Norwegian University of Life Sciences

Master's Thesis 2023 30 ECTS
Faculty of Science and Technology

Beyond Extractive: Advancing Abstractive Automatic Text Summarization in Norwegian with Transformers

Jørgen Navjord
Data Science
Jon-Mikkel R. Korsvik
Data Science

Abstract

Automatic summarization is a key area in natural language processing (NLP) and machine learning which attempts to generate informative summaries of articles and documents. Despite its evolution since the 1950s, research on automatically summarising Norwegian text has remained relatively underdeveloped. Though there have been some strides made in extractive systems, which generate summaries by selecting and condensing key phrases directly from the source material, the field of abstractive summarization remains unexplored for the Norwegian language. Abstractive summarization is distinct as it generates summaries incorporating new words and phrases not present in the original text.

This Master's thesis revolves around one key question: Is it possible to create a machine learning system capable of performing abstractive summarization in Norwegian? To answer this question, we generate and release the first two Norwegian datasets for creating and evaluating Norwegian summarization models. One of these datasets is a web scrape of Store Norske Leksikon (SNL), and the other is a machine-translated version of CNN/Daily Mail. Using these datasets, we fine-tune two Norwegian T5 language models with 580M and 1.2B parameters to create summaries. To assess the quality of the models, we employed both automatic ROUGE scores and human evaluations on the generated summaries. In an effort to better understand the model's behaviour, we measure how a model generates summaries with various metrics, including our own novel contribution which we name "Match Ratio" which measures sentence similarities between summaries and articles based on Levenshtein distances.

The top-performing models achieved ROUGE-1 scores of 35.07 and 34.02 on SNL and CNN/DM, respectively. In terms of human evaluation, the best model yielded an average score of 3.96/5.00 for SNL and 4.64/5.00 for CNN/Daily Mail across various criteria. Based on these results, we conclude that it is possible to perform abstractive summarization of Norwegian with high-quality summaries. With this research, we have laid a foundation that hopefully will facilitate future research, empowering others to build upon our findings and contribute further to the development of Norwegian summarization models.

PREFACE

This work rounds off our five-year degrees at the Norwegian University of Life Sciences (NMBU). It is the culmination of our journey through the five-year Master's programme in Data Science which we started together in 2018.

We wish to give special acknowledgement to our supervisor, Kristian Hovde Lilland, for his guidance and insightful feedback throughout the course of this thesis. His deep understanding of machine learning and academic writing was invaluable to this work.
We also extend our appreciation to our co-supervisor, Nader Aeinehchi from The Norwegian Tax Administration for giving us the chance to work on this research. He has given us countless insightful suggestions, constructive critiques, and valuable expertise that greatly enhanced the quality of this work.
Our sincere thanks are also due to Per Egil Kummervold at the Norwegian National Library for his significant contributions to the Norwegian NLP field through his work on Norwegian language models and corpora. Similarly, we acknowledge the guidance and support provided by Gabriel Borg at the Swedish National Archives. His experiences in translating datasets to Swedish and within ATS have provided invaluable assets for our work.
Lastly, we would like to express our gratitude to each other. The collaboration's shared enthusiasm and mutual support have not only made this research possible but also made it an immensely enriching and enjoyable journey.

This thesis is a labour of love, and we are both grateful to have had the opportunity to invest the past few months exploring and contributing to one of the most exciting fields of research. As we wrap up this thesis, our conviction has only grown stronger. More so than when we began, we firmly believe that the topics we've explored will continue to shape our everyday lives for many years to come.

[illegible] Jørgen Navjord

Jon-Mikkel R. Korsvik Jørgen Navjord

Ås, May 14, 2023

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

The forthcoming thesis delves into the application of state-of-the-art transformer models for executing abstractive summarization of the Norwegian natural language.
The thesis is undertaken in collaboration with the Norwegian Tax Administration (Skatteetaten), a forward-thinking organization that recognizes the value of integrating modern technologies, such as summarization, into their operations. Skatteetaten's commitment to innovation is driven by a desire to enhance the experience of its users, both internally and externally.

Internally, employees of Skatteetaten stand to benefit from the implementation of abstractive summarization by gaining quick and concise access to relevant information. By automating the process of summarizing lengthy documents and reports, employees can focus on higher-level tasks, such as decision-making and strategic planning. Additionally, incorporating summarization technologies can streamline internal communication and promote knowledge sharing across departments, ultimately fostering a more efficient and collaborative work environment.
Externally, taxpayers and other stakeholders interacting with Skatteetaten will also reap the benefits of abstractive summarization. With the ability to generate succinct and coherent summaries of complex tax regulations, guidelines, and other documentation, the organization can provide clearer and more accessible information to the public. This improved communication can lead to increased understanding and compliance, reduced errors, and overall enhanced satisfaction among taxpayers.

