN-based deep learning models are used for several NLP tasks, such as information extraction, machine translation, text summarization, or time series analysis. Nevertheless, RNN also have some disadvantages. When the sequences are too long, the gradient can explode or disappear. Long short term memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) mitigate the problem of long-term dependencies by introducing input, forget, and output gates. Cho et al. (2014) further adapted the architecture using an update gate instead of an input and forget gate, a so-called gate recurrent unit (GRU). Bidirectional RNN, as the name suggests, allows the information to flow forward and backward via two connected hidden layers. Hence, the output layer receives information from the past and the future states simultaneously. These networks and units are described as Bi-RNN, Bi-LSTM, and Bi-GRU.

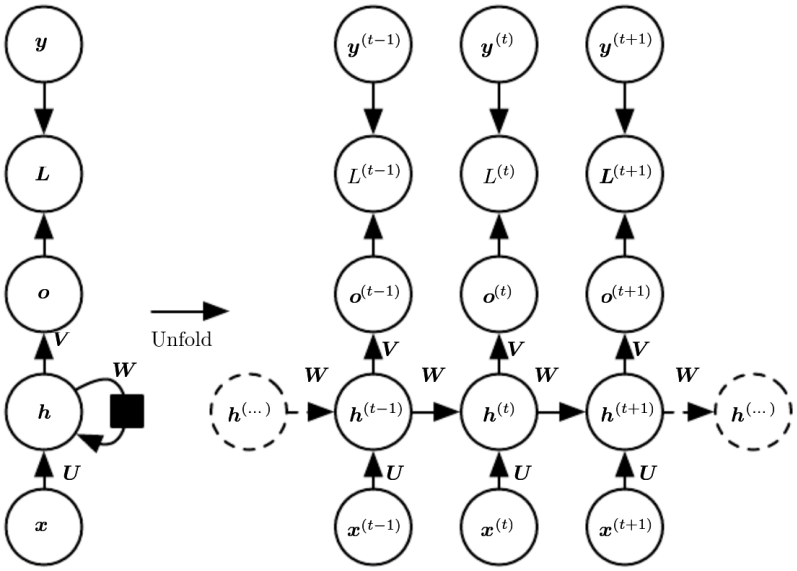


Figure 2.2.: An illustration of the time-unfolded computational graph. Left, a hidden layer with a recurrent layer. Right, the same graph unfolded over time. (Goodfellow et al., 2016)

2.2.2. Convolutional Neural Networks

CNN are specifically designed to process grid-like data (Goodfellow et al., 2016). The following points explain the structure and properties.

Structure

For example, sequential or time series data can be seen as a 1-D grid with a value at each time step. The fundamental structure of a CNN consists of convolution, activation, and pooling. First, a convolution is applied to extract features. Second, after activation, the extracted features are pooled to retrieve features with different levels of complexity. Figure 2.3 shows a basic framework for CNNs. It consists of an input layer, M convolutional layers followed by b pooling layers. Such units, a combination of a convolutional layer and a pooling layer, can appear N times. After that, typically K fully connected layers serve as a classifier.

Properties

Convolution exploits three principles, namely sparse interactions, parameter sharing, and equivariant representations (Goodfellow et al., 2016). Usually, in a neural network, all input units are connected to each output unit. In CNN a kernel smaller than the input size is used to process the data. By doing so, significant small-scale features can be revealed. In addition to improving statistical efficiency, this method requires fewer parameters, and

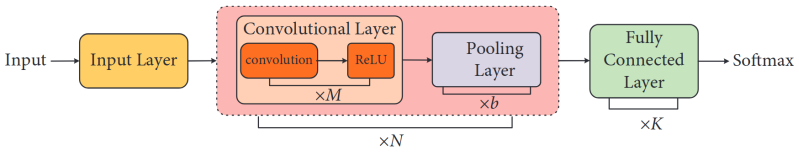


Figure 2.3.: Framework for convolutional neural networks. (M. Zhang et al., 2022)

therefore less memory. In CNN, the parameters are shared. The kernel with the same parameters is applied at every position of the input, in contrast to learning a different set of parameters for every location. This method further improves memory requirements. Parameter sharing enables an additional property, namely equivariance to translation. If the input of an equivariant function changes, the output changes in the same way. For example, if we apply a convolution to sequential or time series data, it generates a timeline that reveals at which time step specific features appear.

2.2.3. Graph Neural Networks

When developing GNNs, the graph structure used must be defined (Salchner and Jatowt, 2022). The structure and properties are explained in the following points.

Structure

A graph is represented by its nodes V and edges E, formally written as $G = (V, E)$. A node $v_i \in V$ is connected to another node $v_j \in V$ through an edge $e \in E$, also written as $e_{i,j} = (v_i, v_j)$. The graph has $n = |V|$ nodes and $m = |E|$ edges. A feature matrix can be defined where $X \in R^{n \times d}$. Each node i is described by a feature vector $x_i \in R^d$ with dimension d.

Data can be inherently structured or inherently unstructured (Salchner and Jatowt, 2022). In the case of text, the data have an implicit structure. However, it is not directly accessible as a graph. Usually, in ATS the text is converted to a graph by using words or sentences as nodes. It is common to use heterogeneous instead of homogeneous graphs for ATS tasks. The reason is that heterogeneous graphs can contain different structures. Additional structural information can be introduced to the graph using directed or indirected edges. Edges indicate the flow of information. Hence, the word or sentence order could be explicitly encoded.

Properties

GNNs differ in three aspects from general neural networks: input, output, and information aggregation (Salchner and Jatowt, 2022). The input is always

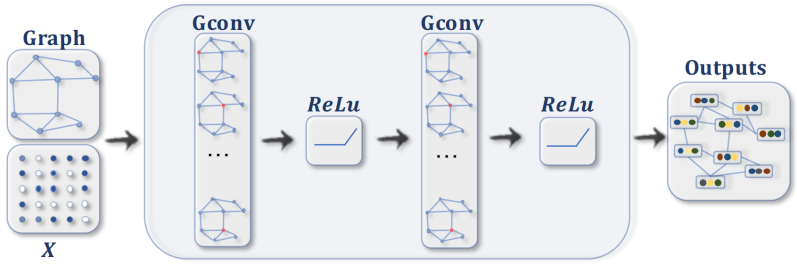


Figure 2.4.: A graph neural network (GNN) with multiple graph convolutional (Gconv) layers. A typical network for node classification. (Wu et al., 2021)

graph-structured. The output can be at node level, edge level, or graph level. Prediction and classification tasks are performed at the node or edge level. The output at the graph level is used for subgraph prediction and graph segmentation. In the case of ATS binary classification is performed to select a subset of words, phrases, or sentences. There are two ways to aggregate information, using spatial or spectral convolution. Spatial convolution is based on the convolution operator with the extension of processing graph structures. Spectral convolution builds on the idea of graph signal processing. For ATS spatial convolution is preferred over spectral convolution due to its efficiency, flexibility, and better generalisation properties. Instead of processing the entire graph at once, it can also handle batches of nodes. This is advantageous for large graphs, such as long texts in ATS. GNNs are only used for ETS as they can only select a subset of sentences and do not have a generative mechanism.

The spatial convolution in GNN is similar to the convolution in CNN (Salchner and Jatowt, 2022). However, convolution in CNN performs on regular grid data such as 1D or 2D arrays, for example, images. In the case of graphs, a target node has a varying number of unordered neighbours with different feature vector representations. Hence, graph data contain irregular grids and subsequently have to be processed in a different way. In GNNs convoultion is performed by information propagation, the so-called message passing. Hereby, nodes exchange messages, or in other words perform convolution with their neighbours for a number of iterations. The information is spread throughout the graph. This means that the higher the number of iterations, the further the propagation of information.

Veličković et al. (2018) proposed a graph attention network (GAT) that uses the attention mechanism similar to transformer networks (Vaswani et al., 2017). GATs give each neighbour of a node an attention score that reflects the importance of that neighbour and its messages. Brody et al. (2021) further developed a specialised form called GATv2. GNN models for ATS widely use GTA layers because of their performance.

When GNNs are combined with other models, a pooling layer is usually

necessary (Salchner and Jatowt, 2022). The pooling operation generates either a global representation or a subset of the graph by processing the features of the nodes. Figure 2.4 shows a simple illustration of a convolutional GNN (convGNN), which is used for node classification (Wu et al., 2021). The graph convolutional layer (Gconv) encodes the hidden representation of each node. This is done by aggregating the feature information of the node's neighbours. As a next step, a nonlinear activation is applied. Each node's final hidden representation receives information from further neighbourhoods by stacking several layers.

2.3. Encoder-Decoder Framework

The Seq2Seq framework, also called the encoder-decoder framework, is a widely applied technique in NLP tasks, such as machine translation, speech recognition, or question-answer applications (M. Zhang et al., 2022). First, the encoder parses the input sequence and encodes it into a high-dimensional representation in the form of a feature vector. After that, the decoder interprets the feature vector and decodes it, generating the output sequence. The encoder-decoder framework is the basis for many DL models in ABS.

2.3.1. Encoder-Decoder Systems with Attention Mechanism

In general, attention is a selective process that filters essential information (de Santana Correia and Colombini, 2022). Formally, the attention operation can be described as a mapping function that involves a query, a key-value pair, and an output (Vaswani et al., 2017). Thereby, a query combined with a key-value pair is mapped to an output. The output is generated by summing up the weighted values. The weight describes the compatibility between the query and the corresponding key and is computed by a function.

Issues in Classical Neural Networks

According to de Santana Correia and Colombini (2022), the attention mechanism facilitates improvements in DNN and helped to achieve state-of-the-art performance in many application domains such as NLP and computer vision. The main reason for the success of attention-based models is their ability to mitigate three classic issues of classical neural networks (de Santana Correia and Colombini, 2022). First, models with attention layers can better process long temporal dependencies. Classical neural networks, as well as RNNs and LSTMs, have difficulties in retaining earlier information due to memory limitations. Second, attention-based models have less problems in processing

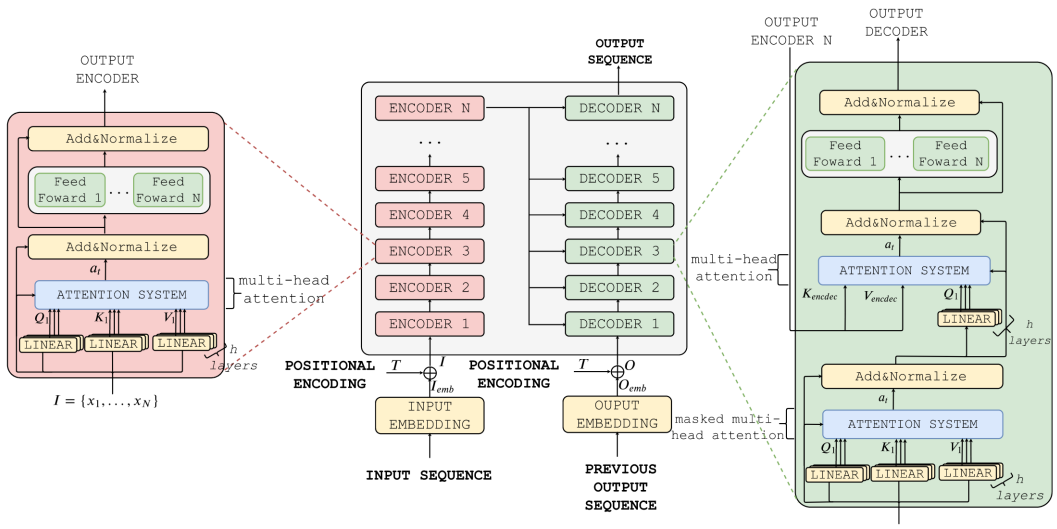


Figure 2.5.: Architecture of a transformer model. It consists of multiple stacked attentional units with an encoder and decoder. The encoder processes the input sequence and extracts salient information. The decoder generates the output sequence. Both encoder and decoder are build up of multiple heads of self-attention, fusion, and normalisation layers. (de Santana Correia and Colombini, 2022)

long-sequenced input. Deep-layered classical neural networks sometimes suffer from the issue of vanishing gradients. Third, models based on the attention mechanism have better noise-reduction properties compared to classical neural networks.

Categories of Attention Mechanism

As reported by de Santana Correia and Colombini (2022), the mechanism can be classified into soft attention, hard attention, and self-attention.

Soft attention considers the interdependence between the input and target. It assigns a weight between 0 and 1 to each element in the input using the softmax function. Hence, the attentional model is deterministic and differentiable. However, for large inputs, soft attention mechanisms are computationally intensive.

Hard attention, similar to soft attention, assumes also interdependence between the network's input and the target, but assigns only 0 or 1 as a weight. Hence, only parts of the input are considered. The hard attention mechanism is a stochastic process and therefore is the model not differentiable. Reinforcement learning methods are unavoidable in training models with hard attention layers. Since, unlike soft attention, only parts of the input are processed, less inference time and computational costs are needed.

Self-attention makes the assumption that there is interdependence between the input elements. The mechanism learns which interactions it

should pay attention to by quantifying them through simple matrix calculations. These calculations can be parallelisable, a reason why self-attention is preferred over soft and hard attention for long inputs.

Improvements by Attention Mechanisms

Bahdanau et al. (2015) made remarkable improvements in machine translation with an RNN-based encoder-decoder architecture with attention layers, called RNNSearch. Classical encoder-decoder suffer from the bottleneck problem. Hereby, the encoder encodes the input sentence into a fixed-length vector. The issue is that with increasing sentence length, the performance decreases. Bahdanau et al. (2015) reduces the bottleneck with the proposed encoder-decoder. The encoder is a BiRNN, and the decoder consists of an RNN combined with attention layers. The output is generated by computing the probability distribution of the output symbols from a context vector. Rather than encoding the input sentence into a fixed-sized vector, it produces a sequence of vectors from which a subset is chosen to generate the output. The model with the introduced attention layers achieved better results than the classical encoder-decoder models.

Rush et al. (2015) is one of the first to use the encoder-decoder framework for an ABS task. Instead of encoding the input with Bag-of-Words (BoW), they applied an attention-based encoding. In addition, they used a beam search strategy, alternatively to greedy decoding.

Vaswani et al. (2017) proposed a breakthrough end-to-end attention-based architecture widely used in the current literature, called transformer. The model contains multiple units of self-attention in parallel, so-called multi-head attention. This profound architecture allows the model to jointly learn different linguistic features and representations. Transformer models do not contain recurrences or convolutions. Hence, they cannot exploit the information about the order of the sequences. The authors introduce a technique for adding the relative and absolute positions to the embeddings. The positional encoding is computed using sinusoidal functions and additively combined with the word embedding. The general architecture of a transformer model is shown in Figure 2.5. Many state-of-the-art models (Guo et al., 2022; Phang et al., 2022; Xiong et al., 2023) follow this architecture in NLP for ATS tasks because it outperforms several other model designs (de Santana Correia and Colombini, 2022).

2.3.2. Hierarchical Encoder-Decoder Models

The basic single-layer encoder-decoder cannot fully grasp the context in a long document, also referred to as the problem of long distance dependence (M. Zhang et al., 2022). However, long documents consist of an inherent hier-

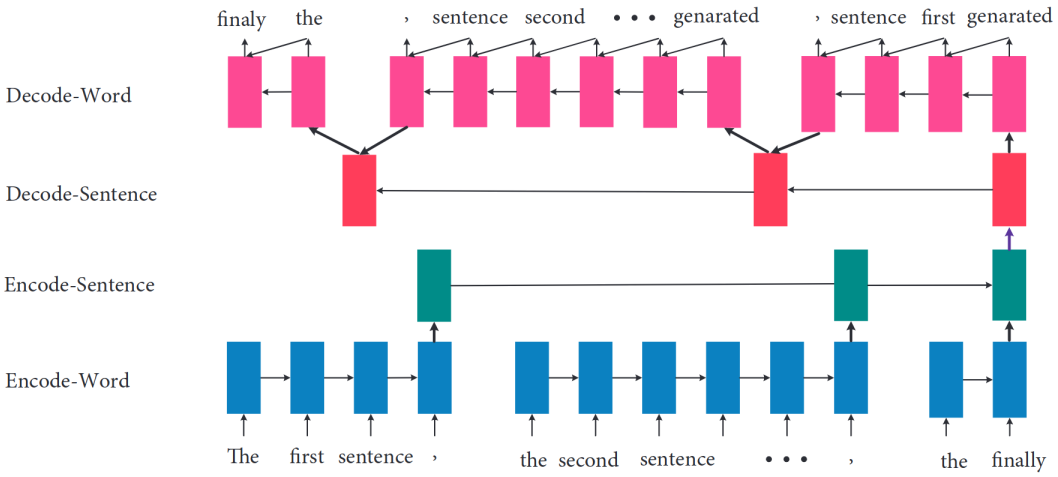


Figure 2.6.: The fundamental design of a hierarchical encoder-decoder. It is divided in word and sentence levels. First, each word is encoded and processed into a sentence. Then, the sentences are combined to generate a document representation. The document representation is decoded to produce the output sequence. (M. Zhang et al., 2022)

archical structure of word and sentence levels. Hence, researchers proposed a hierarchical encoder-decoder to mine this structure and to minimise the long-distance dependence issue. Figure 2.6 shows the fundamental architecture of the hierarchical encoder-decoder for ABS tasks. Hierarchical models perform significantly better with respect to informativity and readability compared to similar models. However, basic hierarchical encoder-decoder models capture structural information only on the word-sentence, but not on global document level. W. Li et al. (2018) exploit the structure of the document and increase the performance of ABS models. They introduced structural compression and coverage terms to regularise the generation of the output. These mechanisms ensure concise and diverse sentences.

2.3.3. Out-Of-Vocabulary and Repetition Problem

Out-of-vocabulary (OOV) and repetition problems, common in ABS, deteriorate the performance of models (M. Zhang et al., 2022). Usually, transformer models have a limited vocabulary size for their output sequence (Guo et al., 2022; Phang et al., 2022; Xiong et al., 2023). Hence, OOV words in the input have an impact on the quality of the summary output. However, it was found that nearly all OOV words appear with a low frequency in the source text. The researchers came up with the idea of a copy mechanism that finds the corresponding word in the input and copies it to the output. To minimise the repetition problem, they introduced a penalisation for duplicated words and phrases. Hence, the model avoids repetitive output.

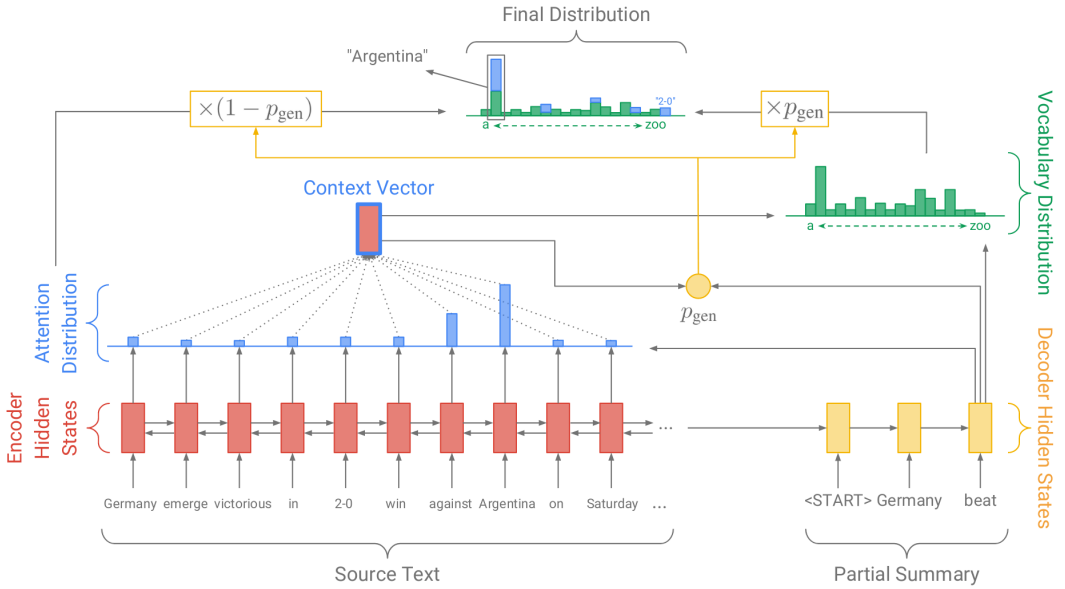


Figure 2.7.: The pointer generator model. A generation probability p_{gen} is calculated, which is used to decide whether to generate words or copy words from the input text. For prediction, the attention and vocabulary distribution are combined to get the final distribution. (See et al. (2017))

Alleviation of the Out-Of-Vocabulary Problem

Gu et al. (2016) applied the copy mechanism in an ABS model, called CopyNet. The network has an encoder-decoder architecture with a copy technique integrated in the decoder. The selected words and phrases in the input text are taken and appropriately placed in the right position in the output text.

See et al. (2017) proposed a pointer generator (PG) model for ABS. The network uses pointers to the selected words in the input text. These pointers support precise information replication without reducing generation abilities. The authors also incorporated a coverage mechanism to penalise repetitive behaviour. However, PG has some limitations. First, the model can only copy the exact word or phrase without changing its form. Hence, words cannot be aligned to fit a specific context. Second, the model is biased towards copying sentences from the input text. Thus, the network produces fewer novel sentences. To alleviate these two problems X. Shen et al. (2019) introduced a more generalised pointer generator (GPG) network. Rather than a simple copying of exact words, the mechanism allows word editing by a learnt relation embedding. GPG learns better and richer alignments than the simpler copy technique of PG.

Tokenization of Transformer Models

It should be mentioned that many transformer-based models use specialised tokenization. For example, the BERT (Devlin et al., 2019) architecture applies word piece tokenization initially outlined in Schuster and Nakajima (2012). Instead of word-level, the encoder provides subword representation. Unknown words are sliced into subwords, even to the single-character level. Another widely applied tokenization technique is the byte-pair encoding (BPE) proposed in Sennrich et al. (2016), which is used in models such as BART (Lewis et al. (2020)). BPE encodes the most frequent character combinations, starting with a single symbol. Sentence piece tokenization introduced by Kudo and Richardson (2018) is an additional highly relevant tokenization algorithm. This method is used as a simple language-independent subword tokenizer. The tokenizer is generally applicable to different languages. For example, T5 (Raffel et al., 2020) operates on the principle of sentence piece tokenization. In general, models based on the mentioned tokenization algorithms are more robust towards OOV words, since models trained with such tokenizer are able to adaptively generate new unseen words.

2.3.4. Hallucination Problem

Transformer-based sequence-to-sequence models have led to a performance boost in recent years. They produce more fluent and coherent texts than other models. However, these advanced systems are not perfect and are prone to hallucination. This leads to performance degradation and makes these systems unsuitable in real-world applications (M. Zhang et al., 2022).

Hallucination Definition

In simple terms, hallucination describes the scenario in which something is perceived as legitimate, even though it is unreal (Ji et al., 2022). In the context of NLP, hallucinated content is defined as nonsensical or unfaithful with respect to the source text. Hallucinations can be categorised as intrinsic or extrinsic. The former specifies the case where the output is in conflict with the source input. The latter is used in the context in which the generated output cannot be verified or contradicted by the source information. However, extrinsic hallucination is not always erroneous since factual correct information could also be provided from external knowledge sources. Due to its unverifiable nature, it is still carefully treated.

Faithfulness and Factuality

Hallucination is closely related to the terms faithfulness and factuality (Ji et al., 2022). Faithfulness is an antonym to hallucination. Hence, when re-

searchers try to minimise hallucination, they inevitably also aim to maximise the faithfulness of their models. Factuality describes the degree of correctness with respect to the underlying facts. There exist different definitions of the expressions fact and factfuality in terms of their source. Some compare the generated output to facts in the input source, others to generally accepted facts in world knowledge.

Hallucination Metrics

The hullucination effect plays an enormous role in ABS. Approximately 25% of the output summaries generated by recent models were found to contain hallucinated content (Ji et al., 2022). In addition, even though models can score high in ROUGE, they can produce highly hallucinatory content. Hence, it is important to test with specific metrics how faithful a model is.

There exist several approaches for hallucination metrics in ABS, categorised in unsupervised and semi-supervised. Unsupervised metrics are based on information extraction (IE), natural language inference (NLI), or question-answer (QA) (Ji et al., 2022). Section 2.6.2 describes the metrics in more detail.

Hallucination Mitigation Methods

The methods for minimising hallucinations can be divided on the basis of architecture, training, or post-processing (Ji et al., 2022).

Architecture. By adapting the architecture, the models can have a minimised tendency to produce hallucinative content (Ji et al., 2022). The researchers changed the encoder, decoder, or entire encoder-decoder framework. For example, the model proposed in Zhu et al. (2021) extracts information from the source input and automatically constructs relation tuples via a GNN. These encoded factual relations are incorporated into the decoder to support fact-aware summarization (shown in Figure 2.8). Aralikatte et al. (2021) came up with a focus-attention mechanism in the decoder that learns to reinforce attention to tokens with topical similarity to words in the source text. H. Li et al. (2018) modify both the encoder and the decoder, resulting in an entailment-aware model that integrates the entailment knowledge into the summarization process. All of the mentioned methods efficiently generate more faithful output summaries.

Training. Training methods improve models with respect to hallucination without modifying the model design (Ji et al., 2022). Cao and Wang (2021) proposes a technique based on contrastive learning. The training corpus is divided into faithful and hallucinatory samples. The system learns to differentiate between positive and negative data.