Although summarization has been extensively explored in English and other languages, there remains a noticeable gap in the development of systems tailored for summarizing Norwegian text. While certain strides have been made in the realm of extractive summarization - a subfield that constructs summaries by extracting words and sentences from source documents - little attention has been devoted to abstractive summarization. This type of summarization, which involves the generation of novel sentences has yet to be fully explored within the context of the Norwegian language.

One of the primary factors contributing to this underexplored area of research is the scarcity of publicly available datasets suitable for training machine learning summarization models. High quality datasets play a crucial role in the development of efficient and accurate models, especially in the field of natural language processing. The availability of such resources in English and other widely spoken languages has enabled significant advancements in those languages, whereas the Norwegian language has been relatively underserved.

Based on the unresearched area of abstraction summarization of the Norwegian natural language, the following research question was developed:

Is it possible to create a machine learning system capable of performing abstractive summarization on the Norwegian natural language?
To find the answer to this question, we formalize three objectives which will help us answer the question.

Objectives

The following objectives form the basis of this thesis:

1. Explore and identify datasets for automatic summarization for the Norwegian language. If no fitting datasets are found, explore the possibilities of generating datasets ourselves.
2. Design and implement an automatic summarization model capable of generating summaries on Norwegian text using a transformer model.
3. Assess the effectiveness of the developed automatic summarization model through quantitative metrics such as ROUGE scores, as well as human evaluation of the generated summaries.

1.1 Motivation

The rapid growth of digital information has led to an overwhelming abundance of data on the internet. Every day, countless articles, research papers, and reports are published, making it nearly impossible for individuals to process and consume all relevant information. This information overload has necessitated effective and efficient methods to condense large volumes of text while retaining the essence and value of the original content. Automatic text summarization systems address this need by providing concise and coherent summaries, enabling users to quickly grasp the main ideas of a document without having to read it in its entirety.

Automatic text summarization finds numerous applications across various domains. For instance, in journalism, summarization systems can help news agencies generate brief news summaries for readers who prefer skimming headlines and key points. In academia, researchers benefit from summaries of scholarly articles to quickly assess their relevance and decide whether to delve deeper into the full text. Additionally, businesses can utilize summarization systems to create executive summaries of lengthy reports or synthesize information from multiple sources for decision-making purposes.

A key motivation for creating summarization systems is enhancing accessibility. Developing systems that produce shorter versions of large text documents allows individuals with reading difficulties, cognitive impairments, or limited time to more easily access and comprehend information. This expands opportunities for a wider audience to benefit from the wealth of knowledge available in text format.

Another significant motivation behind the development of summarization systems is personalized content delivery. Tailoring summaries to individual user preferences, summarization systems can make content more engaging and user-specific. This leads to increased user satisfaction and better information retention, as users receive information more relevant to their interests and needs.

1.2 Background

A summary is a concise and focused version of a text that conveys the primary ideas, arguments, or themes of the original content. The purpose of a summary is to provide readers with an abridged overview of the essential information, allowing them to understand the main points without reading the entire text. Summaries can be created for various types of content, such as articles, books, research papers, speeches, or multimedia presentations. The process of summarizing involves identifying the most significant elements within the material, removing less relevant or redundant information, and presenting the core ideas in a clear and coherent manner. A well-crafted summary should enable readers to grasp the essence of the original content while maintaining the overall context and meaning.

Summaries play a crucial role in various contexts of our daily lives, helping us manage information efficiently and effectively. In the workplace, summaries condense complex reports or meeting minutes, enabling colleagues to quickly grasp the main points and make informed decisions. In academia, students often rely on summaries to review essential concepts from textbooks, lectures, or research articles, aiding comprehension and retention of the material. In the news and current events, summaries in the form of headlines or brief articles help us stay informed about world affairs without delving into lengthy, detailed coverage. Additionally, summaries are commonly employed in the entertainment industry, where movie synopses, book blurbs, or product descriptions give potential consumers a quick idea of the content, allowing them to make informed choices.

Creating high-quality summaries poses a significant challenge, as it requires a deep understanding of the source material, the ability to identify its most critical elements, and the capacity to convey those elements in a clear and coherent manner. One of the main difficulties lies in accurately interpreting the context, nuances, and intent of the original content, which often necessitates a comprehensive grasp of the subject matter and language subtleties. Additionally, selecting the most pertinent information and distinguishing it from less relevant or redundant content can be a complex task, particularly when dealing with dense or multifaceted materials. Furthermore, crafting a well-structured and concise summary involves skilful paraphrasing and synthesis of ideas while preserving the overall meaning and coherence. This process demands not only linguistic proficiency but also critical thinking and analytical abilities, making it a difficult task for both humans and automated systems.

Research fields which work with natural language such as summaries are usually a part of the natural language processing (NLP) research field. NLP is a subfield of artificial intelligence that focuses on enabling computers to understand, interpret, and generate human language. NLP bridges the gap between human communication and computer understanding, allowing machines to analyze, process, and respond to textual and spoken data in a manner similar to humans. By leveraging techniques from linguistics, computer science, and machine learning, NLP aims to develop algorithms and models that can effectively handle various language-related tasks, such as sentiment analysis, machine translation, speech recognition, and summarization.

As a subfield of NLP, Automatic Text Summarization (ATS) focuses on developing computer algorithms and systems capable of generating concise and coherent summaries from original text sources. ATS aims to automate the process of identifying key information, eliminating redundancies, and distilling the core ideas into a shortened version that retains the overall context and meaning. By leveraging natural language processing, machine learning, and artificial intelligence techniques, ATS seeks to create summaries that accurately represent the primary points of the source material while maintaining readability and relevance for the target audience.

The ultimate goal of ATS is to provide users with an efficient and effective means of accessing and understanding the most significant content from vast amounts of textual data, reducing the time and effort required for manual summarization and enabling the rapid dissemination of critical information.

The history of automatic text summarization began with Hans Petter Luhn's work at IBM in the 1950s [1], which proposed a method for extracting significant sentences from documents based on term frequency. Over the years, researchers have explored various approaches, including rule-based methods, statistical techniques, and machine learning algorithms to improve the quality of generated summaries.

In the early years, most summarization techniques relied on heuristics, such as identifying keywords, phrases, and sentence structures that indicate importance within a text. These rule-based methods achieved moderate success but often struggled with complex texts and varying domains.

The 1990s saw the introduction of statistical approaches, which leveraged mathematical models to identify important sentences or phrases based on their frequency and distribution in the document. These methods generally outperformed rule-based approaches but still faced challenges in handling complex texts and producing coherent summaries.

The advent of machine learning and deep learning techniques has revolutionized the field of automatic text summarization. With the emergence of neural networks and advanced NLP models like transformers, researchers have been able to create more accurate and coherent summaries by training algorithms on large-scale datasets. This has led to the development of state-of-the-art systems highly capable of summarization, such as OpenAI's GPT-models [2]and T5-based models [3].

1.3 Structure of thesis

Chapter 2 presents the theoretical foundation for this thesis. The methodology employed is detailed in Chapter 3. Chapter 4 showcases the results obtained. An in-depth discussion of the results is provided in Chapter 5 and compared with the research question and objectives. Lastly, Chapter 6 offers a summary of the findings, along with concluding thoughts and remarks. The appendix includes additional results which had a minor impact on the thesis.

References to the datasets developed as part of this research project can be accessed in Table B.1, each linked to their respective locations on the Huggingface Hub. We've provided an overview of the trained models used throughout this study in Table B.2, also directly linked to their respective Huggingface Hub pages. For a comprehensive understanding of our work, the complete source code used in deriving the results of this thesis is available at https://github.com/navjordj/Master-Norwegian-Abstractive-Summarization, where the final version of the code for this thesis corresponds to the *b985300* commit.

CHAPTER 2 THEORY

This chapter delves into the underlying theory used to address the proposed objectives, spanning dataset curation, abstractive summarization modelling, and evaluation.

We begin with an introduction to the field of natural language processing (NLP) in Section 2.1, covering essential topics such as tokenization, word embeddings, and language models. Next, Section 2.2 explores the field of automatic text summarization (ATS) by discussing different types of extractive and abstractive techniques and examining summary evaluation methods. Section 2.3 covers the field of machine translation, introducing various translation techniques and explaining how translation evaluation is performed. Subsequently, Section 2.4 provides an introduction to machine learning, followed by an overview of artificial neural networks in Section 2.5, including different variants such as RNNs and Encoder-decoder models. Section 2.6 discusses how these models can effectively generate sequences of text using various token decoding techniques. Finally, building on the topics covered, we introduce the state-of-the-art Transformer architecture and its T5 variant in Section 2.7.

2.1 NLP

Natural Language Processing (NLP) is a branch of artificial intelligence that deals with the interface between computers and human languages [4, pg. ix]. NLP aims to create computational methods and algorithms that facilitate machines in comprehending, interpreting, and generating human language in a manner that is both meaningful and effective. The scope of NLP is broad and encompasses various tasks such as part-of-speech tagging, sentiment analysis, machine translation, and named entity recognition, among others. To perform these tasks, NLP requires the ability to process and analyze extensive volumes of textual data, often referred to as corpora, to learn valuable information and insights from them. Technologies based on NLP, like predictive text, handwriting recognition and machine translation have become increasingly prevalent, and language processing plays a central role in the multilingual information society, offering more natural human-machine interfaces and sophisticated access to stored information [4].