Post-processing. Post-processing methods edit the output after summary

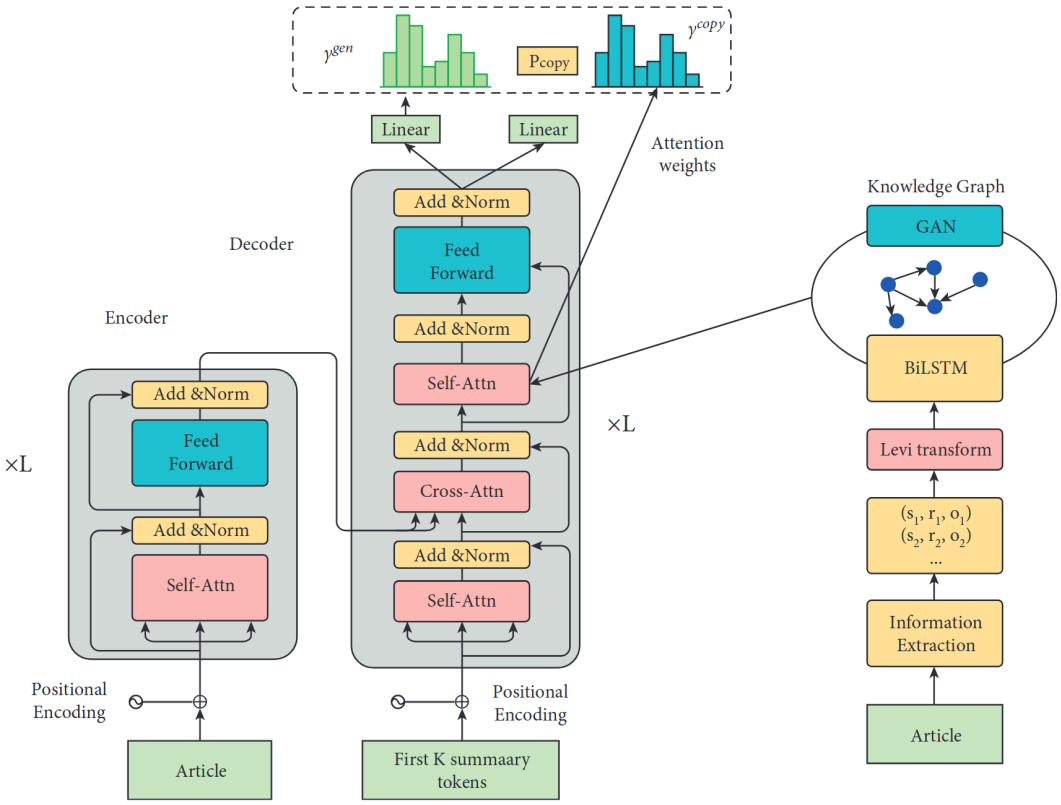


Figure 2.8.: Model FASum. It consists of an encoder and a decoder. In addition, relation tuples are extracted and incorporated into the decoder. (Zhu et al., 2021)

generation to reduce hallucination (Ji et al., 2022). For example, Z. Zhao et al. (2020) introduces the HERMAN system, which recognises and verifies quantities such as date, time, or percentage. Chen et al. (2021) presents a different approach based on candidate generation. The model selects the best candidates for named entities from the source document and replaces the corresponding ones in the summary text.

2.3.5. Interpretability of Transformer Networks

Neural network models are insufficiently interpretable (de Santana Correia and Colombini, 2022). However, there is a strong urge in academic and industry to increase the explainability of models.

Reasons for Interpretable Systems

Interpretability is important for aspects such as critical decision-making, failure analysis, verification, or model improvements (de Santana Correia and Colombini, 2022).

Critical decisions. In some critical fields, such as medical analysis, stock markets, or autonomous driving, decisions must be made with great urgency and precision. In addition, explanations of the decision process are important to increase the confirmability and traceability of the systems.

Failure analysis. In case a system is faulty, interpretable models can be retrospectively analysed and flawed decisions can be identified. The conclusions drawn provide essential information for improving the system.

Verification. It is crucial to verify the results and performance of a model. Robustness and generalisability are essential properties of systems. However, it is often challenging to interpret influences such as spurious correlations on performance or identify causalities for results that vary in their outcome.

Model improvements. Interpretability can be a guide for decisions regarding model improvement and provides a starting point for enhancement measures.

Attention Mechanism for Interpretability

The possibility of using the attention mechanism as an explanation tool depends on the definition of the term explainability (de Santana Correia and Colombini, 2022). It was shown that attention is sufficient to interpret the plausible explainability. However, a faithful and correct explanation for the connection between input and output is not always possible. Experiments have also revealed that the attentional weights of models for single sequence tasks do not indicate the reasoning for the model's decision. In contrast, models for pair-sequence tasks showed a strong indication that the attentional weights are important for the explanation of the reasoning.

In general, it was found that the efficient explainability of attention-based models depends on the dataset used and the characteristics of the system (de Santana Correia and Colombini, 2022).

Explainability in Abstract Text Summarization

Explainability is also an important topic in ABS since it greatly contributes to the faithfulness and reliability of the model. As already discussed, it is still difficult to interpret the reasoning and generation process of end-to-end transformer models. For better understanding and comprehension, it is inevitable to reveal where the errors occurred. If it were clear where the mistake originates, countermeasures or error correction mechanisms could be implemented.

Current work on explainable ABS often concentrates on highlighting phrases and sentences from the original text (H. Li et al., 2021; Z. Zhao et al., 2020). This is often accomplished by using hybrid text summarization approaches. Hence, important sentences are first extracted and then abstracted.

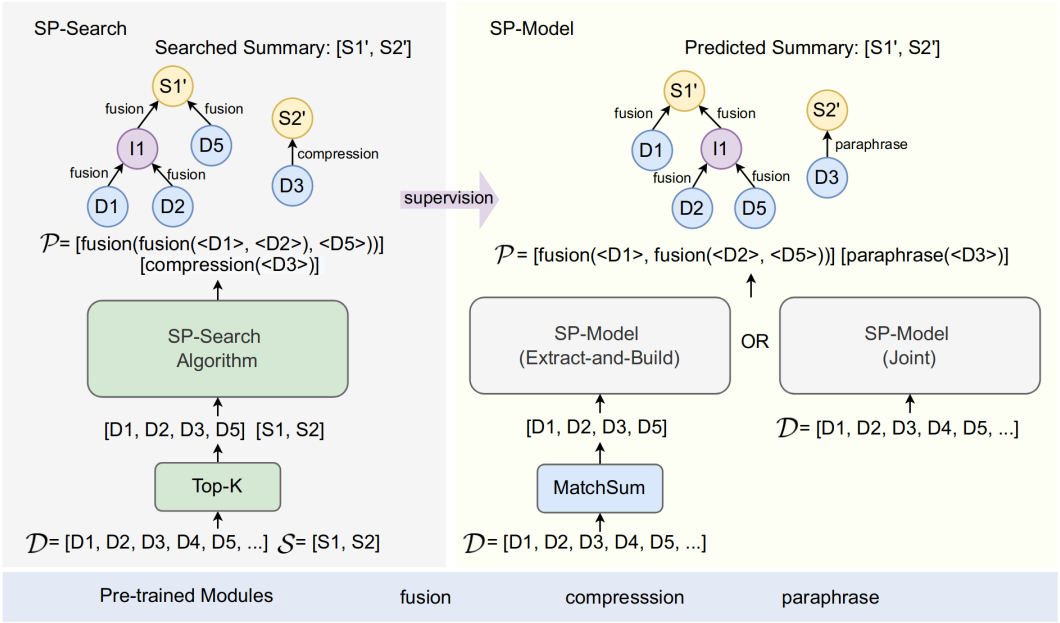


Figure 2.9.: SP-Search algorithms. The algorithm combines sentence fusion, compression and paraphrasing and generates a graphical description. (Saha et al., 2022)

However, these methods usually perform worse than pure abstractive techniques. In addition, a pure textual explanation often does not fully describe the summary generation process.

Recently, some methods emerged including graphical descriptions. For example, Saha et al., 2022 propose an interpretable modular framework called SP-Search. Every sentence in the summary is thereby encoded by a binary tree. The tree represents the generation process using operations such as sentence paraphrasing, compression, and fusion. For each operation, a specially trained model is applied. This approach should simulate the intuitive human summarization process. A major advantage of this method is that every intermediate step can be analysed and explained. SP-Search was implemented for two tasks, to find the sentences in the source text from which the summary originates, and to generate summaries including the intermediate steps. Figure 2.9 shows the principles for both sentence search and summary generation. For the searched summary are the top-k sentences selected from the original text. Then the SP-Search algorithm transforms the sentence selection into the interpretable graph structure. The predicted summary is generated by either the extract-and-build approach or directly from the document. The summary search algorithm could be used to supervise the generation process of other models and make them more interpretable.

2.4. Efficient Long-Document Transformer

Recent research on transformer architectures has put a focus on long-sequence processing. Researchers adapted widely used transformer models (Lewis et al., 2020; Raffel et al., 2020; J. Zhang et al., 2020) for the task of long-document summarization. Some state-of-the-art publicy available models include BART-LS (Xiong et al., 2023), Pegasus-X (Phang et al., 2022), or LongT5 (Guo et al., 2022). Despite their optimisation methods, the models still have a large number of parameters. This leads to high computational costs. To overcome this problem, the researchers proposed techniques using fewer resources and efficient computations.

2.4.1. Computational Cost of Transformer

The main computational costs of Transformer are to compute the attention matrix and the feed-forward layers at every transformer block (Tay et al., 2022), also shown in Figure 2.5. The computational cost for the attention matrix is quadratic in their input sequence. The self-attention mechanism is generated by a query Q, key K, and value V pair. The output matrix is computed by the following formular:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.2}$$

, where d is the dimension of the input key. Therefore, the computation requires $N \times N$ time and memory. With increasing input length, the computation becomes infeasible. Furthermore, the feedforward layer after each transformer block (shown in Figure 2.5) demands around half the computation time. The computational complexity of the layer increases linearly. However, in general, it is still expensive. Therefore, a lot of research is going into the development of efficient models that can be scaled up without enlargening computational costs tremendously.

2.4.2. Low-Ressource Efficient Approaches

According to the survey of Tay et al., 2022 there are multiple approaches to address the problem of memory and time complexity. Most of the proposed techniques try to reduce the memory usage of self-attention methods. The core approaches are briefly described below.

Fixed Patterns. One of the simplest and earliest techniques is to introduce sparsity to the attention layers by applying fixed patterns such as local windows and block patterns using fixed strides.

Combination of Patterns. Similarly to fixed patters, combination of patterns reduces memory complexity. However, it is possible to gain better

coverage of the self-attention mechanism by efficient combination of patterns.

Learnable Patterns. This class of patterns still uses fixed patterns. In addition, a sorting or clustering method is learnt to determine token relevance. The goal of these models is to learn patterns in a data-driven way.

Neural Memory. Thereby, a learnable memory module is used. The module has access to multiple sequence tokens and acts like a pooling operation to compress the input sequence.

Low-Rank Methods. This method uses projecting techniques to lower the dimension of keys and values. Instead of a $N \times N$ matrix multiplication, it is now a $N \times k$ computation. Hence, the memory complexity is much lower.

Kernels. Similar to low-rank approaches, kernels use mathematical methods to lower the computational costs of the self-attention mechanism.

Recurrence. In this case, several blocks are connected through recurrence. This method is a natural extension to block-wise patterns.

Downsampling. With this technique, the input sequence is downsampled to a lower resolution. Therefore, the computational costs are reduced.

Sparse Models. These models do not have sparse attention, but introduce sparsity to their parameter activation. Hence viewer operations are needed.

2.4.3. Attention Patterns

As discussed in Section 2.4.2, a strategy involves utilizing various attention patterns in combination. Prominent examples include Longformer (Beltagy et al., 2020) and BiG Bird (Zaheer et al., 2020). These methods substitute full n^2 attention with more efficient versions to incorporate sparsity into the model. Specifically, they implement the following attention mechanism:

- **Random attention**: Each token may focus on a random subset of other tokens.
- **Window attention**: Each token focuses on a predetermined group of adjacent tokens.
- **Global attention**: Specific important tokens have the ability to focus on every other token and receive attention in return.

These strategies enhance the efficiency of computational processes. Nonetheless, there exists a compromise between the efficiency of computation and the richness of the attention pattern.

2.4.4. Fusion-in-Decoder Approach

Usually, efficient transformer models include several optimisation methods. These customised model architectures require expensive training from

scratch. Recently, approaches have been proposed to reuse short-text pre-trained language models. The work of Ivgi et al., 2023 suggested SLED (SLiding Encoder and Decoder), which uses a sliding window to preprocess the input. The input is divided into multiple overlapping chunks of the same size. Each of the chunks is encoded by the language model. The pre-trained decoder fuses the encoded chunks to generate a single representation. The whole process is similar to a fusion-in-decoder (Izacard and Grave, 2021) approach. Ivgi et al. (2023) experiments show that this technique competes with specialised architectures and models with a much larger number of parameters. Hence, this is an efficient low-resource approach leveraging pre-trained language models.

2.4.5. Additional Efficient Approaches

According to Treviso et al. (2023), not only the model design, as explained in Section 2.4.2 and Section 2.4.4 is an aspect that makes the process more efficient. Models can be influenced by data, pretraining, fine-tuning, inference, and compression, as well as hardware utilisation, to efficiently achieve high performance with lower resources. However, only the model design, special inference and compression techniques, or efficient hardware use can reduce the memory and computation costs during training and inference.

2.5. Controllable Summarization

Controllable summarization focusses on user-specific attributes and incorporates them into the final summary (Y. Zhang et al., 2023). The output can be customised according to the user's needs. There are different types of attributes, such as length, entity, aspect, content, style, granularity, or abstractiveness. Examples for research works on controllable summarization are CTRLsum (He et al., 2022), FacetSum (Meng et al., 2021) or MacSum (Y. Zhang et al., 2023), where the first two also include experiments of scientific articles. MacSum has a focus on the news and meeting domain. A related task is query-focused summarization, where a summary answer is formulated based on a user question. However, the response is not explicitly generated to a specific output style.

2.5.1. Control Mechanism

Prompt learning is a commonly used mechanism to give language models instructions and guidance. In general, there are two types of control mechanism namely, prompt-tuning and prefix-tuning (Y. Zhang et al., 2023).

Prompt-tuning. The instructions are given in the form of words or phrases as part of the input. The model learns to adapt the context based on these prefix words. Sometimes these prompt words have to be carefully selected to obtain the needed information.

Prefix-tuning. Hereby, a vector is appended to each layer of the language model. Only the added vectors are trainable, the rest of the network parameters are frozen. Hence, only a small part of the model is trained and fine-tuned for a specific task.

2.5.2. Attribute Types

As already mentioned, there are several attribute types. A selection of them is described in more detail below.

Aspect Control. Thereby, the focus lays on a certain aspect (Meng et al., 2021). For example, in the case of Facetsum (Meng et al., 2021), the aspects are purpose, method, findings, and value. Each of the aspect summaries describes the input article from a different perspective or in a particular context.

Entity Control. The aim of entity control is to generate summaries that pay special attention to the given input entities (He et al., 2022). Usually, entities are provided in the form of keywords or are included in short prompts.

Length Control. Controlling the length gives the user the possibility to generate summaries of different sizes (He et al., 2022; Y. Zhang et al., 2023). To prepare the training data, the reference summaries are assigned to different buckets. Each bucket is described through a length parameter. During training, the parameter is provided in addition to the input. The user can then control the summary length during inference by stating the required length parameter.

2.6. Evaluation

Evaluation metrics measure how effective a model is performing. Therefore, a well-suited measurement is crucial for efficiently training and testing models.

2.6.1. Performance Metrics

In general, performance metrics can be categorised into lexical and semantic matching metrics. In the following points selected metrics are explained. Table 2.1 shows the described metrics and some characteristics.

Lexical Matching Metrics

ROUGE. The most common evaluation metric is ROUGE (Lin, 2004), which is used for many NLP tasks, including machine translation. The score is calculated by automatically comparing the summary generated with a ground-truth reference. There exist multiple variants of ROUGE, the most widely used being ROUGE-N and ROUGE-L. ROUGE-N is a recall measurement between the n-grams of reference and the generated summary. Usually, ROUGE-1 and ROUGE-2 are the most common forms, representing unigrams and bigrams, respectively. ROUGE-L makes use of the longest common subsequences and counts the matches.

BLEU. Another lexical matching metric is BLEU (Papineni et al., 2002), which measures precision rather than recall. For that, the number of candidate words in the reference summary is divided by the total number of words in the summary generated.

Perplexity (Jelinek et al., 1977) assesses the quality of a language model. This metric calculates the average negative logarithmic probability of a sequence or word. A lower perplexity is more favourable as it indicates a higher grammatical quality of the summary by using less frequently appearing words.

Pyramid (Nenkova et al., 2007) is a quantitative measurement method for evaluating the selection of content for a generated summary. It consists of Summary Content Units (SCUs), each representing a different meaning. If multiple human-written summaries agree on certain content, the SCU is more important and, therefore, weighted higher. To evaluate the quality of the summary, the number and importance of the SCUs is calculated. The pyramid is strongly correlated with human assessment. However, human experts are needed to annotate and evaluate.

Responsiveness (Louis and Nenkova, 2013) measures the content and linguistic quality of a summary without comparing the candidate summary with a human-written gold reference. However, the overall score relies on human assessment in the form of summary ratings.

Semantic Matching Metrics

METEOR (Banerjee and Lavie, 2005) is an enhanced version of BLUE, which also takes synonym words into account by using knowledge graphs. Therefore, the metric is evaluated on the basis of the semantic meaning of words instead of the exact matching of sequences.

SUPERT (Gao et al., 2020) is an unsupervised metric for multi-document summarization. The algorithm extracts salient sentences using contextualised text encoders and builds a pseudo-reference summary. Finally, the semantic overlap between the pseudo-reference and the candidate summary is calculated.

	Lexical Matching Metrics					Semantic Matching Metrics				
Eval. Metric	ROUGE	BLUE	Perplexity	Pyramid	Responsive-ness	METEOR	SUPERT	BERTScore	MoverScore	Human Evaluation
Autom. eval.	yes	yes	yes	no	no	yes	yes	yes	yes	no
Gold summ.	yes	yes	no	yes	no	yes	no	yes	yes	-

Table 2.1.: Common evaluation metrics and their characteristics include automatic evaluation and the necessity of a gold summary.

BERTScore (J. Zhang et al., 2020) is based on pre-trained language models to evaluate semantic similarities between sentence embeddings. It addresses the problems of metrics based on n-grams, such as exact paraphrase matching, distance dependencies, or semantically critical word ordering.

MoverScore (W. Zhao et al., 2019), similar to BERTScore, is built on contextualised representations and semantic distance measurements. It captures not only the similarity between the reference and the generated summary but also the deviation from the reference.

Human evaluation is the most accurate measurement. However, it is resource-intensive and requires human annotations.

2.6.2. Faithfulness Metrics

Hallucinations and factual inconsistencies are common issues among transformer models (see Section 2.3.4). Several evaluation metrics were designed to measure the factuality of the models. They can be categorised, as described in Section 2.3.4, into entailment classification, question-answering (QA), and fact-based approaches.

Entailment Classification Approach

NLI-based metrics assume that a summary is faithful if it can be entailed by the gold summary (Ji et al., 2022). Otherwise, the output generated contains hallucinations. In this approach, smaller parts of the candidate summary are examined, for example, on the level of words, phrases, or sentences. For text entailment, natural language inference (NLI) approaches are applied. Units extracted from the generated summary are verified against the input document.

However, it was shown by Falke et al. (2019) that models trained on NLI datasets do not perform well in ABS tasks and therefore are unreliable. This is mainly due to the characteristics of NLI-datasets, as they use shorter premises than for documents used for ABS. Models trained on longer premises achieved better results. Laban et al. (2022) introduced an efficient NLI approach called SUMMAC that achieves very good results by simply processing sentences segments rather than entire documents.

A common semi-supervised classifier of factual consistency is FactCC (Kryscinski et al., 2020), which is based on BERT (Devlin et al., 2019). Semi-supervised metrics use models to predict whether the summaries contain hallucinatory content (Ji et al., 2022). The classifier was trained on a specific artificially created dataset. Positive labels are extracted or paraphrased sentences, and negative labels are created synthetically by corrupting sentences with false information. The FactCC model is trained simultaneously in three tasks. The first task is to analyse whether the synthetic sentences are still factual consistent after transformation. Next, the source text is searched for sections, called spans, that support the consistency prediction. Then, in case of inconsistency, spans from the summaries that indicate the contradiction are selected, revealing the inconsistency between the synthetic sentence and the source text. The resulting score is the number of faithful sentences divided by the total number of sentences.

Question-Answering Approach

QA-based metrics make the assumption that the summary and the source input provide similar responses to the same QA tasks (Ji et al., 2022). QA-models are a suitable approach, as fact checking is a task of logical inference.

Usually, the models consist of two parts, a question generation (QG) and a QA unit (M. Li et al., 2022). Many works propose procedures that consist of three steps. First, a QG model formulates queries from the summary. Second, a QA model generates responses from the input text. Finally, the system compares the set of answers from the summaries and the input documents and computes a score. If the answers align, the text is factual consistent; otherwise, inconsistent. QA-based metrics were found to be more closely correlated with human evaluations reviewing faithfulness.

In recent years, several QA-based metrics have emerged. According to M. Li et al. (2022), **QAGS** (Wang et al., 2020) and **FEQA** (Durmus et al., 2020) were among the first metrics to cover this aspect. Both metrics are based on BERT (Devlin et al., 2019) and BART (Lewis et al., 2020) for their QA and QG modules. QAGS extracts n-grams, while FEQA concentrates on entity extraction. **QuestEval** (Scialom et al., 2021) extended the proposed framework of QAGS and FEQA and introduced an additional procedure. **QUALS** (Nan et al., 2021) further simplified the steps and came up with a single model (QAGen), which generates question-answer pairs directly from the source text. **UniEval** (Zhong et al., 2022) evaluates texts using boolean QA. The evaluator can assess the generated text on multiple dimensions, including coherence, consistency, fluency, and relevance. UniEval reaches a higher correlation with human judgements in text summarization than other evaluators.

Fact-based Approach

Intuitively, checking the fact overlap is the easiest way to guarantee faithfulness. There are several ways to represent facts, including entities, n-grams, or relation triples (M. Li et al., 2022). Either the entity is inconsistent or its relation.

IE-based systems apply IE methods for the extraction of tuples from relations that result in subjects-relation-object pairs (Ji et al., 2022). These tuples are extracted from both the source and the generated text. After that, it is determined how factual accurate the output of the model is. However, IE-based systems are not fully reliable. Some researchers (Nan et al., 2021) proposed a more robust alternative based on named entity recognition (NER). NER models only entail a set of named entities from the text. Both sets of the generated and the gold summary are compared. If the sets differ, hallucination is assumed.

EntityAlign (Nan et al., 2021) is an approach based on named entity recognition. The faithfulness score is calculated by dividing the number of entities in the candidate summary, which also occur in the source text by the total number of entities in the summary. **TripleAlign** (Goodrich et al., 2019), unlike EntityAlign, describes facts with entities and their relationship in the form of subject, relation, object pairs.

Other Approaches

In addition to the approaches mentioned above, methods based on other mechanisms are suggested. The most prominent is **BARTScore** (Yuan et al., 2021), which is based on the pre-trained language model BART (Lewis et al., 2020). It formulates the evaluation of the generated summary as a text generation problem. The final score is calculated by summarising the weighted logarithmic probabilities of the generated output given a specific input. Several aspects can be assessed, including faithfulness, but also precision and recall.

2.6.3. Text Understandability

Generally, multiple features characterise the ease of understandability of a text. Aspects such as readability, text coherence, factual consistency, fluency, or relevance are crucial for the reader to comprehend the key points of the text. There are several ways to automatically assess text quality, based either on linear functions or pre-trained language models (Feng, 2010; Zhong et al., 2022).

Traditional Readability Metrics

According to Feng (2010), traditional readability metrics usually use simple linear functions with a few lexical and syntactical text features to estimate a single readability score. For example, text characteristics such as sentence lengths, number of syllables, number of characters, or common and difficult words are taken into account. The most widely used metrics are Flesch Reading Ease (FRE) and the Flesch-Kincaid grade level formula (Flesch, 1979), Gunning FOG Index (Gunning, 1952), SMOG Index (McLaughlin, 1969), Automated Readability Index (Senter and Smith, 1967), Dale-Chall formula (Dale and Chall, 1949), Coleman Liau Index (Coleman and Liau, 1975) or, the Linsear Write Formular (Klare, 1974). Most formulas try to estimate the level of education given in US grade levels, which is necessary to understand the text.

The Python library *Textstat* library[1] can be used to compute the scores mentioned. A very common metric is the Flesch Reading Ease (FRE). The FRE index estimates the readability of texts by assigning a value between 0 and 100, where the first is the hardest, and the latter is the easiest reading score. Additionally, the library also provides a combined evaluator that uses eight metrics to assess the grade level. The combined evaluator applies a majority vote using eight metrics that include the equivalent grade of the FRE, the Flesch-Kincaid Grade (Flesch, 1979), the SMOG Index (McLaughlin, 1969), the Coleman Liau Index (Coleman and Liau, 1975), the Automated Readability Index (Senter and Smith, 1967), the Dale Chall Readability Score (Dale and Chall, 1949), the Linsear Write Formular (Klare, 1974), and the Gunning Fog Index (Gunning, 1952). The final grade is the most common value of all the tests mentioned.

Single and multi-dimensional evaluators

Here, one model estimates one or more dimensions with respect to text quality (Zhong et al., 2022). These dimensions are usually coherence, consistency, fluency, and relevance. The characteristics of each dimension are as follows:

- **Coherence**: Estimates how coherent the text body is.
- **Consistency**: Measures the factual alignment between the input document and the output summary.
- **Fluency**: It is a measurement for the quality of sentences.
- **Relevance**: Evaluates whether the summary contains important information.

Recently, Zhong et al. (2022) developed a model called UniEval to evaluate all four dimensions. In addition to estimating the factual consistency, as

[1] https://github.com/textstat/textstat

already described in Section 2.6.2, the model also assesses coherence, fluency, and relevance. The benefits of applying only one evaluator are that it is easier to use, joint training is possible, the model can be enhanced by external knowledge, and the model is extensible and transferable. As the dimensions are related, they can contribute to each other if they are trained together. The authors of UniEval state that the model has a higher correlation with human judgment than other existing evaluators.

2.7. Datasets

There are several publicly available datasets for the task of text summarization, each having its own characteristic features (Koh et al., 2022). Model properties such as model performance or summarization type have to be found to be influenced by the characteristic of the dataset. A course classification can be made by the length of the documents, that is, short- and long-document. A more fine-grained comparison is possible on the basis of the intrinsic characteristic.

2.7.1. Comparison of Datasets

According to the survey of Koh et al. (2022), there are differences between the datasets in several aspects, such as length, compression, abstractiveness, diversity, or uniformity.

Document Length

The survey of Koh et al. (2022) found that common long document datasets have between 1,600 and 9,400 tokens per document on average. In contrast, documents in corpora composed of short texts are much smaller, with around 400 to 800 tokens per document.

Compression Ratio

The compression ratio is defined by the word ratio between the article A and the summary S:

$$\text{Compression}(A, S) = |A|/|S| \tag{2.3}$$

Long documents have on average a higher compression ratios for tokens as well as sentences. The survey of Koh et al. (2022) hypothesises three scenarios, with the finding that the first two are more likely to be the reason for the higher compression. They speculate that the loss of information is greater, the distribution of content is sparser, and that there is a greater

proportion of redundant information in the document. Clearly, it is more difficult to capture the key narrative of longer documents than of shorter ones. Thus, the user's needs for generalised summarization are hard to satisfy. A more user-centred approach, such as controllable summarization, could be more efficient in covering the reader's requirements.

Abstractiveness and Diversity.