One of the primary challenges in NLP is the inherent ambiguity present in human languages, which can manifest at different levels, such as syntax, semantics, and pragmatics [5]. This ambiguity, coupled with the variability of human language, makes it difficult for machines to derive meaning from text. Early approaches to NLP tried to solve this problem by relying heavily on rule-based and statistical methods, which focused on hand-crafted features and probabilistic models to represent linguistic structures and relationships [6].

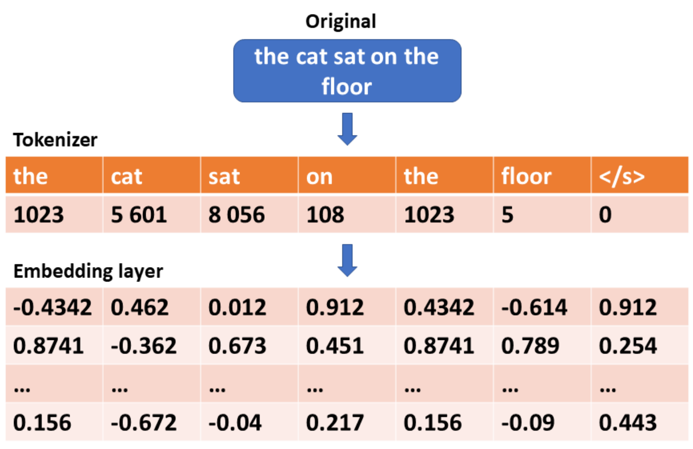
However, the advent of deep learning has revolutionized the field of NLP, paving the way for more advanced and powerful techniques. Deep learning models, such as Recurrent Neural Networks (RNNs) and Transformer-based architectures [7], have demonstrated their ability to learn and represent complex linguistic patterns from large-scale data. These models have achieved state-of-the-art performance on NLP tasks such as abstractive summarization and machine translation [8, 9].

2.1.1 Tokenization

Since computers are inherently not designed to process human language directly, it is necessary to convert characters and words into numerical representations, such as integers, that can be more easily manipulated by computers. In the domain of NLP, tokenization serves as a crucial step in this transformation process, converting raw textual data into a structured format. As a fundamental pre-processing technique, tokenization entails breaking down text into smaller units, or tokens, which can be words, subwords, or characters, depending on the selected approach.

Figure 2.1 illustrates an example of a tokenizer that divides a sentence into individual words and associates them with corresponding integer values as well as converting the integers to a dense vector with an embedding layer explained in Section 2.1.2. The word "the" appears twice in the sentence, and both occurrences are mapped to the same integer. Additionally, the tokenizer appends a $< /s >$ token, which some models discussed later in the thesis require as a signal which indicates the end of a sequence or sentence.

These tokens serve as the building blocks for various NLP tasks, such as machine translation and sentiment analysis. In this section, we will delve into the various tokenization techniques and their applications in the Norwegian language. Moreover, we will discuss the impact of tokenization choices on the performance of NLP models and how these decisions can affect the quality of downstream tasks.



the	cat	sat	on	the	floor	</s>
1023	5 601	8 056	108	1023	5	0

-0.4342	0.462	0.012	0.912	0.4342	-0.614	0.912
0.8741	-0.362	0.673	0.451	0.8741	0.789	0.254
...	...	...	...	...	...	...
0.156	-0.672	-0.04	0.217	0.156	-0.09	0.443

Figure 2.1: Example of word-based tokenization and embedding of the sentence "the cat sat on the floor"

Tokenization techniques can be broadly categorized into three main approaches: word-based, subword-based, and character-based tokenization. Each of these methods has its own set of advantages and drawbacks, which makes them more suitable for specific tasks or languages. In this section, we will explore these three approaches in detail, along with their respective tokenization algorithms and their applications in NLP tasks.

Word-based Tokenization

Word-based tokenization is the most intuitive method, wherein textual data is segmented into individual words. This approach is widely used in languages with clear word boundaries, such as English, where spaces separate words. However, word-based tokenization may not be as effective for languages that lack explicit word boundaries, such as Chinese or Japanese. The primary advantage of this method is its simplicity and interpretability, but it may suffer from issues like data sparsity and out-of-vocabulary (OOV) tokens. OOV tokens can occur when a token is found in the test set which does not occur in the train set.