The abstractiveness of a summary can be measured by the coverage and density of the extractive fragments. Extractive coverage describes the percentual ratio of the tokens in the summary that are derived from the source document. Coverage is defined by the following formula:

$$\text{Coverage}(A, S) = \frac{1}{|S|} \sum_{f \in \mathcal{F}(A,S)} |f| \tag{2.4}$$

,where S is the summary and f are the extractive fragments of the summary derived from the original text. Therefore, a high score indicates that the summary is of extractive nature, and a low score an abstractive nature.

The density formula is similar to the coverage formula, just with the difference that the fragment length is squared:

$$\text{Density}(A, S) = \frac{1}{|S|} \sum_{f \in \mathcal{F}(A,S)} |f|^2 \tag{2.5}$$

The density measures how well word sequences from the summary can be described by sequences of extracted words. If a dataset has high coverage but low density, it means that individual words appear in the input text; however, subphrases are shorter and words are ordered differently in the summary. Therefore, the summary could still convey different ideas.

According to Koh et al. (2022), long document datasets have compared to short document datasets greater coverage. This might be due to the domain specificity and complexity of long-document datasets.

Uniformity

Koh et al. (2022) noticed from their evaluation of the ArXiv (Cohan et al., 2018) dataset that important information in scientific articles is evenly distributed throughout the document. In comparison, short documents usually contain salient content in the first 30% of the text. These characteristics can be measured by uniformity:

$$UNF\left(\text{unigram}_{pos}\right) = H_{\text{norm}}\left(\text{unigram}_{pos}\right) \tag{2.6}$$

	Short Document Datasets					Long Document Datasets							L./S.*
	CNN-DM	NWS	XSum	WikiHow	Reddit	ArXiv	PubMed	BigPatent	BillSum	GovReport	FacetSum	SciTLDR	
#Docs	278 K	955 K	203 K	231 K	120 K	215 K	133 K	1,340K	21.3 K	19.5 K	5.8 K	3.2 K	–
Tokens per Summ	55	31	24	70	23	242	208	117	243	607	195	21	6.9x
Tokens per Doc	774	767	438	501	444	6446	3143	3573	1686	9409	5698	5066	8.3x
Sents per Summ	3.8	1.5	1	5.3	1.4	6.3	7.1	3.6	7.1	21.4	8.5	1	3.7x
Sents per Doc	29	31	19	27	22	251	102	143	42	300	225	224	6.5x
Compress. $_{\text{token}}$	14.8	31.7	19.7	7.2	18.4	41.2	16.6	36.3	12.2	18.7	31.9	291.7	1.4x
Compress. $_{\text{sent}}$	8.3	22.4	18.9	3.3	14.5	44.3	15.6	58.7	9.7	18.1	28.4	219.9	2.2x
Coverage	0.890	0.855	0.675	0.610	0.728	0.920	0.893	0.861	0.913	0.942	0.937	0.920	1.2x
Density	3.6	9.8	1.1	1.1	1.4	3.7	5.6	2.1	6.6	7.7	6.2	3.7	1.5x
Redundancy	0.157	0.088	–	0.324	0.078	0.144	0.146	0.223	0.163	0.124	0.167	0	1.0x
Uniformity	0.856	0.781	0.841	0.813	0.777	0.894	0.896	0.922	0.903	0.932	0.942	0.897	1.2x

* Long vs. Short avg. ratio

Table 2.2.: Comparison of single-document datasets, adapted from Koh et al. (2022).

,where H_{norm} is the normalised entropy measured of the decile positions of salient unigrams in the source document. Salient unigrams are the top 20 keywords extracted [2] from the target summary.

Redundancy

The redundancy of a summary can be quantified by the sequence overlap of sentences (Koh et al., 2022). First, a distinct pair of sentences is created. Second, the ROUGE-L F1 score is calculated between two sentences. The final redundancy score is the average of all sentence pairs (see Formula 2.7).

$$\text{REDUNDANCY}(S) = \underset{(x_i, x_j) \in \mathrm{S} \times \mathrm{S}, x_i \neq x_j}{\text{average}} \text{ROUGE}\left(x_i, x_j\right) \tag{2.7}$$

Inference between Intrinsic Characteristics.

An analysis of the correlations between the metrics found that redundancy and coverage are negatively correlated (Koh et al., 2022). It is suggested that human writers compose more vorbose texts that include unnecessary information when requested to write abstractive summaries. Furthermore, the long-document datasets showed a negative interrelationship between coverage and compression. One hypothesis is that writers are forced to be concise and paraphrase more due to the length constraint of the summary.

2.7.2. Scientific Single-document Datasets

ArXiv (Cohan et al., 2018) and **PubMed** (Cohan et al., 2018) are the most widely used scientific datasets for summarization tasks. These datasets were collected from scientific repositories such as ArXiv[3] and PubMed[4]. The text of the article serves as an input, while the corresponding abstract represents the target summary.

[2] Keyword extraction tool such as NLTK-RAKE.

[3] https://arxiv.org/

[4] https://pubmed.ncbi.nlm.nih.gov/

Facetsum (Meng et al., 2021) introduced a facet-divided scientific dataset sourced from a scientific journal. The dataset, unlike ArXiv and PubMed (Cohan et al., 2018), contains four summary sections written by the authors, namely purpose, method, findings, and value. Each section covers a different aspect of the article. Thus, it is possible to implement an aspect-oriented summarization.

SciTDLR (Cachola et al., 2020) proposed an extreme multi-target summarization dataset aimed at highly compressing source articles into very condensed summaries. Each article in the dataset includes an author-written summary as well as multiple summaries provided by peer reviewers. This results in a multi-target dataset (available only for the test and validation sets) with significant variability in the target summaries, distinguishing it from datasets like ArXiv and PubMed (Cohan et al., 2018).

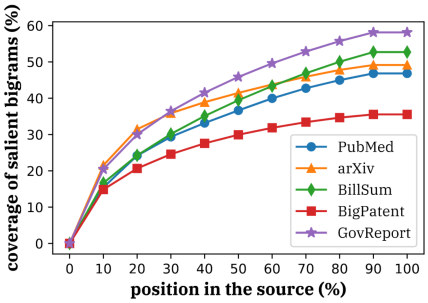
QASPER (Dasigi et al., 2021) is a question-answer (QA) dataset with a focus on scientific articles. Each question is written by a human and covers a specific information need. The questions were answered by several workers, including also evidence sentences from the source text.

Long-Document Datasets Comparison

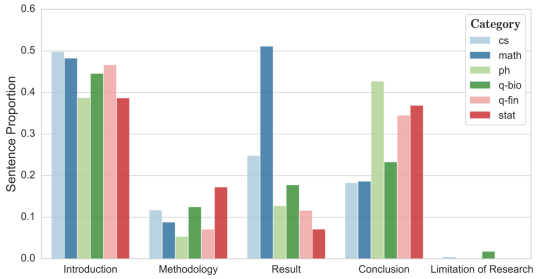
Compared to other long document datasets, ArXiv (Cohan et al., 2018) and BigPatent (E. Sharma et al., 2019) have shorter and more paraphrased summaries. GovReport (L. Huang et al., 2021) has the longest summaries with a length of around 600 tokens on average. Furthermore, the salient information in GovReport documents is scattered throughout the source text. For example, in Figure 2.10a it can be seen that in ArXiv or BigPatent, only 10% novel bigrams occur in the lower half of the document, compared to 18% in GovReport and BillSum (Kornilova and Eidelman, 2019). Hence, in the latter, the important content is more evenly distributed. In general, scientific datasets differ slightly from other long document datasets.

Dataset Noise

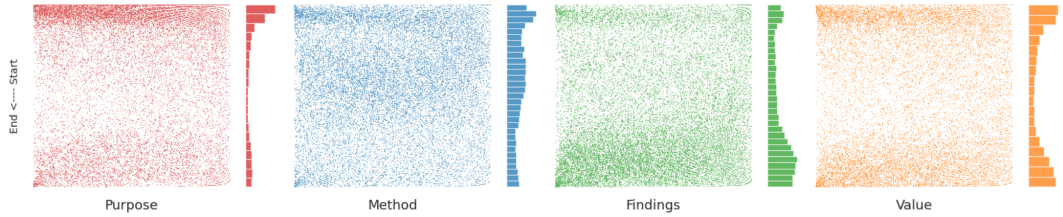
ArXiv and PubMed datasets were both published by Cohan et al. (2018). The survey of Koh et al. (2022) provides a fine-grained analysis of the ArXiv corpus. The ArXiv dataset was found to be a very noisy dataset with approximately 15% significant errors in the reference summaries of a randomly chosen subset. Most errors seem to occur due to an inaccurate data scraping process. The same inaccuracies probably also arise in the PubMed dataset, as the same pre-processing methods were used. In general, it was found that in many datasets at least a small percentage of the samples were erroneous.



a) Distribution of novel bigrams. (L. Huang et al., 2021)



b) Summary sentence distribution throughout different sections. (Koh et al., 2022)



c) Summary sentence distribution across different aspects. (Meng et al., 2021)

Figure 2.10.: These graphs show the bigram novelty and the summary sentence distribution over across different sections and aspects.

Content Coverage

The sentences in the reference summaries of the arXiv dataset originate from different sections of the source text. The survey of Koh et al. (2022) examined the introductions, methodologies, results, conclusions, and limitations of the research sections (see Figure 2.10b). For example, it was found that articles in the mathematics domain have a greater focus on the contribution, whereas documents with a physics topic concentrate more on conclusions. Figure 2.10c shows the distribution of summary sentences in scientific papers with an economic background, depending on the aspect. The summaries are primarily based on the beginning (introduction section), followed by the end (results and conclusion sections).

FacetSum (Meng et al., 2021) also found that, depending on the aspect, the sentence distribution varies (see Figure 2.10c). For example, sentences about the purpose are mostly in the beginning of a paper, whereas findings are usually at the end. The descriptions of the method are uniformly distributed, and the value of a paper is found equally at the beginning and the end of an article.

SAPGraph (Qi et al., 2022), like FacetSum, has encountered that introduction and conclusion are usually sufficient to cover the key points of an article. However, other sections are essential for higher performance in generalised summarization. Therefore, it is suggested to use the full articles as an input.

2.7.3. Scientific Multi-document Datasets

Like scientific single-document, multi-document datasets with academic background are comparably rare. Only in recent years have large-scale datasets from the scientific domain emerged.

Multi-XScience (Lu et al., 2020) is one of the most common datasets in the field of scientific multi-documentation. The dataset is sourced from the ArXiv website and is enriched with references using an academic graph. The resulting corpora consist of pairs of target summary and multiple reference abstracts. The target summary is the related work section of the query paper. Therefore, this dataset is suitable as a training set for the generation of related work.

SciSummNet (Yasunaga et al., 2019) constructed a corpus of 1,000 most cited papers and their citation sentences. The target summaries are written by domain experts by analysing the source abstract and their citation sentences.

Semantic Scholar Network (SSN) dataset (An et al., 2021) is a large-scale corpus that combines scientific articles and their citation relationships. Therefore, models can make use of the citation graph to extract additional information.

MS2 (DeYoung et al., 2021) is constructed from medical surveys and their respective reviews. The abstracts of the study and the background statements serve as input. The target summary is composed of the review's target statements. Therefore, this dataset does not have the typical section-divided characteristics of scientific articles. However, the studies have a scientific background.

PeerSum (M. Li et al., 2022) is an academic dataset scraped from two well-known international conferences. For each paper, they collected multiple reviews, comments, and responses. The meta-review serves as the ground truth target summary. The summary is highly abstractive and, therefore, more challenging to generate.

2.7.4. Benchmark for Long Text

Recently, Shaham et al. (2022) introduced a benchmark called SCROLLS (Standardized CompaRison Over Long Language Sequences) for NLP tasks with a focus on long text. The benchmark contains multiple tasks such as summarization, question answering, and natural inference tasks, covering various domains. Shaham et al. (2022) selected datasets with specific characteristics, such as the ability to contextualise and abstract information over a long range of text. These long-sequence datasets are particularly challenging for transformer models with a self-attention mechanism due to their high computational costs. Lately, adapted architectures and techniques were proposed to overcome this problem (Tay et al., 2022). SCROLLS offers

an evaluation suite to make cross-model comparisons and to highlight the strengths and weaknesses of models with specific tasks. The performance evaluation can be seen in Table 2.3. The leaderboard is provided on the official website [5].

Model Name	Reference	#Params	Input Length	Score (Average)
CoLT5 XL	Ainslie et al. (2023), Google	5.3B	16K	43.51
LongT5 XL	Guo et al. (2022), Google	3B	16K	42.53
CoLT5 Large	Ainslie et al. (2023), Google	1.46B	16K	41.04
LongT5 Large	Guo et al. (2022), Google	770M	16K	41.03
BART-LS	Xiong et al. (2023), Meta AI	460M	16K	39.76
LongT5 Base	Guo et al. (2022), Google	220M	16K	38.6
BART-large SLED	Ivgi et al. (2023)	406M	16K	37.99
UL2	Tay et al. (2023), Google	20B	2K	37.87
CoLT5 Base	Ainslie et al. (2023), Google	433M	16K	37.64
BART-base SLED	Ivgi et al., 2023	139M	16K	35.4
LED Base	Beltagy et al. (2020), Al2	162M	16K	29.16
BART Base	Xiong et al. (2023), Meta AI	139M	1K	29.01
LED Base	Beltagy et al. (2020), Al2	162M	4K	28.30
BART Base	Xiong et al. (2023), Meta AI	139M	512	27.58
LED Base	Beltagy et al. (2020), Al2	162M	1K	27.06
BART Base	Xiong et al. (2023), Meta AI	139M	256	26.35
Naive	Shaham et al. (2022)	-	-	19.35

Table 2.3.: SCROLLS performance evaluation with multiple models. The models vary in the number of parameters and input size. The score represents the average across multiple long text tasks.

2.8. Content Extraction for Scientific Documents

According to Z. Shen et al. (2022) scientific documents are usually published in Portable Document Format (PDF). There are several tools for extracting text-based content from PDFs. All extraction tools apply machine learning models to identify and categorise text sequences in the document. The classified sequences are then presented in a structured manner. Scientific publications usually contain a title, keywords, abstract, and sections with their respective paragraphs and headlines, as well as references (see Section 2.1.7). Additionally, meta-information such as the digital object identifier (DOI) can also be extracted. The tools are often based on other PDF parsing engines. A selection of recent tools is described in the following points.

[5]https://www.scrolls-benchmark.com/leaderboard

- **ScienceParse.** (Ammar et al., 2018) This tool parses scientific PDF papers and outputs them in structured form. This tool is mainly programmed in Java and Scala. The project is openly available[6].
- **GROBID.** ("GROBID," 2008–2023) The abbreviation stands for GeneRation Of BIbliographic Data. The library provides several machine learning methods to extract, parse, and restructure text information from PDF files. It can provide a detailed fine-structured output of scientific articles. The project is open-source; a detailed description is available in their repository[7].
- **Corpuse Conversion Service.** (Staar et al., 2018) It is a scalable cloud-based platform to parse and annotate documents. The platform converts documents into a structured data representation. However, the tool is not openly accessible.

2.9. Unsupervised Keyphrase Extraction

According to the survey Song et al. (2023), keyphrase extraction (KE) is an essential task in NLP to identify important phrases in a document. Unsupervised KE models do not need additional training and can be applied directly. Usually, the KE process includes extracting candidate phrases and selecting keyphrases. Traditional and embedding-based models have two main steps in common: tokenising and tagging the document with part-of-speech (POS) tags and extracting keyphrases based on the tags by regular expression. In addition, pruning techniques are applied to obtain better candidates. Finally, the keyphrases are ranked according to their importance. The authors of the survey Song et al. (2023) divide the methods into traditional and embedding-based techniques.

2.9.1. Traditional Models

Traditional models can be mainly categorised into statistics-based, topic-based, and graph-based models (Song et al., 2023). The models use different features, such as word frequency, position, or topic information, to estimate importance. Examples of commonly used traditional models are RAKE (Rose et al., 2010), YAKE (Campos et al., 2020), or TF-IDF, which analyse local text features.

RAKE (Rose et al., 2010) stands for Rapid Automatic Keyword Extraction. It is an efficient keyword extraction tool that can be easily applied to documents from different domains. RAKE partitions documents into candidate

[6]https://github.com/allenai/science-parse
[7]https://github.com/kermitt2/grobid

keywords using stop words and phrase delimiters. Then the importance score is computed on the basis of the word co-occurrence.

YAKE (Campos et al., 2020) is a statistical feature-based system to extract keywords from single documents. The tool supports multilinguality and does not require dictionaries or thesaurus. YAKE outperforms RAKE and TF-IDF in many datasets.

TF-IDF (Salton and McGill, 1983). Each word's weight depends on its frequency within the document and its overall occurrence in the entire corpus. Words that appear relatively rarely in the corpus have higher weights, and words with high frequency within a document also gain increased weight.

2.9.2. Embedding-based Models

In contrast to traditional models, embedding-based models use high-level features such as syntactic and semantic information (Song et al., 2023). Pre-trained language models are applied to obtain phrase and document embeddings. These embeddings are then used to calculate the importance score. Recent research and tools, such as KeyBERT (Grootendorst, 2020), have a focus on embedding-based methods, as they have reached state-of-the-art results.

KeyBERT (Grootendorst, 2020) is a simple and minimal keyword extraction technique that is easy to use. The authors provided a Python library [8] to quickly set up the extraction script. This method is a BERT-based (Devlin et al., 2019) solution, and hence, the algorithm operates on word and document embeddings to find the most similar words that describe the entire input text best. The similarity is computed by applying the simple cosine similarity. The tool requires the user to define a predefined n-gram length. This raises two issues. First, the user needs to determine the optimal n-gram lengths through experiments. Second, the generated keyphrases are limited by this length factor and often do not sound grammatically correct. To overcome these problems, PatternRank (Schopf et al., 2022) suggested adding keyphrase vectorizers to the extraction process. First, the input text is tokenised. Second, word tokens are labelled with part-of-speech (PoS) tags. Vectorizers then extract candidate keyphrases according to specific PoS patterns. The authors of PatternRank developed the *KeyphraseVectorizers* [9] package, which can be easily used in combination with the KeyBERT implementation. The extracted candidate keyphrases are the input for KeyBERT, which computes the phrase embeddings and similarities. The keyphrases are rankend in descending order, depending on their similarity to the input document. Only the top-N keyphrases are chosen as representative candidates for the input text. Figure 2.11 shows the entire extraction process.

[8] https://maartengr.github.io/KeyBERT/index.html
[9] https://pypi.org/project/keyphrase-vectorizers/

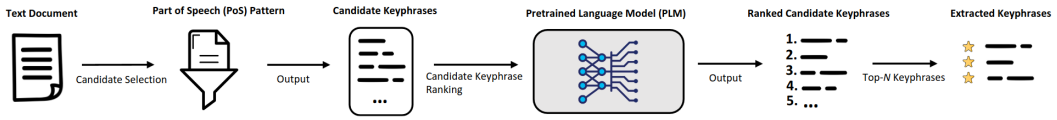


Figure 2.11.: PatternRank approach for unsupervised keyphrase extraction. (Schopf et al., 2022)

2.10. Summary

In conclusion, summarizing long documents effectively presents significant challenges. It requires consideration of numerous factors, ranging from the features of the text to the architecture of the models. Presently, transformer models achieve top-tier performance, but they are not without limitations, including substantial computational demands and problems with generating inaccurate content. However, transformer models enable the integration of controllability features, thus enhancing the interactivity of the summarization process.

The efficient transformer approaches address the issue of high computational demands. Various models are presented that include attention mechanisms or fusion-in-decoder strategies. Nonetheless, these methods usually involve compromises between computational efficiency and the richness of information.

Performance evaluation metrics are vital for assessing a model's effectiveness. Thus, selecting a suitable metric is critical for efficient model training and evaluation. Furthermore, faithfulness metrics are important for measuring text inaccuracies. Readability metrics also play a role in evaluating text clarity, including factors such as coherence and fluency.

Furthermore, training models effectively require high-quality datasets. Numerous extensive single- and multi-document datasets are publicly accessible, spanning various domains and topics. However, only a select number incorporate features such as controllability attributes or provide multi-targeting.

To produce valuable data from input documents, necessary pre-processing steps such as content extraction and unsupervised keyphrase extraction are required. In this process, it is essential to consider the structure of scientific documents. Numerous open-source tools are available to carry out these processes effectively.

Finally, several steps are needed to efficiently summarize long scientific texts, from datasets, pre-processing, model selection, and post-processing. A well-designed pipeline guarantees a good result and user satisfaction.

3. Experimental Setup

This chapter provides an overview of the overall idea, concept, and model architecture on which the experiments were based. The idea is derived from the background and related work section, while the requirements are based on the project's conceptual framework. These requirements are listed and explained in detail. Finally, design decisions are discussed and illustrated through a conceptual architecture.

3.1. Idea and Requirements

The idea of this work is to implement a summarization model to efficiently process long-sequence text documents in the scientific field with a focus on traceability. The model should be able to capture scientific phrases and integrate specific user needs.

Several issues arise from scientific documents, such as document length or specific scientific vocabulary, as described in Section 2.7. Therefore, the system should meet several requirements, such as capturing relationships in long documents, efficient processing of long input documents, and contextualisation of scientific phrases.

Furthermore, there are only a few systems in the literature that integrate user needs for scientific summarization (see Section 2.5). Some kind of user participation is beneficial and can improve the result. Also, the user should know from where in the input text the summary sentences are derived from.

Therefore, requirements such as capturing long-range relationships, efficient processing of long input text, contextualising scientific vocabulary, user-centred design, and traceability of summary sentences are identified. A brief description of the requirements is given in the following subsections.

3.1.1. Long-range Relationships

The model should be able to capture long-range relationships that span several paragraphs in text documents. In scientific articles, unlike short text documents such as news articles, information is scattered over the whole text. Usually, for models processing short text, it is sufficient to capture information with low locality. On the contrary, models for long-text documents need to process information that spans several sentences

and paragraphs. These long-range dependencies are hard to grasp and need sophisticated approaches. Therefore, as discussed in Section 2.4 a long document transformer is required to generate abstractive summaries of long input text.

3.1.2. Efficient Processing of Long Text

Recently, approaches built on the transformer architecture have become state-of-the-art for text processing. However, these models also come with drawbacks. Due to the characteristics of self-attention, computational efforts increase quadratically with respect to the input size (see Section 2.4.1). Hence, processing long-text documents can become infeasible and time-consuming. Currently, research in the field of long-text processing has focused on resource-saving approaches. The researchers suggested efficient attention layers or windowed encoder-decoder models to reduce computational costs (see Section 2.4). The goal is to construct architectures with the ability to efficiently process text information that are also suitable for low-resource environments. The low-resource and efficient transformer (see Section 2.4.2) or the fusion-in-decoder approach (see Section 2.4.4) is suitable in this case.

3.1.3. Scientific Vocabulary Contexualization

Models must be able to contextualise vocabulary from the scientific field. In general, scientific articles contain more complex words and terminology compared to short text from non-scientific domains, such as news articles or social media posts. To properly train such models, datasets from the scientific field are needed (see Section 2.7). For good performance, models are required to generate meaningful embeddings in order to efficiently extract information and relationships between phrases. To evaluate the model, common evaluation methods such as ROUGE and BERTScore (see Section 2.6) are used.

3.1.4. User-centred System

Most summarization systems deal with generic summarization. Therefore, user needs are neglected. Usually, the user wants to gain knowledge about a specific aspect of the text or define how detailed the information should be. Hence, controllable summarization is inevitable to fulfil the user's requirements. There exist several ways to express information needs, reaching from topics, keywords to questions. The model needs the ability to meaningfully process the query and produce an appropriate output that covers the user's

information need. Therefore, control tokens as described in Section 2.5.2 must be defined that the user can select for the summarization process.

Summarization Types

There are two types this work concentrates on, namely general summarization, aspect-oriented summarization. The summaries should be provided in an abstractive and extractive manner.

General summarization is the process of generating an excerpt of the input texts that covers the main points. The model extracts the core principles and ideas. It should also be able to highlight the main contributions of the paper. The summary should include more specific data, such as methods, datasets, and ideas of the authors.

In an **aspect-oriented summarization**, the summary emphasises a certain aspect of the text, for example, the purpose of the article or the findings of the authors.

Control Mechanism

To control the search result and satisfy the user needs, two control mechanisms are used, aspect and length tokens:

- **Aspect control tokens** are used to summarise a certain aspect of the input article.
- **Length control tokens** are used to define the summary length. Several text lengths are defined, reaching from brief to extra long.

3.1.5. Traceability of Summary Sentences

Faithfulness is an important characteristic for models in the scientific field. Therefore, the output must be traceable and the information origin must be verifiable. However, as described in Section 2.3.5 the interpretability of transformer networks is a difficult and partly unsolved problem. Therefore, a basic approach, such as the sentence similarity search, is often a sufficient alternative.

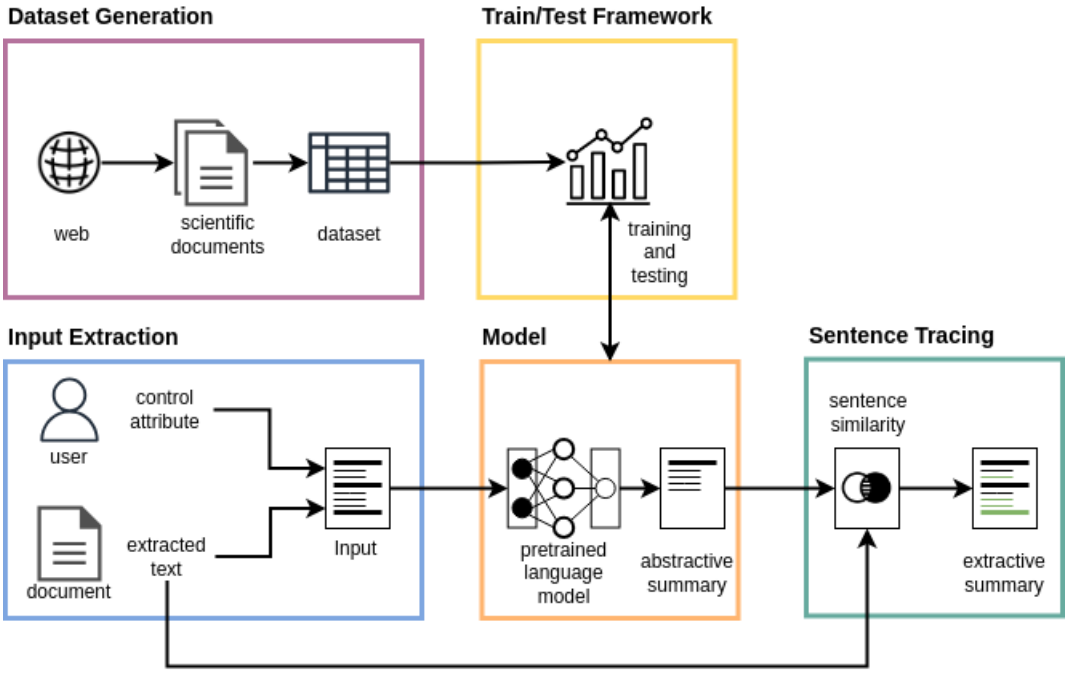


Figure 3.1.: Conceptual architecture of the system. The system includes a data generation process, a training and testing framework, input extraction, the model, and a sentence tracing mechanism.

3.2. Concept and Components

In this section, the conceptual idea and the general decision process for the components are described. In addition, the components are listed and explained. From the requirements, we can derive four main components for the conceptual architecture. The four basic components are dataset generation, input extraction, model framework, and sentence tracing. The conceptual architecture and its components are illustrated in Figure 3.1.

3.2.1. Dataset Generation

Specific training and testing data are needed for model training and evaluation. Therefore, scientific documents are scraped from journals and review websites. After training with this dataset, the model should be able to contextualize and process academic texts effectively.

3.2.2. Input Extraction

The model input is generated by extracting the text from the input document and combining it with the control attribute selected by the user. This makes

the system user-centered, requiring the model to respond to the user's needs.

3.2.3. Model

The model is designed to generate abstractive summaries and must efficiently process long input texts. Initially, the model is trained using the scraped scientific dataset. Once trained, it can produce summaries of unseen scientific articles.

3.2.4. Train and Test Framework

A training and testing framework is employed to evaluate the model. Commonly used metrics are applied in the evaluation process to compare its performance with other models.

3.2.5. Sentence Tracing

To trace the origin of the sentences in the abstractive summaries, a sentence similarity method is used. The most similar sentences then form the extractive summary. This allows the user to generate both an abstractive and an extractive summary. Additionally, this process enhances the trustability of our model.

3.3. Design Decisions

In this section, the design decisions for each component are described. Tools and libraries are specified and explained in more detail. Decisions are made regarding the model selection, text extraction tools, datasets for training, and similarity comparison techniques.

3.3.1. Model

One of the first steps was to do some research for suitable models that meet the requirements mentioned above. The idea was to choose state-of-the-art models based on the transformer architecture. During the research the choice fell on long-document transformer models (see Section 2.4) because of their efficient processing of long input text and their ability to elaborate user prompts into the search. The selection of the model is mainly based on the computation time and the memory consumption on the SCROLLS evaluation data (see Section 2.7.4).

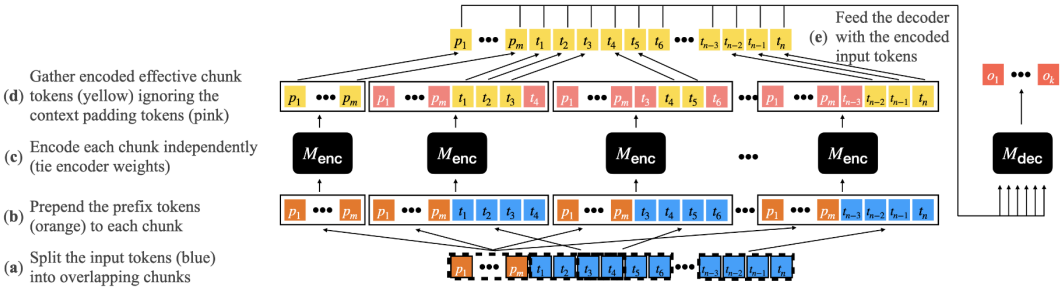


Figure 3.2.: Explanation of the SLED model. (Ivgi et al., 2023)

Three parameters are significant, namely, the input size, the number of parameters, and the average score. As can be seen in Table 2.3, the SLED (Ivgi et al., 2023) model shows a good performance trade-off. Compared to the low number of parameters, the model achieves a good average score. Other models do have a significantly higher number of parameters or perform poorer on average.

The SLED (Ivgi et al., 2023) model meets all the requirements mentioned above, such as long-range relationships, efficient processing, and user-centred design. The algorithm uses pre-trained language models to process long input text (long-range relationships). The input tokens are divided into multiple overlapping chunks. Therefore, a model with a smaller input size is sufficient (efficient processing). The control prefix token is then added to each chunk (user-centred). The encoder encodes each chunk separately. The encoded chunks are gathered and fed to the decoder, ignoring the context tokens. This system follows the fusion-in-decoder approach and is visually explained in Figure 3.2.

The authors of SLED also provide a training and testing framework. The framework is based on PyTorch[1], which is an optimised tensor library for deep learning using GPUs and CPUs. In addition, the authors published several pretrained SLED models on HuggingFace[2]. With the HuggingFace library trained models can be easily integrated into training, testing and inference pipelines.

3.3.2. Scientific Text Extraction

Based on the available tools described in Section 2.8 we selected GROBID (GeneRation Of BIbliographic Data) ("GROBID," 2008–2023), because it is open source and has suitable text extraction capabilities. For easier integration into the extraction pipeline, the Python library *SciPDF parser* [3] is used.

[1]https://pytorch.org/
[2]https://huggingface.co/
[3]https://github.com/titipata/scipdf_parser/tree/master

SciPDF is based on GROBID, which is, as already described in Section 2.8, a machine learning library for parsing and extracting information from raw documents, such as academic papers in PDF form. The tool has a special focus on technical and scientific papers. GROBID must be run locally as a service. SciPDF offers easy-to-use wrapper functions to communicate with the service. The PDF file must be provided as either a direct URL or a local file path. The extracted information can be provided in JSON format and includes the title, abstract, sections with their corresponding heading, and text, as well as reference and figure descriptions. The tool can also be used to extract figures from the PDF. However, this functionality was not needed because only the text information was of interest.

3.3.3. Training Datasets

To satisfy the requirement of capturing scientific vocabulary, scientific training datasets are selected. However, most of the existing datasets do not fully satisfy our needs. In Section 2.7.2 the main scientific single-document datasets are listed and briefly described. ArXiv and PubMed (Cohan et al., 2018) have a large number of samples. However, these two datasets provide only the abstracts of the papers as reference summaries. SciTLDR (Cachola et al., 2020) contributes human-written multi-target reference summaries, but these are condensed into a single sentence. Therefore, the summaries are very short and do not provide much information. QASPER (Dasigi et al., 2021) has a scientific background, but provides only a focus on question-answering. The questions are answered briefly in a couple of phrases, and additionally evidence is provided for the answers. However, the dataset is not suitable for the summarization task. FacetSum (Meng et al., 2021) is a faceted summary dataset. Summaries are written by humans and provide information on multiple aspects. For each document, only one summary exists; therefore, the dataset is not multi-targeted.

To conclude, the only scientific dataset that can be used for user-centred training is FacetSum (Meng et al., 2021). However, we also want a length-dependent multi-targeted abstractive summary dataset. Therefore, we decided to scrape two new datasets from an open peer review website. To generate and create the two new datasets and obtain the existing one, data was extracted from two websites, namely *Emerald*[4] and *OpenReview*[5]. The OpenReview dataset was solely obtained from the latter one.

[4]www.emerald.com
[5]www.openreview.net

Emerald Scientific Publisher

To access the dataset described by Facetsum (Meng et al., 2021) it is required to pull the specific documents from *Emerald*. *Emerald* is a digital publisher of research articles. It is mainly focused on business-related topics. What makes them different from other publishers is that articles are summarised according to specific aspects. These short summaries are provided by the authors and include views on the value, purpose, findings, and methods of the article. The authors of Facetsum do not provide the dataset directly as there is only limited access to the website. Only with an accepted university account is it possible to view the summaries and the associated scientific article. Therefore, the dataset is not publicly published.

OpenReview Peer Review

The second website accessed to generate the two dataset OpenReview Summary and OpenReview Contribution was *OpenReview*. The website follows open-review principles that include open peer review, publishing, access, discussion, and recommendations. Therefore, scientific articles are openly peer-reviewed and commented on by members of universities. This procedure guarantees transparency and quality of scientific work. Only a few journals make their review process publicly available and publish on *OpenReview*. However, some larger journals have published for several years and provide a large number of reviewed articles. In particular, four journals were used for the OpenReview Summary dataset and four for the OpenReview Contribution dataset. These include the Conference on Robot Learning (CoRL), International Conference on Learning Representations (ICLR), Medical Imaging with Deep Learning (MIDL), and Neural Information Processing Systems (NeurIPS) for the former and NeurIPs, as well as Uncertainty in Artificial Intelligence (UAI), Automated Machine Learning (AutoML), and Transactions on Machine Learning Research (TMLR) for the latter dataset (see Table 3.1). All journals have a computer science or engineering background. Depending on the journal, reviewers incorporate specific aspects in their reviews such as the advantage, disadvantage, or contribution of the scientific work. For each paper, several summaries written by experts are provided. Each summary differs in length and specifity. Hence, the website is a perfect source for length-dependent abstractive summarization datasets.

OpenReview Contribution	OpenReview Summary
Neural Information Processing Systems	
Uncertainty in Artificial Intelligence	International Conference on Learning Representations
Automated Machine Learning	Medical Imaging with Deep Learning
Transactions on Machine Learning Research	Conference on Robot Learning

Table 3.1.: Journals selected for inclusion in each dataset

Tools and Libraries

Facetsum (Meng et al., 2021) provides a crawler script for the website *Emerald* in their repository[6]. To access the *Emerald* website, one must first register on the website. After logging in, the user has access to the full paper and its summary. The session information can then be copied and used for the crawler script. The script is based on the HTTP library *Requests* and the *BeautifulSoup* HTTP parser. The script was adapted to make the scraping process more efficient and the dataset generation easier. The list of the required scientific articles is provided in the Facetsum repository.

The data scraping scripts for the newly generated summary datasets are based on the *OpenReview Python client*[7] with which it is possible to interact with the OpenReview REST API. The API is maintained and created by the official OpenReview team. With the client, it is possible to request various information about the journals, reviews, and comments.

3.3.4. Sentence Tracing and Semantic Search

To find the original sentences in the input document semantic search methods are used. In particular, SBERT (Reimers and Gurevych, 2019) is applied to create sentence embeddings. Sentences from the input text with the highest cosine similarity to sentences of the abstractive summary are chosen as representatives. The selected sentences then create the extractive summary.

To minimise the computation of the semantic search, candidate sentences from the input text are pre-selected. The pre-selection is based on the keywords and entities contained in the abstractive summary. For the extraction process of keywords and entities, KeyBERT (Grootendorst, 2020) was applied. As described in Section 2.9, KeyBERT is a simple and easy-to-use embedding-based keyphrase extraction tool.

Instead of SBERT (Reimers and Gurevych, 2019), as in PatternRank (Schopf et al., 2022), SciBERT (Beltagy et al., 2019) and SciDeBERTa (Jeong and Kim, 2022) were used in this work to create semantic vector representations of keyphrases. The advantage of the latter is the learnt intrinsic scientific vocabulary of the pre-trained model. SciBERT is trained on scientific papers from the SemanticScholar [8] corpus. Similarly to SciBERT, SciDeBERTa was trained on data in the science technology domain. Therefore, the models have better capabilities to embed certain scientific words and phrases.

[6]https://github.com/hfthair/emerald_crawler
[7]https://github.com/openreview/openreview-py
[8]https://www.semanticscholar.org/

3.4. Summary

In summary, we have five components in our conceptual architecture, namely, dataset generation, input extraction, model, training and testing framework, and sentence tracing. This work focuses on evaluating the benefits of efficient long-document transformers and the similarity search approach in comparison to other extractive and abstractive methods. Therefore, we chose SLED (Ivgi et al., 2023) as our experimental model due to the low number of parameters of the backbone model, the maximum input size and the computational efficiency compared to other alternative models. The authors also provide a framework for testing and training the model effectively. To extract text from scientific work, we select the GROBID ("GROBID," 2008–2023) tool in combination with the SciPDF Python wrapper. As this tool has sufficient text extraction capabilities and is open source. As datasets, we use the existing scientific corpus FacetSum (Meng et al., 2021) and scrape our own dataset from OpenReview. We call the newly generated datasets *OpenReview Summary* and *OpenReview Contribution*. These datasets allow for controllable summarization by attributes, such as text length for OpenReview. The final component is the tracing of sentences and the generation of extractive summaries. To find similar sentences, embedding-based sentence semantic search techniques using SentBERT (Reimers and Gurevych, 2019) are applied. Candidate sentences are pre-selected by their keyphrases generated by the KeyBERT (Grootendorst, 2020) extraction tool. Finally, the system can generate abstractive and extractive summaries based on attributes specialised on scientific work and is able to trace the origin of summary sentences in the source document.

4. Method

In this chapter, the methods used are explained, the system architecture, and the progress of the training. Additionally, we describe the structure of the generated dataset and the similarity search technique.

4.1. Architecture of the System

The architectural design is based on the design decisions of Section 3.3. The system is composed of three main components: preprocessing, inference, and post-processing. The architectural overview can be seen in Figure 4.2.

4.1.1. Preprocessing

In the pre-processing phase, the text of the input document is extracted and combined with the control token of the user. SciPDF is used to communicate with the GROBID ("GROBID," 2008–2023) service via an API. The tool extracts headlines and paragraphs of the input text.

The user can select either an aspect token or a length token. With the former, an aspect-oriented summary is generated, and with the latter, a length-controlled general summary is created.

4.1.2. Inference

The trained SLED (Ivgi et al., 2023) model is used to infer the abstractive summary. The summary depends on the provided control token. The inference script is based on PyTorch[1] and the HuggingFace[2] library. Before providing the input to the model, the input text has to be tokenised and prepared.

4.1.3. Postprocessing

In the last phase, an extractive summary is generated on the basis of the abstractive summary. First, the keyphrases and entities are extracted from the abstractive summary with the KeyBERT (Grootendorst, 2020) keyphrase

[1]https://pytorch.org/
[2]https://huggingface.co/

Figure 4.1.: Overview of scraping pipeline. Information is extracted from DOM objects and saved in CSV files.

extraction tool. Embeddings are generated using SciDeBERTa (Jeong and Kim, 2022) for better capture of the scientific term. Second, based on the keyphrases, candidate sentences are selected from the input text. Lastly, with the help of the sentence similarity search based on SentBERT (Reimers and Gurevych, 2019) the most similar candidate sentences are chosen. These sentences build the extractive summary.

4.1.4. Data Interface

A summary can be generated for every scientific article the user provides. However, for training and testing, the datasets were scraped from websites. The documents are scraped with the help of HTTP libraries. The necessary information is extracted from the DOM objects and saved in CSV documents. The last step is to generate the entire dataset according to the required format. The scraping process is illustrated in Figure 4.1. The pipeline consists of the following steps:

- **HTTP requests**: Sending requests to get the necessary information.
- **Scraping**: Extraction of required data from websites.
- **Dataset generation**: Building the dataset for training and testing.

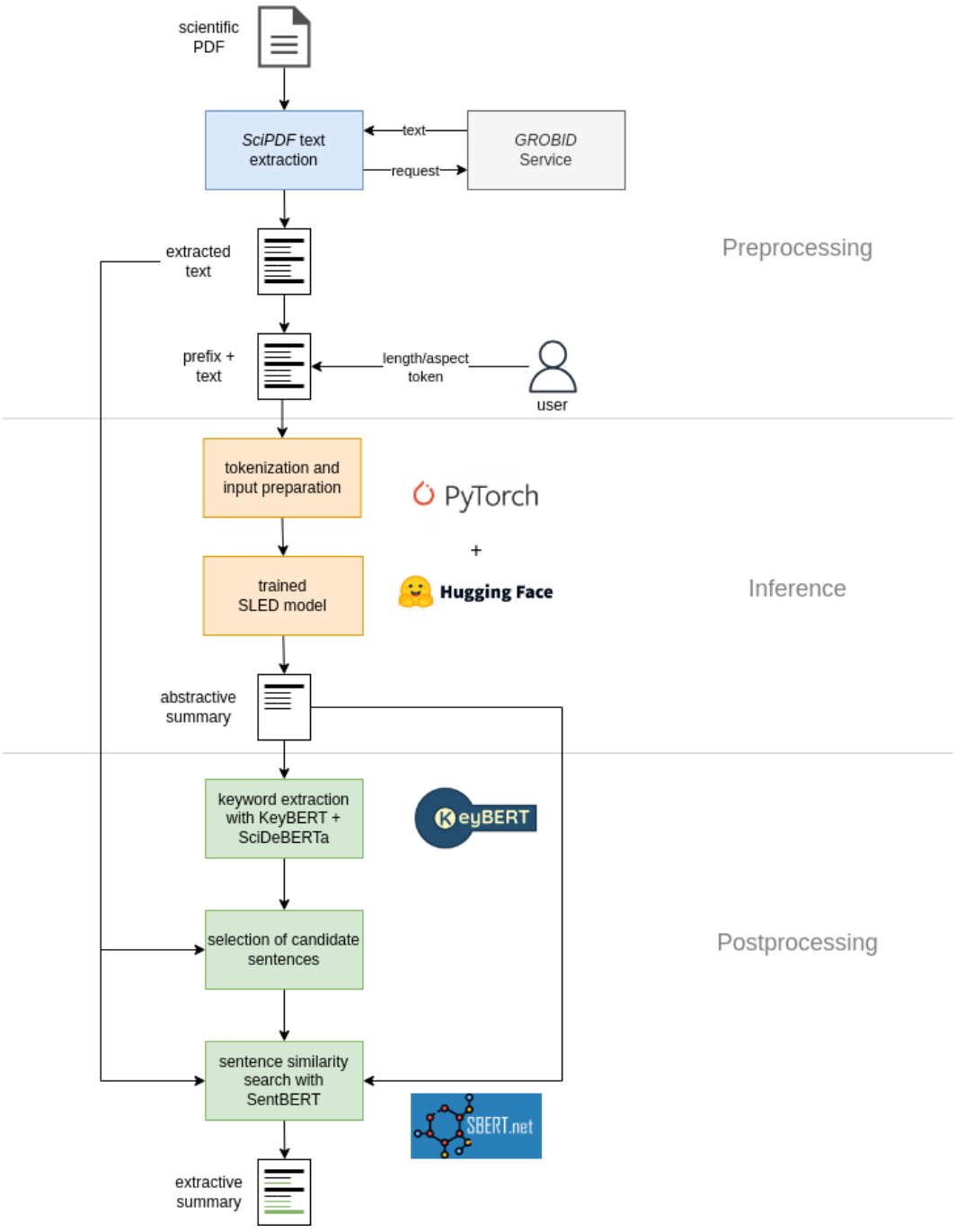


Figure 4.2.: Overview of the system architecture. The system consists of three steps, namely preprocessing, inference and postprocessing. First, the text is extracted from the scientific text and combined with a prefix. Second, the model generates an abstractive summary. Lastly, the sentences in the original text are traced and an extractive summary is generated.

4.2. Dataset Generation

To utilize the framework, the datasets were created following the format described by the authors of SLED (Ivgi et al., 2023). There are three datasets used for training and testing, namely FacetSum (see Section 4.2.2) and OpenReview (see Section 4.2.3), which is divided into OpenReview Summary and OpenReview Contribution.

4.2.1. Dataset Format

Scraped data were converted from JSON to CSV format. The datasets are subsequently stored in a single CSV file, which contain five fields: id, pid, input, input prefix, and output. The following information is provided in the above-mentioned fields:

- **id**: This represents the unique identifier for the data record.
- **pid**: When multiple entries share identical input but have different input prefixes, this field helps in differentiating these entries.
- **input**: This is the content contained within the input document.
- **input prefix**: This field specifies the prefix for a particular task. For summarization tasks, this could include keywords, aspects, or guiding questions.
- **output**: The output represents the intended outcome of the summarization process, providing a succinct version of the input text that considers the specified input prefix.

The input prefix differs depending on the dataset. As seen in Figure 4.2, the user can make a selection between a length token and an aspect token. The OpenReview dataset defines the former, while the FacetSum dataset defines the latter.

4.2.2. Facetsum Dataset Generation

The crawler script (described in subsubsection 3.3.3) provided by the authors was modified to download the FacetSum (Meng et al., 2021) dataset. This script was enhanced with multiprocessing features to speed up the download. The provided list with links served to pinpoint the required files. The input prefix for the FacetSum dataset is limited to purpose, method, findings, and value, following the definitions of the authors.

4.2.3. OpenReview Dataset Generation

Furthermore, another script (described in subsubsection 3.3.3) was developed to gather and organize the data for the OpenReview datasets. Based

on comments from each article, two distinct categories were identified: *"Summarization"* and *"Summary of Contribution"*. Consequently, two datasets were created, named *OpenReview Summary* and *OpenReview Contribution*. The prefixes were defined according to the desired output text lengths. The dataset was segmented into seven relatively evenly distributed bins, with minor modifications to enhance the balance of the labels. The categories were defined by the following length ranges: *"brief"* - $[0, 40)$, *"very short"* - $[40, 55)$, *"short"* - $[55, 70)$, *"middle"* - $[70, 90)$, *"long"* - $[90, 125)$, *"very long"* - $[125, 200)$, and *"extra long"* - $[200, \infty)$. The majority of the summaries are categorized as "short" and "middle".

4.3. Model Framework

The authors of SLED (Ivgi et al., 2023) provide a framework for training and testing. Their models are trained and tested on the SCROLLS (Shaham et al., 2022) datasets. The models used in their experiments as well as the datasets are published on Hugging Face[3] and the Hugging Face dataset hub[4]. Custom datasets can be used as well for training and testing. To load the own dataset locally, the framework was slightly adapted.

For this work, we train our own model. We use the base model of BART (Lewis et al., 2020) *bart-base-sled*[5] as it requires lower computational costs and has good accuracy. An alternative would be T5-based (Raffel et al., 2020) models, however, these models have a higher number of parameters and show lower performance.

The framework is implemented with PyTorch[6]. SLED (Ivgi et al., 2023) is fully compatible with the HuggingFace library and can be easily loaded using its methods. The installation and usage guide for the framework can be seen in their repository[7]. For the model settings, we refer to the original values of their BART base model, as it reaches the best performance. The authors of SLED use a context size of 256 and a window fraction of 0.5. The context size describes the size of the sliding window, and the window fraction refers to the degree of overlap. Additionally, for every chunk, the prefix is encoded and prepended. A technical description of the model framework is illustrated in Figure 3.2.

[3]https://huggingface.co/tau
[4]https://huggingface.co/datasets/tau/sled
[5]https://huggingface.co/facebook/bart-base
[6]https://pytorch.org/
[7]https://github.com/Mivg/SLED

Dataset	Task	max input size	max output size	learning rate	batch size
FacetSum	Aspect-oriented summary	12,000	200	1e-4	1
OpenReview Contribution	Length-dependend summary	12,000	512	1e-4	1
OpenReview Summary	Length-dependend summary	12,000	512	1e-4	1

Table 4.1.: Training settings for each task

4.3.1. Training Settings

For training, we adapt the data settings for the GovReport summarization dataset (L. Huang et al., 2021) provided by the authors of SLED (Ivgi et al., 2023). Adjustments were primarily based on the average and maximum input and output sizes of each data set. Another limiting factor is the GPU available. For this work, the model runs on a Nvidia GPU 3070 with 8 GB of GDDR6-RAM and 256 bit of memory bandwidth. Due to the restricted RAM capacity, only a reduced input and batch size can be used. However, the chosen input values cover the main proportion of document lengths. The maximum length of the prefix was 8 tokens in all cases. Furthermore, the configurations in Table 4.1 were specified for each task.

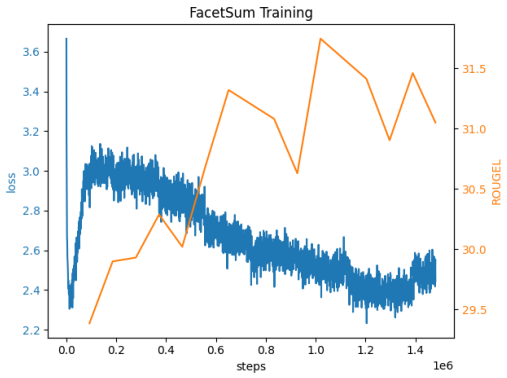
For the learning rate, we refer to the training arguments used by the authors of SLED (Ivgi et al., 2023). In their work, they applied a linear learning rate scheduler and an AdamW optimiser with $\epsilon = 1e-6$, $\beta 1 = 0.9$, and $\beta 2 = 0.98$. Furthermore, they use a warm-up ratio of 10% with a weight decay of 0.001. Due to the memory limitation of the GPU, only a reduced batch size of 1 was possible for training and testing. We adjusted the learning rate accordingly to $1e-4$. In comparison, the authors of SLED choose a learning rate of $\{2e-5, 5e-5, 1e-4\}$ with an effective batch size of $\{8, 16, 32\}$. In total, they train 9 models. However, they do not state which model setting performed best. To measure the performance of the models, ROUGE (Lin, 2004) and BERTScore (J. Zhang et al., 2020) was selected as the evaluation metric.

4.3.2. Training Progress

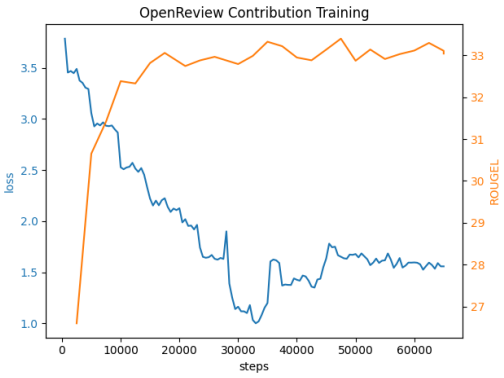
In Figure 4.3 the training progress of each model can be seen. The two newly generated datasets were divided into a 80% training, a 10% testing, and a 10% validation set. For FacetSum, the dataset was split according to the authors' suggestions. Due to the small batch size of 1 (see Table 4.1), a large number of training steps were necessary for a good generalisation.

Aspect-oriented training. The aspect-oriented model was trained for 4 epochs with approximately 1.5 million steps. The loss first increases and then slowly converges (see Figure 4.3a). However, after around 1.2 million steps, the loss increases, indicating that the models start to overfit. For evaluation, the best-performing model was used.

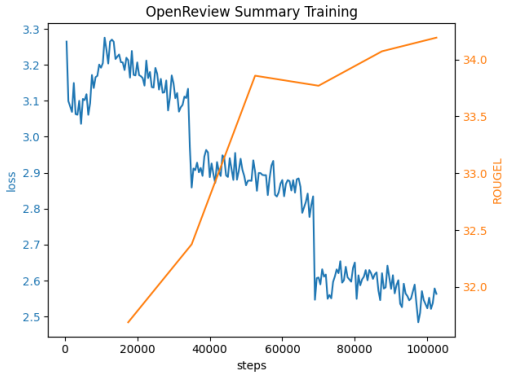
Length-dependend training. First we trained a model with the smaller



a) Aspect-oriented training with FacetSum dataset.



b) Training with OpenReview Contribution dataset.



c) Training with OpenReview Summary dataset.