Subword-based Tokenization

Subword-based tokenization addresses some of the limitations of word-based tokenization, such as handling OOV tokens and reducing data sparsity. This approach segments text into subword units, which can be smaller than words but larger than characters. Subword tokenization is especially useful for morphologically rich languages and can improve the generalization of NLP models. Some of the most used subword tokenization algorithms include:

- **Byte Pair Encoding (BPE)**: BPE is a data compression algorithm that was adapted for tokenization by Sennrich et al. [10]. It merges the most frequent character pairs in the training data iteratively until a predefined vocabulary size is reached. The resulting vocabulary consists of both words and subword units.
- **Unigram**: This tokenization method is proposed by Taku Kudo [11] and is based on a unigram language model. It constructs a vocabulary of subword units by maximizing the likelihood of the training data given the model. The vocabulary is selected in such a way that it minimizes the perplexity (probability of a generated sequence)of the training data, leading to better generalization in NLP tasks.
- **SentencePiece**: SentencePiece [12] is a tokenizer which uses both BPE and Unigram. It includes spaces in the tokens it generates which makes it better suited for languages such as Chinese or Japanese which does not use spaces to separate words like for the English language. Due to this better handling of non-English text, it is widely used for multilingual models, such as mT5 [13].

These subword tokenization algorithms have been widely adopted in modern NLP models, as they offer several advantages over traditional word-based tokenization methods. By representing text as a sequence of subword units, these models can better handle OOV words, rare words, and morphologically rich languages, leading to improved performance and generalization across various tasks and languages.

Character-based Tokenization

Character-based tokenization segments text into individual characters, making it the most granular approach among the three. It has the advantage of eliminating OOV tokens and can capture fine-grained morphological information. One notable issue with character-based tokenization is that it can lead to longer input sequences compared to word-based or subword-based approaches, which may result in increased computational complexity. Moreover, since the meaning of a text is often conveyed at a higher level than individual characters, character-based models may need to learn more complex representations to capture meaningful linguistic structures.

2.1.1.1 Tokenizing Norwegian

While the general principles of tokenization are the same for Norwegian and other languages such as English, some specific considerations must be taken into account due to differences in the linguistic properties of these languages. In this section, we discuss the challenges and differences in tokenization approaches for Norwegian and their comparison to English.

Morphological Differences Norwegian, as a North Germanic language, has a more complex morphology compared to English. It exhibits rich inflexion, particularly in its noun and verb systems. For example, Norwegian nouns can have definite and indefinite forms, as well as plural forms, leading to a larger number of possible word forms than in English. Furthermore, Norwegian has two written standards, Bokmål and Nynorsk, which adds another layer of complexity when tokenizing the text. These morphological differences can have a significant impact on the choice of tokenization strategy. Subword-based tokenization, such as Byte-Pair Encoding (BPE) or SentencePiece, may be more suitable for Norwegian, as it can better handle variations in word forms.

Handling Compound Words Norwegian, like other Germanic languages, is known for its compound words, where multiple words are combined into a single word without spaces. This characteristic can present challenges for tokenization, as splitting compound words into their constituent parts may be necessary to accurately capture their meaning. Additionally, it is common for Norwegian translations to be 5% to 10% shorter than their English counterparts [14], which can also contribute to the difference in the number of characters in sentences and documents.

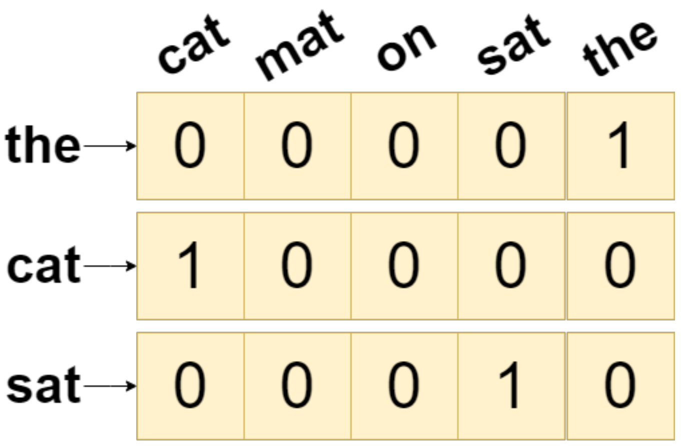
For example, in Norwegian, the compound word "arbeidsmiljø" can be separated into "arbeids" (work) and "miljø" (environment). In English, compound words are generally less frequent and are often divided by spaces, such as "work environment." Therefore, tokenization approaches for Norwegian may need to incorporate techniques for handling compound words, such as rule-based or machine learning-based methods.

2.1.2 Word Embeddings

A crucial aspect of NLP tasks is the representation of words as inputs for the machine learning models. Traditional one-hot encoding [15], which represents words as binary vectors with a single '1' at the index corresponding to the word and '0's elsewhere. An example of the sentence "the cat sat" being one-hot encoded into vectors is shown in Figure 2.2. This results in high-dimensional, sparse, and computationally inefficient representations. One hot-encoding of a single word, for instance, with a vocabulary of 10,000 words would produce a vector in which 99.99% of the elements are zero.

To overcome these limitations, word embeddings were introduced, which are dense vector representations of words that capture their semantic meaning and syntactic relationships with other words in a continuous space [16]. By reducing the dimensionality and capturing semantic relationships, word embeddings enable more efficient and effective training of neural networks for NLP tasks.