Figure 4.3.: Training progress for each model.

OpenReview Contribution dataset for 14 epochs with around 70,000 steps in total. In Figure 4.3b it can be seen that the loss during training is steadily decreasing from the beginning. Accuracy increases and plateaus at a ROUGEL value of 33. Then we fine-tuned the larger OpenReview Summary dataset for 3 epochs with 100,000 steps. The loss and the respective ROUGEL curve can be seen in Figure 4.3c.

We also tracked how fast the model is learning the length token. First, the model generates only short and very short summaries. After some steps, the model reacts to the input token. In the end, the model follows the summary lengths fairly well. Progress in learning can be seen in Figure A.2.

4.4. Summary

For training and testing of the model, the framework by SLED (Ivgi et al., 2023) was used. This framework is built on PyTorch and Hugging Face, providing a robust foundation for developing state-of-the-art machine learning models.

The whole system architecture consists of preprocessing, inference, and postprocessing steps. First, the text of the scientific article is extracted using GROBID ("GROBID," 2008–2023), an efficient tool for parsing and extracting information from scholarly documents. Second, an abstractive summary is generated by the SLED model, which leverages advanced deep learning techniques to produce coherent and concise summaries. Finally, an extractive summary is created based on keywords extracted with KeyBERT (Grootendorst, 2020), a tool for keyword extraction using BERT embeddings. Additionally, a semantic search based on the abstractive summary sentences is conducted using SentBERT, a sentence embedding model, referred to as Sim. Search during this work.

The dataset format for FacetSum (Meng et al., 2021) and OpenReview was established following the guidelines provided by the authors of SLED (Ivgi et al., 2023). To gather the required data from the respective websites, two scripts were developed and customized to suit the specific needs of the project. Furthermore, for the OpenReview datasets, the summary lengths were divided into seven nearly equal categories to define the prefixes, ensuring a comprehensive representation of summary lengths across the dataset.

A total of three models were trained using three different datasets. Due to hardware limitations, the training time and parameters, such as batch size, were constrained. Despite these limitations, the models successfully converged and demonstrated promising results on the respective validation sets. The performance of these models indicates the potential for further refinement and optimization with additional computational resources and fine-tuning of hyperparameters.

5. Evaluation

This chapter explains the evaluation of the dataset and the model trained. First, the generated corpora were compared with other existing single long-document datasets (see Section5.3). Second, the trained models were evaluated for their performance and text quality, the outcome is described in Section 5.4 and Section 5.5 respectively.

5.1. Research Questions

The main focus of this work is exploring the characteristics of large-language models, especially efficient long-document transformers based on the fusion-in-decoder approach. Different extractive and abstractive methods are tested and evaluated for performance and output quality. The size of the model and the associated computational effort are of great interest. Based on this motivation, the following research questions were formulated.

- **[R1] Research Question 1:** *"Is there a benefit in using efficient long-document models over traditional small LLMs for the task of summarising scientific articles?"*
- **[R2] Research Question 2:** *"Can a similarity search approach that is based on abstractive summaries find and cover information more effectively than traditional extractive methods?"*
- **[R3] Research Question 3:** *"What are the qualitative differences between small LLMs and sophisticated models such as GPT?"*

5.2. Study Setup

In the following subsections, the setup is given for the Model 5.2.1, the Baselines 5.2.2, the Dataset 5.2.3, and the Metrics 5.2.4. In addition, the specific setting for each research question is described in Subsection 5.2.5.

5.2.1. Model

To answer the research questions that arise from motivation, an efficient long-document model called SLED (Ivgi et al., 2023) is trained. The model incorporates the fusion-in-decoder approach to efficiently encode long input and reduce computational effort. Based on the summaries that are generated by SLED similar sentences are searched in the original input text, and an extractive summary is generated. This approach is referred to as the "Sim. Search" throughout this work and is described in Section 3.3.4.

5.2.2. Baselines

For comparison with the abstractive SLED (Ivgi et al., 2023) model, a standard LLM called BART (Lewis et al., 2020) is chosen. BART also serves as an underlying base model for SLED. With the available hardware configuration, the SLED model can process up to 12,000 input tokens. In contrast, the standard BART model has only an input size of around 1,000 tokens. Furthermore, the standard TextRank (Mihalcea and Tarau, 2004) algorithm and the simple heuristic of using the abstracts of the paper are used as extractive baselines. For text quality comparison, a sophisticated state-of-the-art GPT model is applied.

5.2.3. Dataset

The models are trained and tested with high-quality scientific datasets. Therefore, two new corpora with unique characteristics from the scientific field are sourced, namely the OpenReview Contribution and OpenReview Summary (see Section 4.2), and an existing dataset called FacetSum (Meng et al., 2021) is selected. In both cases, human-written summaries serve as a ground truth.

5.2.4. Metrics

Performance is evaluated primarily using the lexical-based ROUGE (Lin, 2004) metric. For selected studies, we also include the semantic-based BERTScore (J. Zhang et al., 2020) metric. The properties of the performance metrics used are briefly explained in Table 5.1. Detailed measurement descriptions are available in Section 2.6.1.

Furthermore, the quality of the text (see Section 2.6.3) of the generated summaries is analysed on the basis of traditional readability metrics, as well as dimensions such as coherence, consistency, fluency, and relevance. For the latter, the multidimensional UniEval (Zhong et al., 2022) evaluator is applied, which is explained in more detail in Section 2.6.3.

Approach	Name	Abbreviation	Measurement
lexical-based	ROUGE1	R1	overlap of unigrams
	ROUGE2	R2	overlap of bigrams
	ROUGELsum	RLsum	longest common subsequences
semantic-based	BERTScore	-	precision, recall, F1 measures; similarity between candidate and reference summary

Table 5.1.: Metrics used to evaluate the performance of the models. The metrics can be divided in two categories: lexical- and semantic-based.

5.2.5. Question-specific Overview

In the following paragraphs, the specific setup for each research question is explained in more detail.

Setup [R1]. For the first research question, the advantages of using the efficient long-document model SLED (Ivgi et al., 2023) are examined. Additionally, the model is compared to the standard LLM BART (Lewis et al., 2020), which has, as stated before, a much smaller input size. The performance difference is explored and evaluated in different aspects with the ROUGE (Lin, 2004) and BERTScore (J. Zhang et al., 2020) metric (see Table 5.1).

Setup [R2] After generating the abstractive summary, an extractive recap is produced. This is done by searching for the source sentences on the basis of the similarity to the summary sentences. This approach is called "Sim. Search" in this work. Heuristics such as the abstracts of the papers and TextRank serve as an extractive baseline.

Setup [R3]. For the last research question, the quality of the output text is evaluated. Aspects such as readability, coherence, consistency, fluency, and relevance are examined. The outputs of the different low-resource methods, such as abstracts of papers, TextRank (Mihalcea and Tarau, 2004), Sim. Search (see Section 3.3.4), BART (Lewis et al., 2020), and SLED (Ivgi et al., 2023) are compared with those of a very large and sophisticated GPT model.

5.3. Evaluation of OpenReview Corpora

To verify the quality of human-composed summaries and input documents in the corpora, the newly scraped datasets OpenReview Contribution and OpenReview Summary (see Section 4.2.3) are compared with existing collections in the long-document field. Datasets such as ArXiv (Cohan et al., 2018), PubMed (Cohan et al., 2018), BigPatent (E. Sharma et al., 2019), BillSum (Kornilova and Eidelman, 2019), GovReport (L. Huang et al., 2021), SciTLDR (Cachola et al., 2020), and FacetSum (Meng et al., 2021) are used for comparison. An overview of existing datasets is given in Section 2.7. Datasets are

	Existing Datasets							Own Datasets	
	ArXiv	PubMed	BigPatent	BillSum	GovReport	SciTLDR	FacetSum	OR Contr.	OR Summ.
# of docs	215 K	133 K	1,340K	21.3 K	19.5 K	3.2 K	5.8 K	1.7 K	11 K
Tokens per summ	242	208	117	243	607	21	195	94	93
Tokens per doc	6446	3143	3573	1686	9409	5066	5698	8402	7440
Sents per summ	6.3	7.1	3.6	7.1	21.4	1	8.5	4.6	4.4
Sents per doc	251	102	143	42	300	224	225	364	318
Compress. $_{\text{token}}$	41.2	16.6	36.3	12.2	18.7	291.7	31.9	128.4	116.3
Compress. $_{\text{sent}}$	44.3	15.6	58.7	9.7	18.1	219.9	28.4	105.5	72.3
Coverage	0.920	0.893	0.861	0.913	0.942	0.920	0.937	0.891	0.889
Density	3.7	5.6	2.1	6.6	7.7	3.7	6.2	2.6	2.6
Redundancy	0.144	0.146	0.223	0.163	0.124	0	0.167	0.141	0.142
Uniformity	0.894	0.896	0.922	0.903	0.932	0.897	0.942	0.935	0.935
Multi-target	no	no	no	no	no	yes*	no	yes	yes

* only test/val. set

Table 5.2.: Evaluation of OpenReview corpora, data extended from Koh et al. (2022). Datasets used for this work are highlighted in grey. The OpenReview corpora are comparable to other long-document datasets in several aspects.

evaluated with respect to the number of tokens and sentences, compression, coverage, density, redundancy, and uniformity scores (see Table 5.2). The data of Koh et al. (2022) was extended with the SciTLDR, FacetSum, and OpenReview evaluation scores. The individual evaluation metrics for the comparison of the datasets are explained in Section 2.7.1.

5.3.1. Dataset and Document Size

Both newly generated corpora, namely OpenReview Contribution and OpenReview Summary (see Section 4.2), are compared with respect to their size to other single-document datasets. Unlike most other published long-document collections, OpenReview Contribution and OpenReview Summary have a smaller number of documents. However, both datasets are similar in size to SciTLDR (Cachola et al., 2020) and Facetsum (Meng et al., 2021). The summaries are shorter compared to ArXiv (Cohan et al., 2018), PubMed (Cohan et al., 2018) or FacetSum (Meng et al., 2021), but the input documents are longer on average. Therefore, the compression ratio based on sentences and tokens is much higher (see Table 5.2).

In Figure A.1 the number of tokens per paper is shown. It is noticeable that FacetSum (Meng et al., 2021) has one of the shortest input articles, while ArXiv (Cohan et al., 2018) has the longest with up to almost 35,000 tokens. The documents in OpenReview Contribution and OpenReview Summary can be quite long as well and are comparable to ArXiv in size. Furthermore, the maximum token capacity is indicated with a red mark in Figure A.1. With the hardware limitations given, the SLED model has a capacity of 12,000 tokens. Therefore, at least 85% of the articles for each dataset can be processed entirely.

5.3.2. Abstractiveness

The abstractiveness of documents can be measured by extractive coverage and density (see Section 2.7.1). The coverage of the OpenReview datasets is similar to the values of other scientific corpora such as ArXiv and PubMed (Cohan et al., 2018). However, the density is much smaller and, therefore, the average length of the extractive fragments is shorter. This suggests that words from the summary do exist in the input document, but are in a different order. Therefore, the summary text might appear to be more abstract. Only BigPatent (E. Sharma et al., 2019) has a smaller density with a value of 2.1, compared to 2.6 of the OpenReview Contribution and OpenReview Summary.

5.3.3. Redundancy

Redundancy evaluates the degree to which the information is repeated in the summary (see Section 2.7.1). It is measured by the average ROUGE scores between all pairs of sentences within a summary. OpenReview Contribution and OpenReview Summary have one of the lowest redundancy scores among the datasets. Only GovReport (L. Huang et al., 2021) has a lower value with 0.124. The summary in SciTLDR (Cachola et al., 2020) consists of only one sentence, so redundancy cannot be calculated and is set to 0.

5.3.4. Uniformity

Uniformity measures the degree of distribution of important information throughout the document (see Section 2.7.1). The greater the uniformity, the more information is distributed. OpenReview Contribution and OpenReview Summary have one of the highest uniformity scores, only exceeded by FacetSum (Meng et al., 2021) with a score of 0.942. As a result, the information in these datasets is very dispersed throughout the documents.

5.3.5. Multi-targeting

One of the advantages of the newly formed OpenReview datasets is their multi-target aspect. For every input document, there are several possible summary texts, each composed of a different expert. This characteristic makes the datasets unique, since only SciTLDR (Cachola et al., 2020) provides multiple outputs, but only for their test and validation sets. Therefore, during training with SciTLDR, no multi-targeting is possible. In addition, the OpenReview corpora allow it to train for length controllability, as the summaries can be divided into different length categories.

5.3.6. Domain and Style

The OpenReview corpora are from the domain of scientific articles with a focus on computer science. Only ArXiv (Cohan et al., 2018) and SciTLDR (Cachola et al., 2020) include articles in the field of information technology. All other long-document datasets, such as PubMed (Cohan et al., 2018), BigPatent (E. Sharma et al., 2019), BillSum (Kornilova and Eidelman, 2019), or GovReport (L. Huang et al., 2021), do not cover technology-influenced scientific research topics.

It should also be mentioned that the style of the summaries in the OpenReview datasets is closer to an actual human-written review, as the texts are from a third-person perspective. For example, ArXiv and PubMed adopt only the abstracts of the articles written by the authors as summary references.

5.4. Performance Results

In this section the first and second research question (see Section 5.1) are explored. The trained efficient long-document transformer SLED (Ivgi et al., 2023) is evaluated and compared with abstractive and extractive approaches (baselines described in Section 5.2.2). All models were tested with the newly sourced datasets OpenReview Contribution and OpenReview Summary (see Section 5.3), and the additional selected dataset FacetSum (Meng et al., 2021). Throughout the experiments, the human-written summaries from the summaries serve as a "golden standard".

The performance of the models was measured with the lexical-based ROUGE and, in some cases, additionally with the semantic-based BERTScore metric (see Section 2.6.1). For the backbone of BERTscore we use the *microsoft/deberta-large-mnli*[1] model as it shows good evaluation performance and high human correlation, reported by the official Github repository[2].

5.4.1. OpenReview Models

The summaries of OpenReview datasets are of different lengths, and hence the summary texts were assigned to various length bins. These length tokens were presented as prefixes to the model and served as a guiding signal. The model should learn two aspects: to summarise the main points and the contribution of the text, and to present the content in different lengths. The performance of both characteristics was evaluated. As mentioned above, the

[1] https://huggingface.co/microsoft/deberta-large-mnli
[2] https://github.com/Tiiiger/bert_score

summaries composed by academic experts serve as a ”gold standard” for the evaluation.

Summary Performance

The efficient long-document SLED (Ivgi et al., 2023) model is compared with the abstractive BART (Lewis et al., 2020) model. Furthermore, the extractive similarity search approach (Sim. Search), which is based on the SLED output, is evaluated. TextRank (Mihalcea and Tarau, 2004) serves as an extractive baseline. In addition, the abstracts of the scientific article provide a simple summary heuristic. The summaries are assessed by the metrics ROUGE (Lin, 2004) and BERTScore (J. Zhang et al., 2020).

From Table 5.3 it can be seen that a simple generic summary of TextRank does not cover all information needs. The simple heuristic even outperforms TextRank in terms of semantics (0.072 difference in BERTScore) and lexical similarity (exceeds all ROUGE evaluations). In addition, we can show that the similarity search approach (Sim. Search) performs quite well. The extractive method outperforms the extractive baseline TextRank and the heuristic in both ROUGE and BERTScore.

The trained SLED model demonstrates that it is highly capable of summarising the main information from scientific articles. The SLED model outperforms TextRank by 12.71% on ROUGE-Lsum and by 0.135 on BERTScore, showing that the model has a greater ability to generate good summaries of articles. The SLED model is also compared to a trained $\text{BART}_{\text{base}}$ model with a maximum input size of 1024 tokens. Therefore, for BART only the abstract and parts of the introduction serve as an information source. Surprisingly, the BART model performs on average very well (only 0.06 ROUGE-Lsum and 0.006 BERTScore difference from SLED), although it can process only a subset of the input. Similar results can be seen for the model trained in the more general OpenReview Summary dataset (shown in Table A.3). However, if each length size is compared separately (see Figure 5.1 and Table A.2), it is noticeable that the SLED approach scores slightly higher in terms of ROUGE1 and BERTScore in longer summaries. Therefore, it can be advantageous to use efficient long-document models, which can take the full text of the document as input, for longer and more comprehensive summaries.

Surprisingly, the heuristic of using the abstracts of the scientific articles performs considerable well. In particular, for longer summaries (see Figure 5.1), the heuristic achieves results similar to those obtained with the extractive similarity search approach (Sim. Search).

Method	Input source	Length signal	ROUGE1	ROUGE2	ROUGELsum	BERTScore
heuristic	paper abstr.	-	32.73	9.39	20.23	0.220
TextRank	full paper	no	29.09	6.21	19.26	0.114
TextRank	full paper	yes	30.95	6.52	20.41	0.128
Sim. Search	summ.+paper	yes	35.77	9.60	23.44	0.229
$BART_{base}$	1K tokens	yes	36.81	10.45	33.06	0.276
$SLED_{base}$	12K tokens	no	32.68	9.90	29.40	0.268
$SLED_{base}$	12K tokens	yes	**36.95**	**10.81**	**33.12**	**0.282**

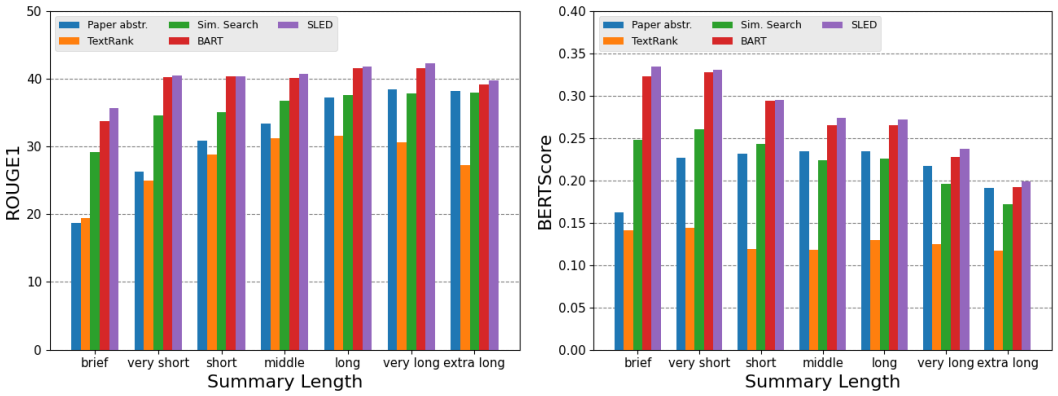
Table 5.3.: Performance comparison on the OpenReview Contribution dataset. SLED outperforms extractive baselines and base-model BART in ROUGE and BERTScore. A length signal as additional input increases performance and is essential for a good summary.

Length Controllability

As the user should be able to control the length and summary lengths have an influence on performance metrics (see Table 5.3), length controllability of the model is essential. Therefore, it was also evaluated how well the SLED (Ivgi et al., 2023) model can control the length of the summaries. The model relies mainly on the length signal to adjust the length of the summaries. The guiding signal is presented as a prefix with the input during training and inference. However, the BART (Lewis et al., 2020) model, which serves as the backbone of the SLED model, also provides a length penalty for the generation process.

The length penalty is a specific parameter of the BART model and influences the generation of the output during the beam search. It can be seen as a scaling factor. Therefore, the greater the length penalty parameter, the more text the model generates (see Table A.1). However, empirical observations showed that as expected a higher length penalty value produced a longer summary text but also increased the degree of hallucination. Therefore, the length is controlled solely by the length signal and the BART length parameter is set to a standard value of 1.

Examples of the length distribution of the summaries generated during training can be seen in Figure A.2. With each training step, the model adjusts the output length according to the input signal. However, the model tends to produce summaries that are slightly shorter than the reference output. The difference in length is measured by comparing the length bins of the reference and the generated summaries and computing the mean absolute deviation (MAD) (see Table A.1). After around 17,500 steps, the model follows the length signal quite well. Further analysis showed that the fully trained SLED and BART models had a Pearson correlation of up to 0.49 and 0.58, respectively. Hence, both models show good length control abilities for the seven possible length categories.



a) Performance on different summary lengths - ROUGE1

b) Performance on different summary lengths - BERTScore

Figure 5.1.: Performance comparison on different summary lengths on the OpenReview Contribution dataset. The efficient long-document SLED model performed on average the best. Details of the scores can be seen in the Appendix (see TableA.2).

5.4.2. FacetSum Aspect-oriented Model

For further exploration of research questions 1 and 2 (see Section 5.1), the SLED (Ivgi et al., 2023) model was also evaluated on the FacetSum (Meng et al., 2021) dataset (see Table5.4). The human-written summaries serve as a "gold standard". The dataset can be used for the task of aspect-oriented summarization as the summaries are divided into several aspects. The results provided by the FacetSum article were adopted for the comparison; however, FacetSum modified the $\text{BART}_{\text{large}}$ (Lewis et al., 2020) model to process up to 10,000 tokens. Therefore, the BART model can handle a ten-times larger input compared to the standard BART model, which has an input size of 1024 tokens. Furthermore, the $\text{SLED}_{\text{base}}$ model has approximately only a third of the parameters compared to the $\text{BART}_{\text{large}}$ model. Taking this into account, the evaluation focusses on the trade-off between efficiency and performance.

The evaluation shows that the $\text{BART}_{\text{large}}$ model performs slightly better than the $\text{SLED}_{\text{base}}$ model, even if the SLED model receives an additional length signal. However, FacetSum states that a 16 GB VRAM is needed to train the respective model, which is double the amount of the efficient long-document SLED model with $\text{BART}_{\text{base}}$ as the backbone.

Therefore, the $\text{SLED}_{\text{base}}$ model shows a good trade-off between performance and efficiency. By providing an additional length signal to the $\text{SLED}_{\text{base}}$ model, we can improve the results in all four aspects. Therefore, a length-oriented summary is advantageous and can help cover more information. However, the scores are still lower compared to the $\text{BART}_{\text{large}}$ results (with a ROUGELsum difference of 0.06 in purpose, 1.66 in method,

Method	Type	Aspect Signal	Input Source	#Params	Aspect			
					Purpose	Method	Findings	Value
Lead-K	exstr.	no	first 3 sent.	-	17.83	15.29	15.92	16.08
Tail-K		no	last 3 sent.	-	21.67	12.62	16.66	17.43
TextRank		no	corr. sect.	-	21.67	13.62	18.63	19.23
HipoRank		no	corr. sect.	-	22.73	15.20	18.38	19.68
Sim. Search		no	summ.+paper	-	30.63	24.45	22.67	23.22
$BART_{large}$	abstr.	no	10K tokens	406M	41.21	20.5	14.33	5.07
$BART_{large}$		yes	10K tokens	406M	**42.55**	**28.07**	**28.98**	**28.70**
$SLED_{base}$		yes	12K tokens	139M	42.23	25.16	24.90	25.94
$SLED_{base}$ + len.		yes	12K tokens	139M	***42.49***	***26.41***	***26.55***	***27.08***

Table 5.4.: Performance comparison on the FacetSum dataset. The results of the baseline methods Lead-K, Tail-K, TextRank, HipoRank, and $BART_{large}$ are adopted from the work of FacetSum (Meng et al., 2021). $SLED_{base}$ incorporating the length signal almost reaches the performance of $BART_{large}$. However, $SLED_{base}$ needs substantially less parameters and thus computational power.

2.43 in findings, and 1.62 in value).

In general, abstractive methods performed better than extractive methods. However, if the BART model is not receiving any guidance signal, the model is not able to summarise aspect-relevant information. The model then has, compared to extractive methods, a lower score on the aspects *"findings"* and *"value"*. Consequently, it is essential to provide guiding signals during training to reach high performance.

The similarity search approach (Sim. Search) exceeds other heuristics and extractive methods and shows on some aspects almost similar results to those of the SLED model. Therefore, we can conclude that this approach extracts sentences effectively and can generate meaningful summaries.

5.5. Comparison of Text Informativeness

To examine the third research question (see Section 5.1) the summaries generated by different methods are compared in respect to text informativeness such as readability (see Section 5.5.3) and text quality (see Section 5.5.4). Low-resource methods (see Sections 5.2.1 and 5.2.2) are put in contrast to a sophisticated state-of-the-art transformer model described in Section 5.5.1. For evaluation, a subset of the OpenReview Contribution test set is composed that includes 91 randomly selected summaries and their respective articles. The summaries are categorised by the label *"long"*, hence, have between 90 and 120 words. Furthermore, the methods are tested on the subset with respect to their performance (see Section 5.5.2) to create a better picture of the comparison. For a further study in Section 5.5.6, differences of selected sample summaries are compared and the characteristic of each method is examined.

5.5.1. Sophisticated Comparison Model

The summaries generated from the trained SLED (Ivgi et al., 2023) model, the BART (Lewis et al., 2020) model, extractive methods such as TextRank (Mihalcea and Tarau, 2004) and Sim. Search (see Section 3.3.4), and a simple heuristic are compared with the recaps produced by a sophisticated GPT (*gpt-3.5-turbo-16k*[3]) model. The selected GPT model is capable of processing up to 16 thousand tokens. Therefore, almost all documents can be processed entirely, even surpassing the SLED model with an input size of 12 thousand tokens using the setup in this work. The GPT model is not fine-tuned on the OpenReview Contribution subset, hence, making it a zero-shot task. To generate summaries with a focus on contribution, the model was prompted through the OpenAI API[4]. Standard parameters and a temperature value of 0 were used to generate correct factual summaries that are comparable to the output texts generated by SLED and BART. To produce texts that are similar in style, the GPT model received the following prompt: *"Summarise this paper in 90 words with a focus on the contribution in a third-person perspective"*. Examples of summaries are shown in Section 5.5.6. A notable difference between the models is the number of parameters. The first version of an article published by Singh et al. (2023) suggests that the selected GPT-3.5-turbo model has around 20 billion parameters. The SLED model with $\text{BART}_{\text{base}}$ as the backbone has about 139 million parameters (Lewis et al., 2020). Consequently, the SLED model is only approximately 0.7 percent of the GPT model size.

5.5.2. Performance Comparison on Subset

To complete the picture of the comparison, the performance of the models is compared on the composed OpenReview Contribution subset. From Table 5.5 it can be seen that the trained SLED (Ivgi et al., 2023) and BART (Lewis et al., 2020) models even outperform the GPT model in both metrics, ROUGE (Lin, 2004) and BERTScore (J. Zhang et al., 2020). For the GPT model, however, it is a zero-shot task. Nevertheless, both models, SLED and BART, are better at producing more similar output texts and mimicking the style of human-written reference summaries. Surprisingly, the SLED model does not exceed BART in all ROUGE measures. However, the SLED model shows a better result on the semantic-based BERTScore metric. It is also worth mentioning that the similarity search approach (Sim. Search) performs better than the extractive baseline TextRank and is almost as good as GPT in terms of ROUGE (only 0.15 difference in ROUGE2). This result suggests that the extracted sentences cover the content well and contain

[3]https://platform.openai.com/docs/models/gpt-3-5
[4]https://platform.openai.com/docs/api-reference/introduction

Method	Type	#Params	ROUGE1	ROUGE2	ROUGELsum	BERTscore
paper abstr.	extr.	-	32.73	9.40	20.22	0.2515
TextRank		-	32.13	6.64	20.83	0.1643
Sim. Search		-	38.30	10.10	24.47	0.2574
$GPT_{zero\text{-}shot}$	abstr.	∼20B	40.45	10.25	25.40	0.2802
$BART_{base}$		139M	**42.55**	12.07	**27.62**	0.2929
$SLED_{base}$		139M	42.31	**12.34**	27.25	**0.2935**

Table 5.5.: Performance comparison on the subset of OpenReview Contribution. The subset only includes summaries with the label *"long"*. The summaries composed of experts serve as references. SLED and BART are trained in the OpenReview Contribution training set and perform best. For GPT it is a zero-shot task. Despite the lack of fine-tuning, GPT almost reaches the SLED and BART scores. Still, SLED and BART are the best in mimicking the style of the human-written summaries.

essential information.

5.5.3. Readability Comparison

The readability of the automatically generated and reference summaries is compared using the *Textstat* library[5]. *Textstat* measures the readability by applying traditional automatic readability metrics (see Section 2.6.3).

From Table 5.6, it can be seen that all summaries are very similar in terms of Flesch Reading Ease (FRE) readability, with a score of around 30. Therefore, the summary texts are either difficult or very difficult to read. However, when assessed by the average grade level, extractive methods seem to be challenging to understand. Abstractive methods, both human-written and automatically generated, generally have better readability. In terms of automatic summarization, surprisingly, the BART (Lewis et al., 2020) approach scores the best, with the GPT method coming second. In addition, the abstracts of the papers are on average more difficult to read than human-written summaries. Readers need around one grade-level higher education to fully grasp the scientific articles. Consequently, human summaries tend to be more readable than the actual scientific text. LLMs that learn from human-written data capture the style and generally produce less complex summaries.

5.5.4. Quality Comparison

The quality of the summary texts is compared in terms of their coherence, factual consistency, fluency and relevance with respect to the human-written references of the OpenReview Contribution dataset. To assess the dimensions, the UniEval evaluator is applied (see Section 2.6.3). In Table 5.6 it

[5]https://github.com/textstat/textstat

Method	Type	#Params	Avg. #Diffic. Words	FRE Score↑	FRE Readability	Grade (Average)↓
paper abstracts	extr.	-	29.94	27.24	very difficult	15.99
TextRank		-	27.02	36.18	difficult	15.80
Sim. Search		-	34.01	27.63	very difficult	16.49
reference summ.	abstr.	-	33.32	32.96	difficult	15.01
$GPT_{zero\text{-}shot}$		∼20B	37.30	28.39	very difficult	14.88
$BART_{base}$		139M	31.77	33.24	difficult	14.67
$SLED_{base}$		139M	32.79	29.44	very difficult	15.55

Table 5.6.: Readability comparison of summaries evaluated with traditional readability metrics. Abstractive methods generate the most understandable output texts with respect to the average grades. Transformer models such as SLED, BART, and GPT are capable of producing understandable text that is comparable to human-written reference summaries.

can be seen that the summary texts of extractive methods show a lack of coherence and relevance. However, the similarity search method (Sim. Search) shows higher factual consistency than SLED (Ivgi et al., 2023) and BART (Lewis et al., 2020). Therefore, simply extracting phrases from the input texts can be beneficial in terms of factuality and can increase the trustability of the system. GPT and SLED generate more relevant summaries with high-qualitative sentences and have the best average score out of all automatic summarization methods. Hence, it can be concluded that abstractive approaches are more suitable for text summarization tasks. In addition, LLMs with fewer parameters are comparable to larger sophisticated models such as GPT in terms of text quality. In Table 5.7, it can be seen that SLED scores only 3.36 points less on the overall score compared to GPT. Therefore, for specific trained tasks, it is sufficient to use smaller and more efficient models to achieve a similar result.

Surprisingly, the GPT model scores on all dimensions are relatively high, almost reaching the same overall score as the abstracts of scientific articles (only 0.69 points less). As expected, the abstracts are more coherent and consistent; however, the summaries by the GPT model are more fluent and contain more relevant information relative to the human-written reference summaries. Furthermore, SLED scores higher on fluency and relevance than abstracts of human-written articles. Therefore, it can be concluded that the strength of transformer models is to generate highly fluent and relevant text if trained with high-quality data.

5.5.5. Text Extraction Locations

In this experiment, important locations in the input documents were investigated. It was detected where the similarity search approach Sim. Search (described in Section 3.3.4) extracts the most likely sentences. The approach

Method	Type	#Params	Coherence	Consistency	Fluency	Relevance	Average
paper abstr.	extr.	-	**94.19**	**94.35**	88.80	85.42	**90.69**
TextRank		-	40.36	68.28	76.71	35.82	55.29
Sim. Search		-	61.55	82.91	87.55	55.21	71.80
$GPT_{zero\text{-}shot}$	abstr.	∼20B	92.37	84.47	**91.63**	**91.52**	90.00
$BART_{base}$		139M	90.22	82.84	86.11	86.81	86.49
$SLED_{base}$		139M	89.08	80.99	88.93	87.54	86.64

Table 5.7.: Text quality comparison including dimensions coherence, consistency, fluency and relevance. The summaries composed by experts serve as a reference. The abstracts of the article are of the highest quality on average. However, they lack fluency and relevance. GPT followed by SLED has the best results of all automatic summarization methods showing strong performance in fluency and relevance.

is based on the summary generated by SLED (Ivgi et al., 2023). Evaluations were performed on the OpenReview Contribution test set. In the first experiment, the texts were divided into ten parts of equal size (see Figure 5.2a) and in the second into the five most common sections of scientific articles, namely the abstract, introduction, method, result, and conclusion (see Figure 5.2b). To identify which section the paragraph belongs to, a similar approach was followed to FacetSum (Meng et al., 2021). Sections were identified using keywords and heuristics based on the structure of scientific articles. The summaries of two labels, namely *"brief"* and *"extra long"*, were examined and compared with each other.

From the experiments (see Figure 5.2) it was found that the brief summaries originate with high probability from the first parts of the article, namely the abstract (46.6%) and the introduction (23.3%), followed by the last part, the conclusion with the appendix (19.4%). Longer summaries have a different distribution. Only 29.7% of the sentences come from the abstract followed by the introduction, method and result section with 20.1%, 19.6%, and 18.4%, respectively. Therefore, longer summaries include more detailed information on the methods and results of the article. However, models must be able to process larger input sizes to incorporate the information. Otherwise, automated summaries might not cover all the information needed, and models show lower performance on long documents by including less relevant information.

5.5.6. Comparison of Summary Examples

Two examples of human-written recaps are compared with the automatic generated summaries from selected scientific articles. The articles are chosen from the OpenReview Contribution dataset and introduce BRAX[6] (Free-

[6]https://github.com/google/brax

man et al., 2021), which is an open source physics engine, and EdgeBank[7] (Poursafaei et al., 2022) a new benchmark for dynamic link prediction. Both papers were submitted at the NeurIPS[8] conference, the former at NeurIPS 2021[9] the latter at NeurIPS 2022[10].

Each summary text has its characteristics, which are discussed in Section 5.5.6. Both examples demonstrate possible model-specific errors and features. Furthermore, in Section 5.5.6 the informativeness of the text is examined in relation to the length of the summary.

Text Characteristics

First, the summaries about BRAX (Freeman et al., 2021) are examined. The summaries generated are shown in Table 5.8. Characteristics of each summary are discussed in the following points.

- **Paper abstract**. The abstract gives a short overview of the main contribution of BRAX. The abstract is relatively short and does not provide detailed information.
- **Human-written**. The human-written summary has a unique structure. The author uses enumerations to present the characteristics of the physics engine. Although written by an expert in the academic field, the author uses an unconventional writing style. For example, the connector "&" is used instead of "and", numbers are not written out, and the name of the physics engine is in lowercase. However, in general, the summary gives a good overview and is well-structured.
- **BART**. The model provides a well-written summary. Names of important engines and abbreviations are mentioned. However, the text also contains a mistake. The model generated "MuJoColike" instead of "MuJoCo-like". Therefore, a hyphen is missing. Furthermore, the model produced hallucination in form of a human-like comment. Some of the training samples were found to include additional unwanted information in the form of notes and opinions at the end of the summary. The model likely produced such comments to meet the length requirement because it could not extract all the important information from the abstract and introduction sections.
- **SLED**. The SLED model produced an informative summary. Only small mistakes were made, for example, by writing the abbreviation with a lower character at the beginning ("jAX" instead of "JAX"). Also, it generated a very long nested sentence, which might make it impractical to read and, therefore, hard to understand.

[7]https://github.com/fpour/DGB
[8]https://neurips.cc/
[9]https://openreview.net/forum?id=VdvDlnnjzIN
[10]https://openreview.net/forum?id=1GVpwr2Tfdg

- **GPT**. The generated text contains very similar information compared to the other summaries. However, no obvious mistakes were made. The text is well written and easy to read.
- **Similarity Search.** The extracted text of the similarity search method mainly includes information from the abstract of the article. Due to the semantic search approach, the content is very similar to the summary of SLED. However, the text provides complementary information to the generated summary of SLED.
- **TextRank**. This extractive method produces the least informative summary. The text seems incoherent and is therefore difficult to read.

The second analysis is about EdgeBank (Poursafaei et al., 2022). The texts produced by the different methods are in Table 5.9. Characteristical differences are discussed in the following points.

- **Paper abstract.** The authors of EdgeBank summarise the article in a clear manner. The abstract is relatively long and contains a lot of information.
- **Human-written.** The text written by the expert contains enumerations. Among other things, it mentions that five new datasets are introduced. However, as also mentioned in the abstract, a total of six datasets are presented. Therefore, this is a content error that reduces the factuality.
- **BART.** The text generated by the BART model is factually correct, and the summary is informative and well-written.
- **SLED.** The SLED model is also characterized by high factuality. The text is easy to read and understand, and the summary effectively covers the main points of the text.
- **GPT.** The sophisticated GPT model generates a comprehensive and factually correct summary. The text is coherent, consistent, fluent, and contains relevant information.
- **Similarity search.** The Similarity Search approach extracts relatively relevant sentences. However, these sentences are not coherent and do not cover all the information included in the summary produced by SLED.
- **TextRank.** The TextRank method leaves out important facts. For example, the new datasets introduced in the paper are not mentioned. The text is also not coherent and therefore more difficult to read.

From the examples, we can conclude that, of the automatic methods, GPT generates a highly informative and most readable summary. However, BART and SLED also produce well-written summaries that capture the main information of the article.

In the first example, BART produced some hallucinative content, probably because relevant information was not included in the abstract and introduction section of the article and the length requirement of the summary had to

be satisfied. Further analysis showed that for BART nine of the 91 samples, therefore 9.98%, contained hallucinative content in the form of invented human-like comments. In comparison, only one summary (1.1%) produced by SLED contained a similar hallucinatory comment.

A contributing factor are also training data. Texts written by humans also show that people sometimes make formal errors and inaccuracies. These errors are then learnt and adopted by the models during training. As a result, the models are sometimes inaccurate, and the texts contain false information, such as improper abbreviations. This contributes to the slightly lower performance of BART and SLED.

Length- and Aspect-dependent Text Informativeness

In Table 5.10 the length controllability of the trained SLED (Ivgi et al., 2023) model can be seen. Several summaries were generated that were controlled by the input prefix, including *"brief"*, *"short"*, *"long"*, and *"extra long"*. The model can follow the length prompt quite well.

From the example in Table 5.10, it is evident that the longer the summary, the more information is included. In the brief summary, only the most important facts about the article are given. The short summary contains additional information. The long summary is already extensive. However, the extra-long summary includes the most details and is the most comprehensive recap. Surprisingly, the model learnt the inherent structure of the human-wirtten summaries and generated an enumeration of contributions at the end.

In this example, the disadvantages of small-language models and imperfect training data generated by humans can be seen. The model generates spurious abbreviations and references. In addition, words are being put together incorrectly, for example, "inJAX" instead of "in JAX". Despite these small inaccuracies, the summaries contain the main information and are still well readable. As seen in Table 5.8 and Table 5.9 humans make comparable mistakes. Therefore, the models are likely to learn these imperfections and might generate similar errors.

In Table 5.11 an example of an aspect-oriented summary of the article by H.-C. Huang (2016) is presented. The text is summarized according to the purpose, value, methods, and findings aspects in the FacetSum dataset. Comparing the summaries shows that the information is quite repetetive. The summaries of the values and methods are quite similar, even starting with the same sentence. The findings do not provide detailed information, but only outline the setup of the experiment. Therefore, the summaries include basic information but can still be improved in their informativeness.

5.6. Findings and Limitations

The following sections outline the key findings from the research and the limitations that need to be addressed for further improvements.

5.6.1. Findings

A high-quality dataset such as OpenReview is essential to efficiently train a model. The newly sourced dataset fulfils this requirement as it is composed by humans and has a multi-target aspect. In addition, the summaries were divided into several length categories. These labels can be used as guiding signals during training.

Furthermore, the trained SLED (Ivgi et al., 2023) efficient long-document transformer model performs very well in the length- and aspect-dependent summerization task. Taking the whole document into account increases performance, with the biggest boosts seen for longer and brief summaries. The SLED model achieves slightly higher results than the BART model for the lexical-based metric ROUGE1, ROUGE2, and ROUGELsum (Lin, 2004) with a difference of 0.14, 0.36, and 0.06, respectively. Furthermore, the SLED model also performs better with regard to the similarity-based BERTScore (J. Zhang et al., 2020) metric with 0.06 difference. These results show that the efficient long-document transformer can effectively boost performance without having more parameters and increasing computational costs.

Moreover, a simple heuristic, such as taking the abstracts of the papers as recaps, does not complete the information need entirely. The abstracts reach lower performance in ROUGE and BERTScore compared to the extractive similarity search approach and the abstractive SLED model. For example, the similarity search method and the SLED model achieve a higher score on the BERTScore metric with differences 0.015 and 0.063, respectively. Therefore, summaries generated by these models are more similar to recaps written by experts.

The similarity search approach, which is based on the summaries generated by the SLED model, performs better compared to the exractive TextRank baseline with a difference of 0.154 in BERTScore. Consequently, this method is more accurate and includes more important information. In addition, it can increase the trustability of the system since users can see the origin of sentences and phrases.

The experiments showed that a guiding signal, such as a length and aspect token, can increase coverage. The length parameter is important to satisfy all information needs. Experiments with and without the signal show that there is a gap for lexical- and similarity-based scores. As a consequence, controllability is essential for a performant summarization system.

Surprisingly, the relatively small SLED model with BART (Lewis et al.,

2020) as the base model shows performance and text quality comparable to those of LLMs with a large number of parameters. SLED achieves results similar to those of the sophisticated GPT model for dimensions of coherence, consistency, fluency, and relevance and scores only 3.36 points less on average. Therefore, the SLED model is a good alternative in low-resource systems, which can still generate high-quality summaries.

However, it was also shown that for some specific tasks, the parameter size of the model made a slight difference. For example, for the aspect-oriented task in the FacetSum (Meng et al., 2021) dataset, the $\text{BART}_{\text{large}}$ model performed better than the smaller $\text{SLED}_{\text{base}}$ model. Nevertheless, this performance boost also comes with double computational costs as a result of the higher number of parameters.

Another finding is that for longer summaries, including more information from the input articles is advantageous. The efficient long-term transformer SLED demonstrates better performance in both ROUGE and BERTScore compared to the standard LLM BART.

5.6.2. Limitations

Hallucination remains a significant issue in natural language processing and the models used, especially when it comes to incorrect abbreviations and comments that mimic human-like nuances but lack accuracy. These errors can lead to misleading information and diminished trust in the system's outputs.

Although advancements like sentence tracing have been implemented to enhance the reliability and transparency of generated content, this approach has yet to fully resolve inaccuracies in summary sentences. Sentence tracing helps to track the origin and evolution of information, thereby increasing trustworthiness, but it does not inherently correct or adjust erroneous summaries.

Furthermore, even with the deployment of highly efficient transformer models, the technology is constrained by existing hardware limitations. These constraints impact the model's performance, including its ability to handle large-scale data and execute complex computations in real-time. Addressing these hardware limitations is crucial for improving the overall accuracy and efficiency of language models, and ongoing research and development are necessary to bridge these gaps.

One of the significant challenges in developing effective summarization models is the difficulty in gathering accurate and reliable datasets. Ensuring that the data is both high-quality and representative of the target tasks is essential, yet obtaining such data can be challenging. Additionally, sourcing a sufficient amount of data to train models robustly is often difficult, especially when dealing with specialized domains or tasks that require

human-generated content. These limitations can hinder the performance and generalizability of models, emphasizing the need for ongoing efforts in data collection and curation.

In text quality measurement, automatic parameters and models have significant limitations. They often miss subtle language nuances like context and tone, leading to incomplete assessments. These models can also struggle with overfitting or underfitting, reducing their effectiveness across different types of text. Their lack of transparency makes it difficult to understand or trust how they evaluate text quality. Additionally, relying on these models can narrow the scope of analysis, potentially overlooking important linguistic and contextual factors.

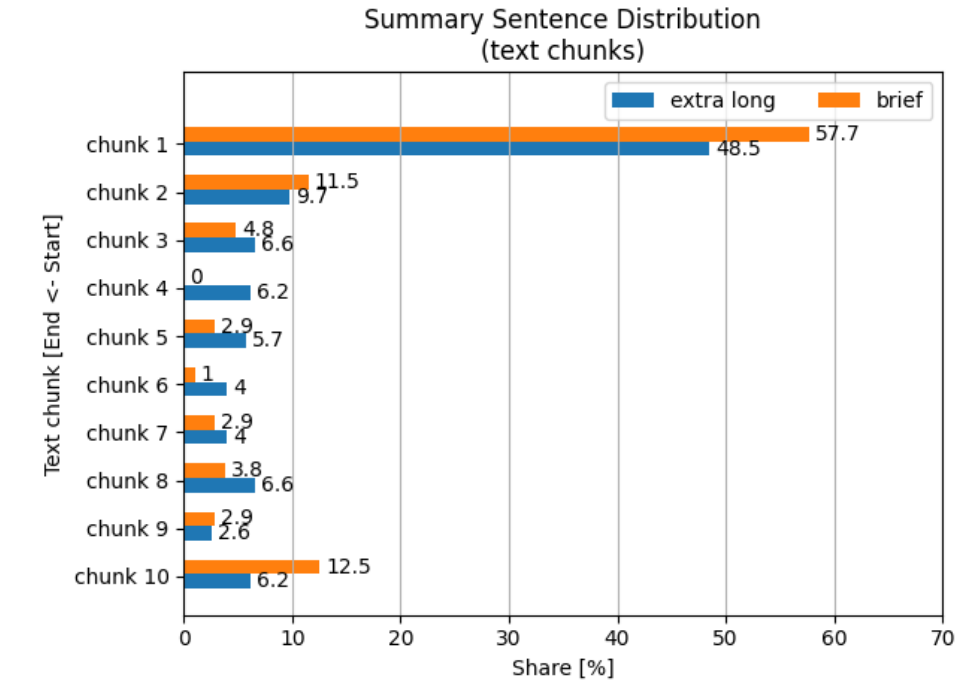
The fusion-in-decoder approach used in the efficient long-document transformer model, while effective, faces limitations as newer models are developed. Upcoming models may introduce more advanced techniques for handling long texts, such as improved mechanisms for capturing dependencies across distant parts of a document. These innovations could outperform the fusion-in-decoder approach by offering more nuanced and accurate text processing. As these new models emerge, the current model might struggle to keep pace with advancements, potentially becoming less effective in capturing the full complexity of long documents. Additionally, the rigid structure of the fusion-in-decoder approach may limit its adaptability to new features and improvements, making it increasingly challenging to maintain its relevance in a rapidly evolving field.

5.7. Summary

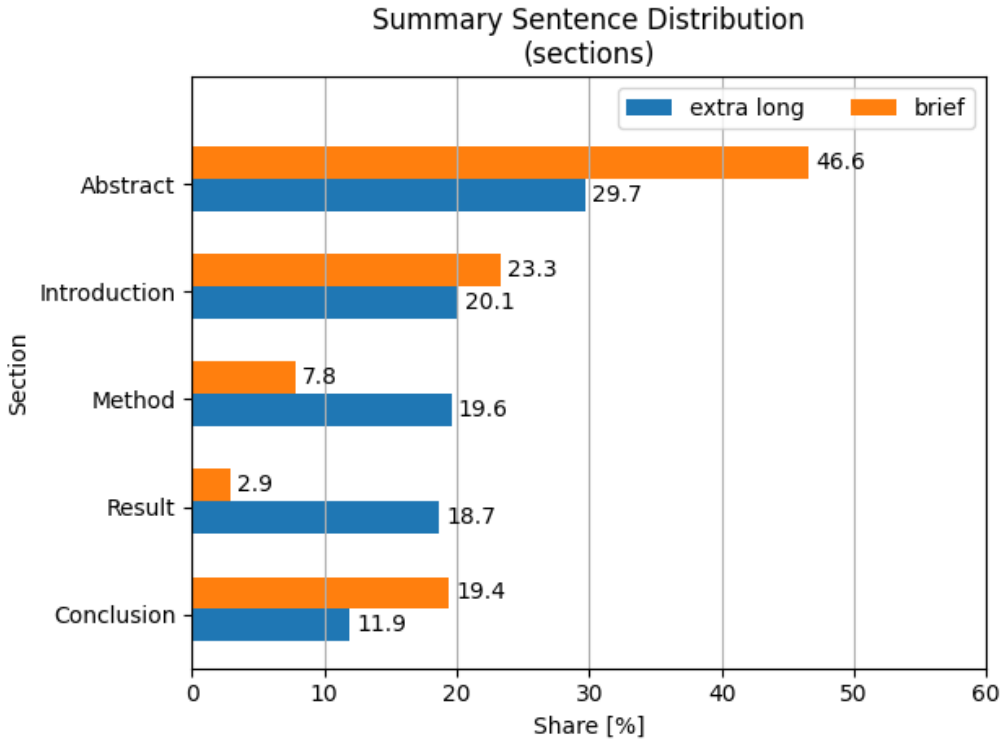
A high-quality dataset like OpenReview is essential for efficiently training models, and the newly sourced dataset meets this need with its human-composed, multi-target aspect summaries, categorized by length for guiding training. The SLED model, an efficient long-document transformer, excels in length- and aspect-dependent summarization tasks, outperforming the BART model in ROUGE and BERTScore metrics, demonstrating the effectiveness of considering entire documents. Simple heuristics like using paper abstracts fall short, as both the similarity search approach and SLED model yield higher BERTScores, making them more comparable to expert-written recaps. The similarity search method, based on SLED summaries, also outperforms the extractive TextRank baseline, enhancing accuracy and trustability. Guiding signals like length and aspect tokens improve coverage and performance, emphasizing the importance of controllability. Despite its smaller size, the SLED model offers performance and text quality comparable to larger LLMs, making it a viable option for low-resource systems. However, for certain tasks, larger models like $\text{BART}_{\text{large}}$ may achieve better

performance, although this comes with increased computational costs. For longer summaries, incorporating more information from input articles benefits performance, with SLED again demonstrating superior results compared to BART.

In conclusion, the experiments showed that small efficient long-document transformer models based on the fusion-in-decoder approach are powerful in some dimensions and can, with proper training data, generate high-quality output comparable to sophisticated models.



a) Summary sentence distribution over text chunks.



b) Summary sentence distribution over sections.

Figure 5.2.: Summary sentence distribution over text.

Method	Text
Paper abstract	We present Brax, an open source library for rigid body simulation with a focus on performance and parallelism on accelerators, written in JAX. We present results on a suite of tasks inspired by the existing reinforcement learning literature, but remade in our engine. Additionally, we provide reimplementations of PPO, SAC, ES, and direct policy optimization in JAX that compile alongside our environments, allowing the learning algorithm and the environment processing to occur on the same device, and to scale seamlessly on accelerators. Finally, we include notebooks that facilitate training of performant policies on common OpenAI Gym MuJoCo-like tasks in minutes.
Human-written reference	This paper proposes a new physics engine for regid body simulation that **1)** is differentiable; **2)** can run environment simulation and RL algorithm optimization on the same accelerator core without data marshelling or transfering. It re-implements three **mujoco** locomotion tasks and two additional manipulation **&** locomotion tasks to demonstrate the correctness and utility of the engine. It further benchmarks **4** algorithms on the proposed task and compares the engine with the traditional **mujoco** physics engine to validate it. The proposed differentiable physics engine, with further development, can be very useful for tackling robotics control tasks like locomotion and manipulation.
BART	This paper introduces Brax, a differentiable physics engine for rigid body simulation. Brax is written in JAX, and provides a suite of tasks, including ant, fetch, grasp, halfcheetah, and humanoid. The authors also provide a set of notebooks that facilitate training of performant policies on common OpenAI Gym **MuJoColike** tasks in minutes. The paper describes the design of Brax and provides some initial demonstrations on the benchmarking tasks based on the proposed library. ****Update after rebuttal**** **I have increased my score from 5 to 7.**
SLED	This paper proposes a differentiable physics engine for rigid body simulation, Brax. Brax is an open source library written in JAX, and it enables the simulation of simple rigid body physics systems in thousands of independent environments across hundreds of connected accelerators. The authors present results on a suite of tasks inspired by the existing reinforcement learning literature, but remade in **our** engine. Additionally, the authors provide reimplementations of PPO, SAC, ES, and direct policy optimization in **j**AX that compile alongside **our** environments, allowing the learning algorithm and the environment processing to occur on the same device, and to scale seamlessly on accelerators, which are also available.
GPT	Brax is an open-source library for rigid body simulation that focuses on performance and parallelism on accelerators. It is written in JAX and provides implementations of popular reinforcement learning algorithms such as PPO, SAC, ES, and direct policy optimization. Brax allows the learning algorithm and the environment processing to occur on the same device, scaling seamlessly on accelerators. The library includes pre-built environments and provides notebooks for training performant policies on common OpenAI Gym MuJoCo-like tasks. Brax achieves fast policy learning by leveraging auto-vectorization, device parallelism, just-in-time compilation, and auto-differentiation primitives of JAX.
Similarity Search	Brax - A Differentiable Physics Engine for Large Scale Rigid Body Simulation Abstract We present Brax, an open source library for rigid body simulation with a focus on performance and parallelism on accelerators, written in JAX. On the simulation side, practitioners have the choice of a variety of engine backends to power their study of simulated environments, including MuJoCo [10], pybullet [15], and physX, among many others, many of which are differentiable [16][17][18][19][20][21][22]14]. MuJoCo Gym-Likes The reinforcement learning and control communities have used the OpenAI Gym MuJoCo tasks as benchmarks for developing algorithms for the past several years. Additionally, we provide reimplementations of PPO, SAC, ES, and direct policy optimization in JAX that compile alongside our environments, allowing the learning algorithm and the environment processing to occur on the same device, and to scale seamlessly on accelerators. Mastery over the control of robots represents a society-transforming opportunity, thus we hope our engine only helps to improve and accelerate the equitable automation of our future.
TextRank	6, we catalog scaling behavior of Brax on accelerators, performance comparisons between Brax and MuJoCo on OpenAI Gym-style learning problems, and report Brax's engine fidelity on a benchmarking task. Qualitatively, for a fixed set of SAC hyperparameters, Brax environments achieve similar reward in a similar number of environment steps (right) MuJoCo-HalfCheetah-v2 and brax-halfcheetah. To illustrate the versatility of Brax as an engine, we include and solve several example environments in our initial release: MuJoCo-likes (Ant, Humanoid, Halfcheetah), Grasp (a dexterous manipulation environment), and Fetch (a goal-based locomotion environment).