Figure 2.3 presents a commonly used illustration of word embeddings. In this hypothetical example, the words "Man," "Woman," "King," and "Queen" have been transformed into 2D



	cat	mat	on	sat	the
the→	0	0	0	0	1
cat→	1	0	0	0	0
sat→	0	0	0	1	0

Figure 2.2: Example of one-hot encoding for the sentence 'the cat sat,' using a vocabulary size of 5. Each word's vector has a '1' in the index corresponding to the word and '0's in all other positions.

word embeddings (vectors). This equation highlights the capacity of the word embedding model to represent the relationships between words in a continuous vector space, enabling meaningful comparisons and operations on the word vectors. The word embedding model has effectively captured the semantic relationships between these words, as demonstrated by performing vector arithmetic on the embeddings:

$$\overrightarrow{\text{King}} - \overrightarrow{\text{Man}} + \overrightarrow{\text{Woman}} \approx \overrightarrow{\text{Queen}} \tag{2.1}$$

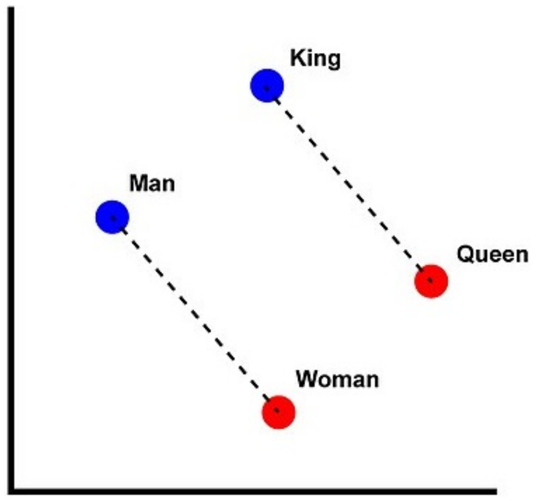


Figure 2.3: 2D visualization of word embeddings for the words "Man," "Woman," "King," and "Queen." The relationships between these words are captured in the vector space, demonstrating that the word embedding model has successfully learned semantic relationships. The plot illustrates the intuitive vector arithmetic involving these embeddings. Modification of [17].

Today's word embedding methodologies predominantly rely on neural networks to learn real-valued vector representations of individual words. Two straightforward techniques include the bag-of-words (BOW) model and the word skip-gram model, commonly known as word2vec models [16]. The BOW model trains a shallow neural network to predict a single word based on a context window comprising neighbouring words, which are usually one-hot encoded. After training a neural language model in this manner, the hidden layer's weights represent a vector space where each vector corresponds to a specific word in the vocabulary. These vectors can then serve as word embeddings. The word skip-gram model inverts the BOW model's task and predicts context words based on a single input word. Other neural networks, such as RNNs and transformers, have also been effectively utilized to learn word embeddings [16] [7].

2.1.3 Language Models

Language Models (LMs) have emerged as a promising approach in the field of NLP, demonstrating remarkable capabilities in understanding, generating, and reasoning over human language. Large Language Models (LLMs) represent a significant evolution within this field. Characterized by their vast number of parameters, often in the range of billions, LLMs leverage extensive amounts of data to learn complex patterns and representations of language. This enables them to capture intricate linguistic relationships and generalize well to a variety of tasks.

Language modelling aims to estimate the probability distribution of a sequence of words or tokens in a given text [18]. Formally, given a sequence of words $w_1, w_2, ..., w_n$, the goal is to model the probability $P(w_1, w_2, ..., w_n)$ by factorizing it into a product of conditional probabilities:

$$P(w_1, w_2, ..., w_n) = \prod_{i=1}^{n} P(w_i \mid w_1, w_2, ..., w_{i-1}) \tag{2.2}$$

By estimating these probabilities, language models can learn to generate coherent and grammatically correct text, while also capturing the context and dependencies between words. For example, consider the sentence "I should pay my taxes" A language model estimates the probability of each word occurring given the previous words in the sentence. When the model reaches the word "taxes," it calculates the probability of "taxes" occurring given the context "I should pay my" As the model processes the entire sentence, it learns the contextual relationships between the words, which allows it to generate contextually relevant and coherent text.

One of the key innovations behind LMs is the use of pre-training and fine-tuning strategies. During the pre-training phase, models are exposed to massive corpora, allowing them to learn general language representations and structures. The fine-tuning phase adapts the pre-trained model to specific tasks, such as abstractive summarization or sentiment analysis, by training on a smaller, task-specific dataset. This two-step process has proven to be highly effective, as LLMs have achieved state-of-the-art performance across a wide array of NLP benchmarks.