Table 5.8.: Comparison of summary texts of BRAX (Freeman et al., 2021) generated by different methods. Unusual style, mistakes, and hallucinations are highlighted in blue.

Method	Text
Paper abstract	Despite the prevalence of recent success in learning from static graphs, learning from time-evolving graphs remains an open challenge. In this work, we design new, more stringent evaluation procedures for link prediction specific to dynamic graphs, which reflect real-world considerations, to better compare the strengths and weaknesses of methods. First, we create two visualization techniques to understand the reoccurring patterns of edges over time and show that many edges reoccur at later time steps. Based on this observation, we propose a pure memorization-based baseline called EdgeBank. EdgeBank achieves surprisingly strong performance across multiple settings which highlights that the negative edges used in the current evaluation are easy. To sample more challenging negative edges, we introduce two novel negative sampling strategies that improve robustness and better match real-world applications. Lastly, we introduce six new dynamic graph datasets from a diverse set of domains missing from current benchmarks, providing new challenges and opportunities for future research. Our code repository is accessible at https://github.com/fpour/DGB.git.
Human-written reference	In this work, the authors point out problems with the evaluation of dynamic link prediction approaches. As approaches often achieve near-perfect results, the authors argue that **a)** datasets are too simple and **b)** current sampling techniques generate negative edges that lead to less differentiation of approaches. The authors offer novel tools for better evaluation of dynamic link prediction and advise others to use them: **a)** They introduce new more robust negative sampling techniques, **b)** strong baseline method for comparing approaches against, **c)** **5 novel dynamic graph datasets** and two new techniques for generating datasets analysis plots.
BART	This paper studies the problem of dynamic link prediction. Specifically, the authors propose a new approach to better evaluate the performance of dynamic graph learning methods. To this end, they create two visualization techniques to understand the reoccurring patterns of edges over time and show that many edges reoccur at later time steps. Based on this observation, they propose a pure memorization-based baseline called EdgeBank. To sample more challenging negative edges, they introduce two novel negative sampling strategies that improve robustness and better match real-world applications. They also introduce six new dynamic graph datasets from a diverse set of domains missing from existing benchmarks.
SLED	This paper proposes a new benchmark for dynamic link prediction. Specifically, six dynamic graph datasets have been proposed. The benchmark focuses on the reoccurring patterns of edges over time and shows that many edges reoccur at later time steps. Based on this observation, the authors propose a pure memorization-based baseline called EdgeBank. EdgeBank achieves surprisingly strong performance across multiple settings which highlights that the negative edges used in the current evaluation are easy. The authors also introduce two novel negative sampling strategies that improve robustness and better match real-world applications. Lastly, EdgeBank introduces six new dynamic graphs datasets from a diverse set of domains, providing new challenges and opportunities for future research.
GPT	In this study, the authors propose new evaluation procedures for link prediction in dynamic graphs. They introduce two visualization techniques to understand the patterns of edges over time and propose a baseline called EdgeBank, which achieves strong performance by memorizing previously observed edges. They also introduce two novel negative sampling strategies to improve evaluation and propose six new dynamic graph datasets from diverse domains. The authors compare the performance of various methods using different evaluation strategies and show that the ranking of methods varies significantly. They highlight the limitations of the current evaluation procedures and datasets and provide solutions for more robust and effective evaluation.
Similarity Search	In this paper, we proposed tools to improve the evaluation of dynamic link prediction. We present six novel dynamic graph datasets from various domains such as politics, transportation, and economics. 5c shows the performance of different methods with the historical NS and inductive NS strategies, respectively. Therefore, it is important to evaluate against different sets of negative edges to better understand the performance of different models. To sample more challenging negative edges, we introduce two novel negative sampling strategies that improve robustness and better match real-world applications. Our code repository is accessible at https://github.com/fpour/DGB.git.
TextRank	We introduce two novel Negative Sampling (NS) strategies designed specifically for dynamic graphs which consider more difficult negative edges based on the reocurrence pattern of observed edges. In a dynamic network, the edges that have been never observed during previous timestamps can be considered as easy negative edges, since it is less likely that these edges occur during the test phase given the reoccurring pattern of dynamic graphs. Negative Sampling (NS) of Edges in Graphs.

Table 5.9.: Comparison of summary texts of EdgeBank (Poursafaei et al., 2022) generated by different methods. Unusual style, mistakes, and hallucinations are highlighted in blue.

Length Signal	Generated Text
brief	This paper proposes a differentiable physics engine for rigid body simulation, Brax. It contains a suite of tasks inspired by PPO, SAC, ES, and direct policy optimization in JAX that compile alongside our environments, allowing the learning algorithm and the environment processing to occur on the same device, and to scale seamlessly on accelerators.
short	This paper proposes a differentiable physics engine for Rigid Body Simulation, named Brax. Brax is an open source library for rigid body simulation with a focus on performance and parallelism on accelerators, written in JAX. Additionally, this paper provides reimplementations of PPO, SAC, ES, and direct policy optimization **inJAX** that compile alongside our environments, allowing the learning algorithm and the environment processing to occur on the same device. Finally, this work provides notebooks that facilitate training of performant policies on common OpenAI Gym MuJoCo tasks in minutes.
long	This paper proposes a differentiable physics engine for rigid body simulation, Brax. Brax is an open source library written in JAX, and it enables the simulation of simple rigid body physics systems in thousands of independent environments across hundreds of connected accelerators. The authors present results on a suite of tasks inspired by the existing reinforcement learning literature, but remade in **our** engine. Additionally, the authors provide reimplementations of PPO, SAC, ES, and direct policy optimization in **j**AX that compile alongside **our** environments, allowing the learning algorithm and the environment processing to occur on the same device, and to scale seamlessly on accelerators, which are also available.
extra long	This paper introduces a differentiable physics engine called Brax. The engine is built in JAX and can run on a large number of connected accelerators. In addition, the engine can be used to run a variety of reinforcement learning algorithms (e.g., PPO, SAC, ES) and direct policy optimization **(APG)**. The engine can run in many differentiable environments, allowing the learning algorithm and the environment processing to occur on the same device, and is able to scale seamlessly on accelerators, allowing training of performant policies on common OpenAI Gym MuJoCo tasks in minutes. The main contributions of this paper are: 1. Proximal Policy Optimization (PPO) 2. A set of tasks inspired by PPO and SAC. 3. The use of the engine for training. 4. The implementation of the environment differentiability. 5. The authors demonstrate the effectiveness of their engine by running some of PPO's environments and learning algorithms on them.

Table 5.10.: Comparison of summaries of BRAX (Freeman et al., 2021) with different lengths. Unusual style, mistakes, and hallucinations are highlighted in blue.

Aspect	Text
Purpose	The purpose of this paper is to develop and validate a model that can be used by network companies to create competitive advantages. The model is tested on a sample of 1,000 online users. The data are analyzed using structural equation modeling.
Value	This study aims to develop a scale for the measurement of the freemium business model, which is based on the development of a questionnaire and are related measurement validation. This study also aims to examine the internal quality of the model by using exploratory factor analysis.
Methods	This study aims to develop a scale for the measurement of the freemium business model, which is based on the theory of business model innovation and practical observations. It also aims to examine the validity of the model by using exploratory factor analysis.
Findings	The data were collected through a questionnaire survey from 1,000 online users. The data were analyzed using exploratory factor analysis and confirmatory factor analyses to determine the reliability and validity of the model. The results of this study were compared to the previous studies.

Table 5.11.: Comparison of summaries on different aspects. For the article of H.-C. Huang (2016) summaries on different aspects were generated.

6. Lessons Learned

Several lessons have been learned from this project, covering topics such as model training, dataset processing, and natural language application evaluation. These points are categorized into literature, implementation, and experiments.

6.1. Literature

Efficiently training large language models is a complex task. As models increase in size, it becomes crucial to find ways to apply them efficiently without sacrificing performance. Currently, many different approaches are being studied, but most methods and techniques are relatively new and still under research. Transformer-based models are the most popular and state-of-the-art. However, transformers also come with some drawbacks such as hallucination and high computational costs. For the latter, techniques such as efficient attention layers and fusion-in-decoder methods have been proposed to enhance training and inference. While these methods perform well for short documents, there is less research focused on the long-document domain.

Datasets are crucial for the effective training of models. There are various types of datasets for text summarization, categorized into single- and multi-document, as well as short- and long-document datasets. The domain is also important when training for a specific task. High-quality scientific long-document datasets in the technical field are sparse, and some popular ones contain noise. The need for high-grade data collections in the scientific field for training and testing models are essential.

Evaluation metrics measure how effectively a model performs, making well-suited measurements essential for efficient training and testing. Performance metrics can generally be categorized into lexical and semantic matching metrics, with lexical-based metrics being more common in text summarization tasks. While human evaluation is the most accurate measurement, it is resource-intensive and requires human annotations. Therefore, automatic evaluation metrics are necessary for efficient assessment. Hallucinations and factual inconsistencies are common issues among transformer models, leading to the development of several evaluation metrics designed to measure factuality. These metrics can be categorized into entailment

classification, question-answering (QA), and fact-based approaches. Additional research is needed to enhance the resistance of transformer models to hallucinations and improve their trustworthiness, along with developing metrics to measure these improvements.

6.2. Implementation

Finding suitable datasets can be challenging in some cases. Therefore, it is important to be able to define and create your own datasets that meet specific requirements. A new dataset was scraped from an open peer-review website, demonstrating that high-quality data collections can be created and sourced from the internet with sufficient effort.

During model training and testing, it is crucial to choose the appropriate evaluation methods and metrics. Depending on a specific aspect, such as performance or factuality, one metric might fit better than another. Therefore, a good metric can help you reach the desired goal faster. Additionally, the metric scores show the efficacy of the models and are easily comparable with other methods and techniques. However, there are many approaches to measurements, all with different advantages and drawbacks. It is difficult to measure performance using only one metric. For an insightful evaluation, a mix of several evaluators is needed, each assessing another dimension.

Hardware is fundamental when working with transformer models and large language models (LLMs). Even training and testing smaller models require specific hardware to ensure efficient and fast processing. A good setup is essential for achieving this.

Numerous frameworks are available for natural language processing tasks, many within the Python ecosystem. Python libraries facilitate the efficient and straightforward creation of pipelines for these tasks, making Python the preferred programming language for this project implementation.

6.3. Experiments

The source and quality of the corpus play an essential role in model training. The higher the quality, the better the results of the trained model in terms of performance and text quality, such as consistency and relevance. However, examples of human-written summaries showed that even experts make small mistakes and include inaccuracies. Moreover, opinions by the experts in the form of comments are appended at the end of some summaries. Models trained on these data might produce similar output and therefore hallucinative content.

Using transformer models also comes with some disadvantages. A major drawback is the lack of explainability of the attention mechanism. However,

explainability and trustibility are crucial as transformers tend to generate content containing hallucinations. Therefore, the output of the transformer models should always be checked and validated. Including explainable systems efficiently in transformer-based LLMs is still under investigation and a huge topic in research. A similarity search approach applied in this project helped to increase the trustability of the model.

Measuring the quality and readability of text is a challenging task. While many formulas have been suggested, most do not encompass all aspects of text quality. Recently, neural network-based evaluators have been proposed to assess multiple dimensions. Although these models do not yet match the accuracy of human evaluators, they provide a good approximation and are suitable for automatic text quality assessment.

7. Conclusion and Future Work

This chapter presents the insights gained from this work, discusses potential improvements to the system, and outlines perspectives for future research.

7.1. Conclusion

In conclusion, efficient long-document transformer models, such as SLED, are beneficial for generating high-quality summaries of scientific articles. The SLED model can cover the main information effectively and incorporate also long-term dependencies.

The experiments show that with increasing summary length, more information is extracted from the results section at the middle and end of the scientific papers. Consequently, taking the whole scientific article into consideration increases the performance and quality of the recap, especially in longer summary texts. However, for brief summaries, it is often sufficient to consider only the abstract of the paper and parts of the introduction.

In addition, training with extra guiding signals, such as length and aspect attributes, can increase the performance of the model and give the possibility to adjust the output with respect to the user's needs. An additional length signal allows the model to generate summaries with different lengths and richness of information. Consequently, controllability is beneficial for the summarization task and helps adapt the summary scope.

High-quality training data are necessary to effectively train transformer models. The newly sourced corpus Openreview is comparable to other long-document datasets in the scientific field. Additionally, the corpus has unique characteristics such as target summaries written by human experts and multi-targeting. The experiments showed that LLMs trained on this dataset produce high-quality summary texts. Therefore, this corpus is very suitable for the summarization task of scientific articles.

Evaluations regarding text quality and readability suggest that models with a low number of parameters can achieve performance comparable to that of state-of-the-art LLMs, which are much larger in size. Therefore, efficient transformer models can be a great alternative in low-resource environments, as they show low computational costs and have a good trade-off between performance and memory usage.

Furthermore, the extractive summary based on the similarity search of

the abstractive recap (referred as Sim. Search) demonstrate higher performance compared to traditional extractive method baselines. In addition, the trustability of the system can be increased as the user can trace the summary sentences in the input text.

7.2. Future Work

For future work, exploring promising models that leverage alternative efficient techniques presents a significant opportunity. This study focused on a specific transformer model based on the fusion-in-decoder approach, namely SLED. Expanding the investigation to include other optimization methods, such as sparse attention matrices, could potentially lead to further improvements in performance. Sparse attention matrices, in particular, have shown promise in reducing computational complexity while maintaining model efficacy, and their integration into transformer architectures warrants further exploration.

Additionally, a deeper examination of the impact of hallucinations in transformer models is crucial. In scientific contexts, it is crucial to guarantee the precision and dependability of the outputs produced. Since the attention mechanisms in transformer-based models are not easily interpretable, enhancing the traceability of summary sentences is essential. This improvement would contribute to greater explainability and trustworthiness of the models, which is critical for applications requiring rigorous validation of information.

Moreover, the current study exclusively considered text for summarization purposes. However, significant information is often contained within tables and figures, which are also essential to comprehensive understanding. Future research could explore methods for incorporating these elements into summaries, potentially creating more holistic and informative outputs.

Lastly, the aspect of controllability in transformer models could be further developed. With the growing prevalence of prompt-based large language models, there is an opportunity to design question-answer applications that are tailored specifically to scientific inquiries. Such models could enhance the precision and relevance of responses, offering valuable tools for research and knowledge dissemination. This focus on controllability and tailored responses could significantly advance the utility and adaptability of language models in various scientific domains.

Bibliography

Ainslie, J., Lei, T., de Jong, M., Ontanon, S., Brahma, S., Zemlyanskiy, Y., Uthus, D., Guo, M., Lee-Thorp, J., Tay, Y., Sung, Y.-H., & Sanghai, S. (2023). CoLT5: Faster long-range transformers with conditional computation. In H. Bouamor, J. Pino, & K. Bali (Eds.), *Proceedings of the 2023 conference on empirical methods in natural language processing* (pp. 5085–5100). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.emnlp-main.309. (Cit. on p. 43)

Ammar, W., Groeneveld, D., Bhagavatula, C., Beltagy, I., Crawford, M., Downey, D., Dunkelberger, J., Elgohary, A., Feldman, S., Ha, V., Kinney, R., Kohlmeier, S., Lo, K., Murray, T., Ooi, H.-H., Peters, M., Power, J., Skjonsberg, S., Wang, L. L., ... Etzioni, O. (2018). Construction of the literature graph in semantic scholar. In S. Bangalore, J. Chu-Carroll, & Y. Li (Eds.), *Proceedings of the 2018 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 3 (industry papers)* (pp. 84–91). Association for Computational Linguistics. https://doi.org/10.18653/v1/N18-3011. (Cit. on p. 44)

An, C., Zhong, M., Chen, Y., Wang, D., Qiu, X., & Huang, X. (2021). Enhancing scientific papers summarization with citation graph. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(14), 12498–12506. https://doi.org/10.1609/aaai.v35i14.17482 (cit. on p. 42)

Aralikatte, R., Narayan, S., Maynez, J., Rothe, S., & McDonald, R. (2021). Focus attention: Promoting faithfulness and diversity in summarization. *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)*, 6078–6095. https://doi.org/10.18653/v1/2021.acl-long.474 (cit. on p. 24)

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate [3rd International Conference on Learning Representations, ICLR 2015 ; Conference date: 07-05-2015 Through 09-05-2015, San Diego, United States] (cit. on p. 20).

Banerjee, S., & Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, 65–72. https://aclanthology.org/W05-0909 (cit. on p. 32)

Barzilay, R., & McKeown, K. R. (2019). Sentence fusion for multidocument news summarization. *Computational Linguistics*, *31*(3), 297–328 (cit. on p. 8).

Baxendale, P. B. (1958). Machine-made index for technical literature—an experiment. *IBM Journal of Research and Development*, *2*(4), 354–361. https://doi.org/10.1147/rd.24.0354 (cit. on p. 7)

Beltagy, I., Lo, K., & Cohan, A. (2019). SciBERT: A pretrained language model for scientific text. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 3615–3620. https://doi.org/10.18653/v1/D19-1371 (cit. on p. 55)

Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document transformer. (Cit. on pp. 29, 43).

Bornmann, L., Haunschild, R., & Mutz, R. (2021). Growth rates of modern science: A latent piecewise growth curve approach to model publication numbers from established and new literature databases. humanities and social sciences communications. *Humanities and Social Sciences Communications*, *8*(224). https://doi.org/10.1057/s41599-021-00903-w (cit. on p. 1)

Brandow, R., Mitze, K., & Rau, L. F. (1995). Automatic condensation of electronic publications by sentence selection. *Information Processing & Management*, *31*(5), 675–685. https://doi.org/10.1016/0306-4573(95)00052-I (cit. on p. 7)

Brody, S., Alon, U., & Yahav, E. (2021). How attentive are graph attention networks? https://doi.org/10.48550/ARXIV.2105.14491. (Cit. on p. 17)

Cachola, I., Lo, K., Cohan, A., & Weld, D. (2020). TLDR: Extreme summarization of scientific documents. *Findings of the Association for Computational Linguistics: EMNLP 2020*, 4766–4777. https://doi.org/10.18653/v1/2020.findings-emnlp.428 (cit. on pp. 40, 53, 67–70)

Campos, R., Mangaravite, V., Pasquali, A., Jorge, A., Nunes, C., & Jatowt, A. (2020). Yake! keyword extraction from single documents using multiple local features. *Information Sciences*, *509*, 257–289. https://doi.org/https://doi.org/10.1016/j.ins.2019.09.013 (cit. on pp. 44, 45)

Cao, S., & Wang, L. (2021). CLIFF: Contrastive learning for improving faithfulness and factuality in abstractive summarization. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 6633–6649. https://doi.org/10.18653/v1/2021.emnlp-main.532 (cit. on p. 24)

Chen, S., Zhang, F., Sone, K., & Roth, D. (2021). Improving faithfulness in abstractive summarization with contrast candidate generation and selection. *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language*

Technologies, 5935–5941. https://doi.org/10.18653/v1/2021.naacl-main.475 (cit. on p. 25)

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734. https://doi.org/10.3115/v1/D14-1179 (cit. on p. 14)

Cohan, A., Dernoncourt, F., Kim, D. S., Bui, T., Kim, S., Chang, W., & Goharian, N. (2018). A discourse-aware attention model for abstractive summarization of long documents. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 615–621. https://doi.org/10.18653/v1/N18-2097 (cit. on pp. 38–40, 53, 67–70)

Coleman, M., & Liau, T. L. (1975). A Computer Readability Formula Designed for Machine Scoring. *Journal of Applied Psychology*, *60*, 283–284. https://doi.org/https://doi.org/10.1037/h0076540 (cit. on p. 36)

Dale, E., & Chall, J. S. (1949). The Concept of Readability. *Elementary English*, *26*(1), 19–26 (cit. on p. 36).

Dasigi, P., Lo, K., Beltagy, I., Cohan, A., Smith, N. A., & Gardner, M. (2021). A dataset of information-seeking questions and answers anchored in research papers. *ArXiv*, *abs/2105.03011*. https://api.semanticscholar.org/CorpusID:234093776 (cit. on pp. 40, 53)

de Santana Correia, A., & Colombini, E. L. (2022). Attention, please! a survey of neural attention models in deep learning. *Artificial Intelligence Review*, *55*(8), 6037–6124. https://doi.org/10.1007/s10462-022-10148-x (cit. on pp. 18–20, 25, 26)

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. https://doi.org/10.18653/v1/N19-1423 (cit. on pp. 8, 10, 23, 34, 45)

DeYoung, J., Beltagy, I., van Zuylen, M., Kuehl, B., & Wang, L. L. (2021). MS2: Multi-document summarization of medical studies. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 7494–7513. https://doi.org/10.18653/v1/2021.emnlp-main.594 (cit. on p. 42)

Durmus, E., He, H., & Diab, M. (2020). FEQA: A question answering evaluation framework for faithfulness assessment in abstractive summarization. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 5055–5070. https://doi.org/10.18653/v1/2020.acl-main.454 (cit. on p. 34)

Edmundson, H. P. (1969). New methods in automatic extracting. *Journal of the ACM*, *16*(2), 264–285. https://doi.org/10.1145/321510.321519 (cit. on p. 7)

Falke, T., Ribeiro, L. F. R., Utama, P. A., Dagan, I., & Gurevych, I. (2019). Ranking generated summaries by correctness: An interesting but challenging application for natural language inference. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2214–2220. https://doi.org/10.18653/v1/P19-1213 (cit. on p. 33)

Feng, L. (2010). *Automatic readability assessment* (Doctoral dissertation) [Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Zuletzt aktualisiert - 2023-03-03]. https://www.proquest.com/dissertations-theses/automatic-readability-assessment/docview/792432882/se-2. (Cit. on pp. 35, 36)

Flesch, R. (1979). *How to write plain English*. Harper and Row. (Cit. on p. 36).

Freeman, C. D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., & Bachem, O. (2021). Brax – a differentiable physics engine for large scale rigid body simulation. (Cit. on pp. 78, 79, 87, 89).

Gao, Y., Zhao, W., & Eger, S. (2020). SUPERT: Towards new frontiers in unsupervised evaluation metrics for multi-document summarization. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 1347–1354. https://doi.org/10.18653/v1/2020.acl-main.124 (cit. on p. 32)

Ghadimi, A., & Beigy, H. (2022). Hybrid multi-document summarization using pre-trained language models. *Expert Systems with Applications*, *192*, 116292. https://doi.org/10.1016/j.eswa.2021.116292 (cit. on p. 9)

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning* [http://www.deeplearningbook.org]. MIT Press. (Cit. on pp. 13–15).

Goodrich, B., Rao, V., Liu, P. J., & Saleh, M. (2019). Assessing the factual accuracy of generated text. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 166–175. https://doi.org/10.1145/3292500.3330955 (cit. on p. 35)

Grobid. (2008–2023). (Cit. on pp. 44, 52, 56, 57, 64).

Grootendorst, M. (2020). Keybert: Minimal keyword extraction with bert. https://doi.org/10.5281/zenodo.4461265. (Cit. on pp. 45, 55–57, 64)

Gu, J., Lu, Z., Li, H., & Li, V. O. (2016). Incorporating copying mechanism in sequence-to-sequence learning. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1631–1640. https://doi.org/10.18653/v1/P16-1154 (cit. on p. 22)

Gunning, R. (1952). *The Technique of Clear Writing*. McGraw-Hill. (Cit. on p. 36).

Guo, M., Ainslie, J., Uthus, D., Ontanon, S., Ni, J., Sung, Y.-H., & Yang, Y. (2022). LongT5: Efficient text-to-text transformer for long sequences.

Findings of the Association for Computational Linguistics: NAACL 2022, 724–736. https://doi.org/10.18653/v1/2022.findings-naacl.55 (cit. on pp. 9–11, 20, 21, 28, 43)

He, J., Kryscinski, W., McCann, B., Rajani, N., & Xiong, C. (2022). CTRLsum: Towards generic controllable text summarization. *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 5879–5915. https://doi.org/10.18653/v1/2022.emnlp-main.396 (cit. on pp. 30, 31)

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735 (cit. on p. 14)

Hovy, E., & Lin, C.-Y. (1998). Automated Text Summarization and the Summarist System. *TIPSTER TEXT PROGRAM PHASE III: Proceedings of a Workshop held at Baltimore, Maryland, October 13-15, 1998*, 197–214. https://doi.org/10.3115/1119089.1119121 (cit. on pp. 4–6)

Huang, H.-C. (2016). Freemium business model: Construct development and measurement validation. *Internet Research*, *26*(3), 604–625. https://doi.org/10.1108/IntR-03-2014-0064 (cit. on pp. 81, 90)

Huang, L., Cao, S., Parulian, N., Ji, H., & Wang, L. (2021). Efficient attentions for long document summarization. *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1419–1436. https://doi.org/10.18653/v1/2021.naacl-main.112 (cit. on pp. 40, 41, 62, 67, 69, 70)

Ibrahim Altmami, N., & El Bachir Menai, M. (2022). Automatic summarization of scientific articles: A survey. *Journal of King Saud University - Computer and Information Sciences*, *34*(4), 1011–1028. https://doi.org/10.1016/j.jksuci.2020.04.020 (cit. on p. 11)

Ivgi, M., Shaham, U., & Berant, J. (2023). Efficient Long-Text Understanding with Short-Text Models. *Transactions of the Association for Computational Linguistics*, *11*, 284–299. https://doi.org/10.1162/tacl_a_00547 (cit. on pp. 2, 3, 30, 43, 52, 56, 57, 60–62, 64, 66, 67, 70–73, 75, 77, 78, 81, 82)

Izacard, G., & Grave, E. (2021). Leveraging passage retrieval with generative models for open domain question answering. *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 874–880. https://doi.org/10.18653/v1/2021.eacl-main.74 (cit. on p. 30)

Jelinek, F., Mercer, R. L., Bahl, L. R., & Baker, J. K. (1977). Perplexity - a measure of the difficulty of speech recognition tasks. *Journal of the Acoustical Society of America*, 63–63. https://doi.org/10.1121/1.2016299 (cit. on p. 32)

Jeong, Y., & Kim, E. (2022). Scideberta: Learning deberta for science technology documents and fine-tuning information extraction tasks. *IEEE Ac-*

cess, *10*, 60805–60813. https://doi.org/10.1109/ACCESS.2022.3180830 (cit. on pp. 55, 58)

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y., Madotto, A., & Fung, P. (2022). Survey of hallucination in natural language generation [Just Accepted]. *ACM Comput. Surv.* https://doi.org/10.1145/3571730 (cit. on pp. 23–25, 33–35)

Kågebäck, M., Mogren, O., Tahmasebi, N., & Dubhashi, D. (2014). Extractive summarization using continuous vector space models. *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, 31–39. https://doi.org/10.3115/v1/W14-1504 (cit. on p. 8)

Klare, G. R. (1974). Assessing readability. *Reading Research Quarterly*, *10*(1), 62–102. Retrieved January 21, 2024, from http://www.jstor.org/stable/747086 (cit. on p. 36)

Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM*, *46*(5), 604–632. https://doi.org/10.1145/324133.324140 (cit. on p. 7)

Kobayashi, H., Noguchi, M., & Yatsuka, T. (2015). Summarization based on embedding distributions. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1984–1989. https://doi.org/10.18653/v1/D15-1232 (cit. on p. 8)

Koh, H. Y., Ju, J., Liu, M., & Pan, S. (2022). An empirical survey on long document summarization: Datasets, models, and metrics. *ACM Computing Surveys*, *55*(8). https://doi.org/10.1145/3545176 (cit. on pp. 1, 10, 11, 37–41, 68)

Kornilova, A., & Eidelman, V. (2019). BillSum: A corpus for automatic summarization of US legislation. *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, 48–56. https://doi.org/10.18653/v1/D19-5406 (cit. on pp. 40, 67, 70)

Kryscinski, W., McCann, B., Xiong, C., & Socher, R. (2020). Evaluating the factual consistency of abstractive text summarization. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 9332–9346. https://doi.org/10.18653/v1/2020.emnlp-main.750 (cit. on p. 34)

Kudo, T., & Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 66–71. https://doi.org/10.18653/v1/D18-2012 (cit. on p. 23)

Laban, P., Schnabel, T., Bennett, P. N., & Hearst, M. A. (2022). SummaC: Re-visiting NLI-based models for inconsistency detection in summarization. *Transactions of the Association for Computational Linguistics*, *10*, 163–177. https://doi.org/10.1162/tacl_a_00453 (cit. on p. 33)

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2020). BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7871–7880. https://doi.org/10.18653/v1/2020.acl-main.703 (cit. on pp. 9, 10, 23, 28, 34, 35, 61, 66, 67, 71–73, 75–77, 82)

Li, H., Einolghozati, A., Iyer, S., Paranjape, B., Mehdad, Y., Gupta, S., & Ghazvininejad, M. (2021). EASE: Extractive-abstractive summarization end-to-end using the information bottleneck principle. *Proceedings of the Third Workshop on New Frontiers in Summarization*, 85–95. https://doi.org/10.18653/v1/2021.newsum-1.10 (cit. on p. 26)

Li, H., Zhu, J., Zhang, J., & Zong, C. (2018). Ensure the correctness of the summary: Incorporate entailment knowledge into abstractive sentence summarization. *Proceedings of the 27th International Conference on Computational Linguistics*, 1430–1441. https://aclanthology.org/C18-1121 (cit. on p. 24)

Li, M., Qi, J., & Lau, J. H. (2022). Peersum: A peer review dataset for abstractive multi-document summarization. https://doi.org/10.48550/ARXIV.2203.01769. (Cit. on pp. 34, 35, 42)

Li, W., Xiao, X., Lyu, Y., & Wang, Y. (2018). Improving neural abstractive document summarization with structural regularization. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 4078–4087. https://doi.org/10.18653/v1/D18-1441 (cit. on p. 21)

Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. *Text Summarization Branches Out*, 74–81. https://aclanthology.org/W04-1013 (cit. on pp. 32, 62, 66, 67, 71, 75, 82)

Liu, Y., & Lapata, M. (2019). Text summarization with pretrained encoders. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 3730–3740. https://doi.org/10.18653/v1/D19-1387 (cit. on p. 8)

Louis, A., & Nenkova, A. (2013). Automatically assessing machine summary content without a gold standard. *Computational Linguistics*, *39*(2), 267–300. https://doi.org/10.1162/COLI_a_00123 (cit. on p. 32)

Lu, Y., Dong, Y., & Charlin, L. (2020). Multi-XScience: A large-scale dataset for extreme multi-document summarization of scientific articles. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 8068–8074. https://doi.org/10.18653/v1/2020.emnlp-main.648 (cit. on p. 42)

Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM Journal of Research and Development*, *2*(2), 159–165. https://doi.org/10.1147/rd.22.0159 (cit. on pp. 4, 6, 7)

Ma, C., Zhang, W. E., Guo, M., Wang, H., & Sheng, Q. Z. (2022). Multi-document summarization via deep learning techniques: A survey. *ACM Comput. Surv.*, *55*(5). https://doi.org/10.1145/3529754 (cit. on pp. 12, 13)

Marsi, E., & Krahmer, E. (2005). Explorations in sentence fusion. *Tenth European Workshop on Natural Language Generation* (cit. on p. 9).

McLaughlin, G. H. (1969). SMOG Grading-a New Readability Formula. *Journal of Reading*, *12*(8), 639–646 (cit. on p. 36).

Meng, R., Thaker, K., Zhang, L., Dong, Y., Yuan, X., Wang, T., & He, D. (2021). Bringing structure into summaries: A faceted summarization dataset for long scientific documents. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 1080–1089. https://doi.org/10.18653/v1/2021.acl-short.137 (cit. on pp. 30, 31, 40, 41, 53–56, 60, 64, 66–70, 73, 74, 78, 83)

Mihalcea, R., & Tarau, P. (2004). TextRank: Bringing order into text. *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 404–411. https://aclanthology.org/W04-3252 (cit. on pp. 7, 66, 67, 71, 75)

Moawad, I. F., & Aref, M. (2012). Semantic graph reduction approach for abstractive text summarization. *2012 Seventh International Conference on Computer Engineering & Systems (ICCES)*, 132–138. https://doi.org/10.1109/ICCES.2012.6408498 (cit. on p. 8)

Nan, F., Nallapati, R., Wang, Z., Nogueira dos Santos, C., Zhu, H., Zhang, D., McKeown, K., & Xiang, B. (2021). Entity-level factual consistency of abstractive text summarization. *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 2727–2733. https://doi.org/10.18653/v1/2021.eacl-main.235 (cit. on pp. 34, 35)

Napoles, C., Callison-Burch, C., Ganitkevitch, J., & Van Durme, B. (2011). Paraphrastic sentence compression with a character-based metric: Tightening without deletion. *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, 84–90. https://aclanthology.org/W11-1610 (cit. on p. 9)

Nenkova, A., Passonneau, R., & McKeown, K. (2007). The pyramid method: Incorporating human content selection variation in summarization evaluation. *ACM Trans. Speech Lang. Process.*, *4*(2), 4–es. https://doi.org/10.1145/1233912.1233913 (cit. on p. 32)

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). *The pagerank citation ranking: Bringing order to the web.* (Technical Report No. 1999-66)

[Previous number = SIDL-WP-1999-0120]. Stanford InfoLab. Stanford InfoLab. http://ilpubs.stanford.edu:8090/422/. (Cit. on p. 7)

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: A method for automatic evaluation of machine translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 311–318. https://doi.org/10.3115/1073083.1073135 (cit. on p. 32)

Phang, J., Zhao, Y., & Liu, P. J. (2022). Investigating efficiently extending transformers for long input summarization. https://doi.org/10.48550/ARXIV.2208.04347. (Cit. on pp. 9–11, 20, 21, 28)

Poursafaei, F., Huang, S., Pelrine, K., & Rabbany, R. (2022). Towards better evaluation for dynamic link prediction. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, & A. Oh (Eds.), *Advances in neural information processing systems* (pp. 32928–32941). Curran Associates, Inc. (Cit. on pp. 79, 80, 88).

Qi, S., Li, L., Li, Y., Jiang, J., Hu, D., Li, Y., Zhu, Y., Zhou, Y., Litvak, M., & Vanetik, N. (2022). SAPGraph: Structure-aware extractive summarization for scientific papers with heterogeneous graph. *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 575–586. https://aclanthology.org/2022.aacl-main.44 (cit. on p. 41)

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & 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. http://jmlr.org/papers/v21/20-074.html (cit. on pp. 9, 10, 23, 28, 61)

Rane, N., & Govilkar, S. (2019). Recent trends in deep learning based abstractive text summarization. *International Journal of Recent Technology and Engineering (IJRTE)*, *8*(3). https://doi.org/10.35940/ijrte.C4996.098319 (cit. on p. 8)

Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. (Cit. on pp. 55, 56, 58).

Rose, S., Engel, D., Cramer, N., & Cowley, W. (2010). Automatic keyword extraction from individual documents. https://doi.org/10.1002/9780470689646.ch1. (Cit. on p. 44)

Rush, A. M., Chopra, S., & Weston, J. (2015). A neural attention model for abstractive sentence summarization. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 379–389. https://doi.org/10.18653/v1/D15-1044 (cit. on pp. 9, 20)

Saha, S., Zhang, S., Hase, P., & Bansal, M. (2022). Summarization programs: Interpretable abstractive summarization with neural modular trees. https://doi.org/10.48550/ARXIV.2209.10492. (Cit. on p. 27)

Salchner, M. F., & Jatowt, A. (2022). A survey of automatic text summarization using graph neural networks. *Proceedings of the 29th International Conference on Computational Linguistics*, 6139–6150. https://aclanthology.org/2022.coling-1.536 (cit. on pp. 16–18)

Salton, G., & McGill, M. J. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill. (Cit. on pp. 7, 45).

Schopf, T., Klimek, S., & Matthes, F. (2022). PatternRank: Leveraging pretrained language models and part of speech for unsupervised keyphrase extraction. *Proceedings of the 14th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*. https://doi.org/10.5220/0011546600003335 (cit. on pp. 45, 46, 55)

Schuster, M., & Nakajima, K. (2012). Japanese and korean voice search. *International Conference on Acoustics, Speech and Signal Processing*, 5149–5152 (cit. on p. 23).

Scialom, T., Dray, P.-A., Lamprier, S., Piwowarski, B., Staiano, J., Wang, A., & Gallinari, P. (2021). QuestEval: Summarization asks for fact-based evaluation. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 6594–6604. https://doi.org/10.18653/v1/2021.emnlp-main.529 (cit. on p. 34)

See, A., Liu, P. J., & Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1073–1083. https://doi.org/10.18653/v1/P17-1099 (cit. on p. 22)

Sennrich, R., Haddow, B., & Birch, A. (2016). Neural machine translation of rare words with subword units. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1715–1725. https://doi.org/10.18653/v1/P16-1162 (cit. on p. 23)

Senter, R. J., & Smith, E. A. (1967). *Automated readability index* (tech. rep.). Cincinnati University. (Cit. on p. 36).

Shaham, U., Segal, E., Ivgi, M., Efrat, A., Yoran, O., Haviv, A., Gupta, A., Xiong, W., Geva, M., Berant, J., & Levy, O. (2022). SCROLLS: Standardized CompaRison over long language sequences. *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 12007–12021. https://aclanthology.org/2022.emnlp-main.823 (cit. on pp. 1, 42, 43, 61)

Sharma, E., Li, C., & Wang, L. (2019). BIGPATENT: A large-scale dataset for abstractive and coherent summarization. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2204–2213. https://doi.org/10.18653/v1/P19-1212 (cit. on pp. 40, 67, 69, 70)

Sharma, G., & Sharma, D. (2022). Automatic text summarization methods: A comprehensive review. *SN Computer Science*, *4*(1), 33. https://doi.org/10.1007/s42979-022-01446-w (cit. on pp. 4, 5, 9)

Shen, X., Zhao, Y., Su, H., & Klakow, D. (2019). Improving latent alignment in text summarization by generalizing the pointer generator. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 3762–3773. https://doi.org/10.18653/v1/D19-1390 (cit. on p. 22)

Shen, Z., Lo, K., Wang, L. L., Kuehl, B., Weld, D. S., & Downey, D. (2022). VILA: Improving Structured Content Extraction from Scientific PDFs Using Visual Layout Groups. *Transactions of the Association for Computational Linguistics*, *10*, 376–392. https://doi.org/10.1162/tacl_a_00466 (cit. on p. 43)

Singh, M., Cambronero, J., Gulwani, S., Le, V., Negreanu, C., & Verbruggen, G. (2023). Codefusion: A pre-trained diffusion model for code generation. https://doi.org/https://doi.org/10.48550/arXiv.2310.17680. (Cit. on pp. 3, 75)

Song, M., Feng, Y., & Jing, L. (2023). A survey on recent advances in keyphrase extraction from pre-trained language models. In A. Vlachos & I. Augenstein (Eds.), *Findings of the association for computational linguistics: Eacl 2023* (pp. 2153–2164). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.findings-eacl.161. (Cit. on pp. 44, 45)

Staar, P. W. J., Dolfi, M., Auer, C., & Bekas, C. (2018). Corpus conversion service: A machine learning platform to ingest documents at scale. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 774–782. https://doi.org/10.1145/3219819.3219834 (cit. on p. 44)

Tay, Y., Dehghani, M., Bahri, D., & Metzler, D. (2022). Efficient transformers: A survey. *ACM Comput. Surv.*, *55*(6). https://doi.org/10.1145/3530811 (cit. on pp. 1, 28, 42)

Tay, Y., Dehghani, M., Tran, V. Q., Garcia, X., Wei, J., Wang, X., Chung, H. W., Shakeri, S., Bahri, D., Schuster, T., Zheng, H. S., Zhou, D., Houlsby, N., & Metzler, D. (2023). Ul2: Unifying language learning paradigms. (Cit. on p. 43).

Treviso, M., Lee, J.-U., Ji, T., Aken, B. v., Cao, Q., Ciosici, M. R., Hassid, M., Heafield, K., Hooker, S., Raffel, C., Martins, P. H., Martins, A. F. T., Forde, J. Z., Milder, P., Simpson, E., Slonim, N., Dodge, J., Strubell, E., Balasubramanian, N., … Schwartz, R. (2023). Efficient Methods for Natural Language Processing: A Survey. *Transactions of the Association for Computational Linguistics*, *11*, 826–860. https://doi.org/10.1162/tacl_a_00577 (cit. on p. 30)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30* (cit. on pp. 17, 18, 20).

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. (Cit. on p. 17).

Wang, A., Cho, K., & Lewis, M. (2020). Asking and answering questions to evaluate the factual consistency of summaries. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 5008–5020. https://doi.org/10.18653/v1/2020.acl-main.450 (cit. on p. 34)

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(1), 4–24. https://doi.org/10.1109/TNNLS.2020.2978386 (cit. on pp. 17, 18)

Xiong, W., Gupta, A., Toshniwal, S., Mehdad, Y., & Yih, S. (2023). Adapting pretrained text-to-text models for long text sequences. In H. Bouamor, J. Pino, & K. Bali (Eds.), *Findings of the association for computational linguistics: Emnlp 2023* (pp. 5566–5578). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.findings-emnlp.370. (Cit. on pp. 9–11, 20, 21, 28, 43)

Yasunaga, M., Kasai, J., Zhang, R., Fabbri, A., Li, I., Friedman, D., & Radev, D. (2019). ScisummNet: A large annotated corpus and content-impact models for scientific paper summarization with citation networks. *Proceedings of AAAI 2019* (cit. on p. 42).

Yuan, W., Neubig, G., & Liu, P. (2021). Bartscore: Evaluating generated text as text generation. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, & J. W. Vaughan (Eds.), *Advances in neural information processing systems* (pp. 27263–27277). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2021/file/e4d2b6e6fdeca3e60e0f1a62fee3d9dd-Paper.pdf. (Cit. on p. 35)

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al. (2020). Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, *33* (cit. on p. 29).

Zajic, D., Dorr, B., Lin, J., Monz, C., & Schwartz, R. (2005). A sentence-trimming approach to multi-document summarization. *Proceedings of DUC 2005* (cit. on p. 9).

Zhang, J., Zhao, Y., Saleh, M., & Liu, P. J. (2020). Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. *Proceedings of the 37th International Conference on Machine Learning* (cit. on pp. 28, 33, 62, 66, 67, 71, 75, 82).

Zhang, M., Zhou, G., Yu, N., W.and Huang, & Liu, W. (2022). A comprehensive survey of abstractive text summarization based on deep

learning. *Computational Intelligence and Neuroscience*, *7132226*. https://doi.org/10.1155/2022/7132226 (cit. on pp. 8, 13, 14, 16, 18, 20, 21, 23)

Zhang, Y., Liu, Y., Yang, Z., Fang, Y., Chen, Y., Radev, D., Zhu, C., Zeng, M., & Zhang, R. (2023). Macsum: Controllable summarization with mixed attributes. *Transactions of the Association for Computational Linguistics* (cit. on pp. 30, 31).

Zhao, W., Peyrard, M., Liu, F., Gao, Y., Meyer, C. M., & Eger, S. (2019). MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 563–578. https://doi.org/10.18653/v1/D19-1053 (cit. on p. 33)

Zhao, Z., Cohen, S. B., & Webber, B. (2020). Reducing quantity hallucinations in abstractive summarization. *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2237–2249. https://doi.org/10.18653/v1/2020.findings-emnlp.203 (cit. on pp. 25, 26)

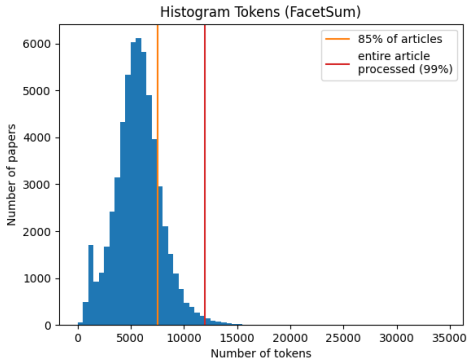
Zhong, M., Liu, Y., Yin, D., Mao, Y., Jiao, Y., Liu, P., Zhu, C., Ji, H., & Han, J. (2022). Towards a unified multi-dimensional evaluator for text generation. In Y. Goldberg, Z. Kozareva, & Y. Zhang (Eds.), *Proceedings of the 2022 conference on empirical methods in natural language processing* (pp. 2023–2038). Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.emnlp-main.131. (Cit. on pp. 34–36, 66)

Zhu, C., Hinthorn, W., Xu, R., Zeng, Q., Zeng, M., Huang, X., & Jiang, M. (2021). Enhancing factual consistency of abstractive summarization. *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 718–733. https://doi.org/10.18653/v1/2021.naacl-main.58 (cit. on pp. 24, 25)

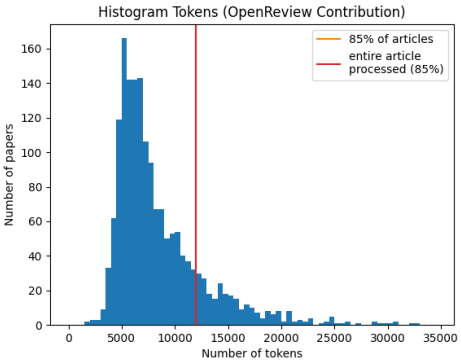
Appendix

Appendix A.

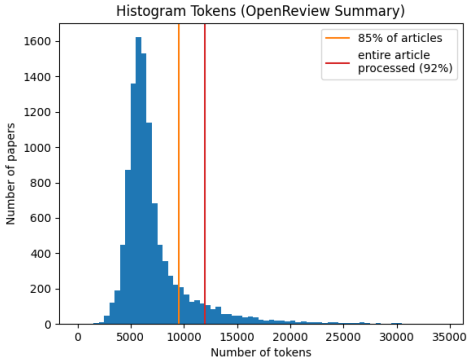
Experiments



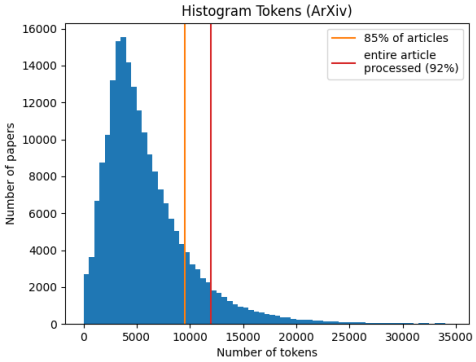
a) FacetSum tokens histogram.



b) OpenReview Contribution tokens histogram.



c) OpenReview Summary tokens histogram.

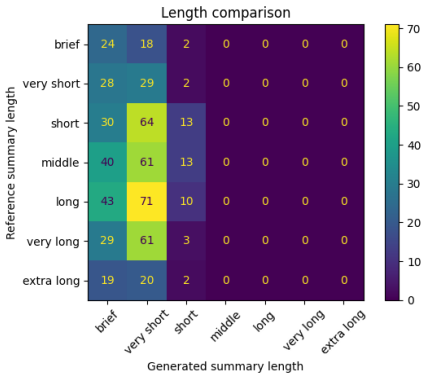


d) ArXiv tokens histogram.

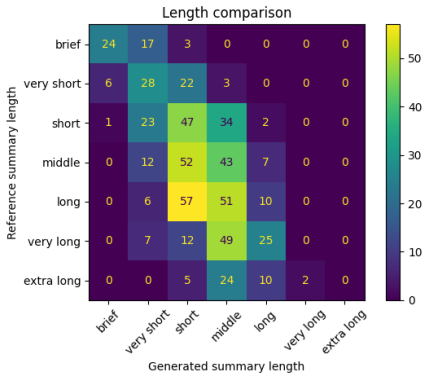
Figure A.1.: Token histograms of selected datasets. The red line indicates the maximum input of the SLED model that can be processed entirely with the given hardware limitations.

	Mean Absolute Deviation (MAD)			
steps	2,500	7,500	12,500	17,500
length penalty 1	2.83	1.51	1.34	0.49
length penalty 2	2.42	0.85	0.91	0.41

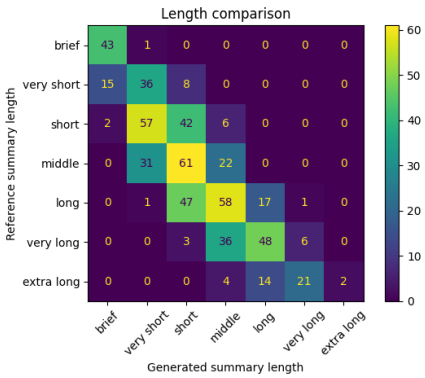
Table A.1.: Comparison of length penalty parameters of the BART model on the OpenReview Contribution dataset. Increased training steps lead to better length adaptation, measured by Mean Absolute Deviation (MAD).



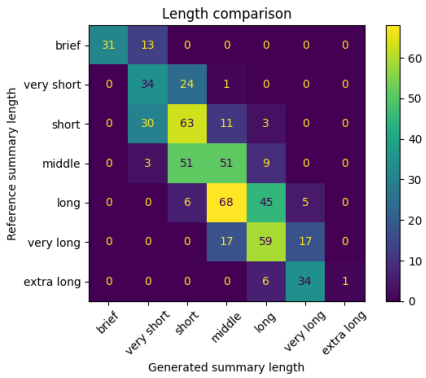
a) Length comparison after 2,500 steps.



b) Length comparison after 7,500 steps.



c) Length comparison after 12,500 steps.



d) Length comparison after 17,500 steps.

Figure A.2.: Length comparison during training. As step size increases, length adaptation improves, achieving a strong correlation.

Length	Metric	Paper abstr.	TextRank	Sim. Search	BART	SLED
brief	R1	18.75	19.45	29.25	33.77	35.75
	R2	5.87	3.89	8.83	11.17	12.90
	RL	12.86	14.24	21.22	25.53	28.01
	BS	0.162	0.141	0.248	0.323	0.335
very short	R1	26.29	24.97	34.56	40.27	40.54
	R2	8.39	5.39	9.91	13.42	13.64
	RL	16.79	16.84	22.96	27.97	28.08
	BS	0.227	0.144	0.261	0.328	0.331
short	R1	30.86	28.88	35.12	40.39	40.42
	R2	9.16	7.03	10.62	12.74	13.22
	RL	18.74	19.12	23.43	27.42	27.19
	BS	0.232	0.119	0.243	0.294	0.295
middle	R1	33.38	31.24	36.74	40.15	40.74
	R2	9.16	7.17	9.60	10.97	11.45
	RL	19.67	20.27	23.19	27.58	25.84
	BS	0.235	0.118	0.224	0.265	0.274
long	R1	37.23	31.66	37.65	41.65	41.82
	R2	9.47	5.90	9.07	10.96	11.20
	RL	21.09	20.09	24.17	27.58	27.12
	BS	0.235	0.130	0.226	0.265	0.272
very long	R1	38.41	30.70	37.83	41.55	42.32
	R2	9.14	6.28	8.98	10.51	10.88
	RL	23.08	21.49	24.49	27.81	27.69
	BS	0.217	0.125	0.196	0.228	0.237
extra long	R1	38.21	27.24	37.96	39.22	39.75
	R2	8.13	5.25	7.09	8.50	8.67
	RL	24.79	20.34	25.35	28.73	28.25
	BS	0.191	0.117	0.172	0.192	0.199

Table A.2.: Performance results depending on different summary lengths on the OpenReview Contribution dataset. Comparisons are based on the ROUGE1 (R1), ROUGE2 (R2), ROUGELSum (RL) and BERTScore (BS) metric.

Method	input source	length signal	ROUGE1	ROUGE2	ROUGELsum
heuristic	paper abstr.	-	32.99	9.63	19.98
TextRank	full paper	no	29.26	6.48	18.97
TextRank	full paper	yes	31.24	6.73	20.16
Sim. Search	summ.+paper	yes	36.80	9.87	23.57
$BART_{base}$	1K tokens	yes	37.93	11.88	34.04
$SLED_{base}$	12K tokens	yes	**38.22**	**11.98**	**34.22**

Table A.3.: Performance comparison on the OpenReview Summary dataset shows that SLED with $BART_{base}$ achieves the highest ROUGE scores.