The transition from traditional LMs to LLMs (Large Language Models) was marked by the introduction of models such as GPT-2 by OpenAI [18]. These LLMs employed more advanced architectures, such as the Transformer (covered in depth in Section 2.7), and were trained on significantly larger datasets. A key feature of these LLMs is their ability to be used in a zero-shot learning context. Zero-shot learning is a machine learning concept where a model makes predictions about data it was not specifically trained on. For instance, a model trained on text classification tasks can be used to classify text in a category it has never seen before. This is achieved by leveraging the model's understanding of language and its ability to generalize from its training to novel tasks.

2.2 Automatic Text Summarization

Automatic Text Summarization (ATS) is a vital and increasingly relevant task in the field of NLP. As the volume of textual data available online continues to grow exponentially, there is a rising need to develop efficient methods for condensing long and complex texts into shorter, more digestible summaries. This process should maintain the core information and meaning of the original text while generating summaries that are coherent, fluent, and appear as if written by humans.

The primary goal of ATS is to facilitate easier and quicker comprehension of large volumes of information, enabling readers to understand the essence of a text without having to read it in its entirety. This is especially useful in various domains, such as journalism, legal documents, research articles, and social media, where professionals and casual readers alike can benefit from the time-efficient consumption of information.

Since its inception in the late 1950s, the field of automatic text summarization (ATS) has undergone significant evolution. Initially, the focus was on using statistical techniques and rule-based algorithms to extract sentences or phrases deemed most important or relevant to the document's overall content. However, summaries generated using these extractive techniques often lacked coherence and structure.

As the field progressed, more sophisticated approaches, such as graph-based methods, clustering algorithms, and neural network models, emerged, leading to more coherent and well-structured summaries. This early technique of extracting important phrases or sentences from the original text to generate an informative summary is now commonly referred to as extractive summarization. While extractive summarization techniques have improved over time, they still face challenges in generating summaries that are coherent and readable.

To address these challenges, there has been a shift towards abstractive summarization techniques, which aim to generate summaries that are more fluent and human-like. This shift has led to the development of more advanced NLP techniques and models, including transformer-based models like GPT [2], which are capable of producing summaries that capture the essential information in the input text while also being more natural and readable. These abstractive summarization techniques have demonstrated significant progress towards generating informative and coherent summaries that closely mimic human-written summaries.

2.2.1 Extractive Summarization

Extractive summarization is one of the main approaches to automatic text summarization. It involves identifying and extracting relevant sentences or phrases from the original text and combining them to form a coherent summary. In this section, we will explore the various methods and techniques used in extractive summarization, starting from early approaches to more advanced techniques such as BERT-based methods. We will also discuss the limitations and challenges faced by extractive summarization.

Early extractive summarization methods relied on simple heuristics and statistical techniques to identify important sentences. Some common strategies included selecting sentences based on their position in the text, keyword frequency, or the presence of cue phrases that indicate importance.

LEAD

One such technique is the LEAD summarization method, which selects a certain number of the initial sentences of the document as the summary [19]. For example, LEAD-1 would select the first sentence, LEAD-2 would select the first two sentences, and so on. This approach is based on the assumption that the beginning of a text often contains crucial information and sets the context for the rest of the document. While simple and efficient, these approaches, including various LEAD methods, were prone to generating summaries that lacked coherence and fluency. Due to this technique's simplicity, it is often used as a baseline to compare against more advanced techniques.

TextRank

TextRank is an unsupervised graph-based extractive summarization method inspired by Google's PageRank algorithm for ranking web pages [20]. It identifies and extracts the highest-ranked sentences from a document to generate an informative summary. The algorithm constructs a graph with sentences as nodes and edges between nodes based on the similarity between sentences, typically calculated using cosine similarity between vector representations of the sentences. Cosine similarity is a measure of similarity between two non-zero vectors, which evaluates the cosine of the angle between them. It ranges from -1 (completely dissimilar) to 1 (completely similar), with 0 indicating orthogonality or no similarity.

Upon constructing the graph, TextRank iteratively assigns scores to each sentence node, with a sentence's score determined by the scores of its neighbouring sentences, weighted by their similarity. After convergence, the top-ranked sentences are selected and combined in their original order from the source text, resulting in a summary. TextRank has demonstrated improved coherence and structure in summaries compared to earlier extractive summarization techniques.

Embedding-based Models

Recent advancements in NLP, particularly the emergence of powerful pre-trained language models and effective sentence embedding techniques, have led to the development of more advanced extractive summarization methods that leverage these rich representations. Unlike word embeddings, which represent individual words, sentence embeddings capture the meaning of entire sentences, enabling better identification and ranking of important sentences in the input document. Models such as BERT [9], Universal Sentence Encoder [21], and InferSent [22] can generate meaningful and contextualized embeddings for sentences.

One such model that leverages the power of BERT is BERTSum [23], which fine-tunes the pre-trained BERT model for the summarization task. BERTSum and similar models operate by processing each sentence in the document, along with its surrounding context, through the chosen language model or embedding method. The generated embeddings attempt to capture the semantic information and relationships between sentences, which are then used to compute importance scores for the sentences. The top-ranked sentences, determined by their importance scores, are selected and combined in their original order in the source document to form the final summary.

2.2.2 Abstractive Summarization

Abstractive summarization is the other major approach to automatic text summarization. Unlike extractive summarization, which involves selecting and combining existing sentences or phrases from the source text, abstractive summarization generates summaries by rewriting or paraphrasing the original content in a more precise manner [24, pg.3]. This approach aims to create more natural, coherent, and concise summaries that closely resemble human-written summaries, capturing the essence of the input text without being constrained by its exact wording.

Early abstractive summarization methods relied on template-based approaches and rule-based systems, which generated summaries by fitting the extracted information from the source text into predefined structures or templates [25]. These methods often required manual effort to define the templates and rules, and the resulting summaries were limited by the rigidity and simplicity of the templates, leading to summaries that might not be as natural or coherent as desired. As the field of natural language processing advanced, more sophisticated abstractive summarization techniques emerged, aiming to overcome the limitations of early approaches and produce higher-quality summaries.

Sequence-to-Sequence and Transformer Models

With the advent of deep learning, sequence-to-sequence (seq2seq) models [26] became a popular approach for abstractive summarization tasks. These models employ an encoder-decoder architecture, wherein the encoder processes the input text and captures its semantic information in a continuous vector representation. The decoder then generates a summary by conditioning this representation. Seq2seq models have shown significant improvement in the quality of abstractive summaries, as they are capable of generating more coherent and fluent summaries compared to earlier methods. However, they still face challenges in handling long input texts and may sometimes produce summaries with incorrect or repetitive information. A detailed discussion of encoder-decoder models for seq2seq tasks is covered in Section 2.5.5.

Building on the success of seq2seq models, transformer-based architectures [7] has emerged as a powerful approach for abstractive summarization, which is covered in more detail in Section 2.7. Transformers leverage the attention mechanism to capture complex dependencies between words in the input text, allowing for more effective encoding of the document's content. State-of-the-art transformer-based models, such as GPT [2] and T5 [3], have demonstrated exceptional performance in generating high-quality, coherent, and concise abstractive summaries.

2.2.3 Measuring Abstractiveness

While a model is designed to perform abstractive summarization, it may still opt to generate summaries by extracting sentences and phrases directly from the source document. This could be attributed to various factors, such as the inherent structure and content of the source document, which may be such that extracting key sentences or phrases can produce a coherent and meaningful summary without the need for extensive paraphrasing.

Additionally, the model's training and evaluation procedures may inadvertently favour extractive summarization strategies. For instance, if the model was trained on a dataset that contains a significant number of extractive summaries, it may learn a bias towards such strategies. Moreover, limitations in the model's architecture or optimization process may also hinder its ability to generate genuinely abstractive summaries, causing it to rely more heavily on extractive techniques. Furthermore, the evaluation metrics used during model development such as ROUGE (Section 2.2.4) often measure n-gram overlap between the generated and reference summaries, potentially encouraging the model to generate summaries that resemble the source text more closely.

Some techniques have been developed to measure the "abstractiveness" of abstractive summarization models. Measuring this is an important stage in understanding how these models work and why they generate the resulting summaries. This is especially relevant in an age of black box models such as Transformers with low explainability. Some techniques have been developed to measure this, primarily based on overlapping words between the summary and document as well as text distances.

In the paper presenting the Newsroom dataset, a comprehensive dataset for summarization comprising more than 1.3 million article-summary pairs [27], the authors introduce two measures to assess summarization strategies: coverage and density. These metrics are designed to measure how much overlap there is between the summary and the article.

Given A consisting of a sequence of tokens in the article and the corresponding S consisting of a sequence of tokens in the summary, the set of extractive fragments $\mathcal{F}(A, S)$ is the set of shared sequences of tokens in A and S. These sequences are identified using a greedy process, which processes the tokens in summary in order and marks the longest prefix possible as extractive at each step. $\mathcal{F}(A, S)$ is a set which includes all the token sequences identified as extractive.

Using $\mathcal{F}(A, S)$, the authors compute two measures: coverage and density.

Coverage Coverage quantifies what percentage of the words in a summary is extracted from the article being summarized using $\textsc{Coverage}(A, S)$. Here, $|S|$ and represents the size of the summary and $|f|$ is the length of a extractive fragment:

$$\textsc{Coverage}(A, S) = \frac{1}{|S|} \sum_{f \in \mathcal{F}(A,S)} |f| \tag{2.3}$$

For instance, if a summary consists of 10 words, out of which 7 words are taken from the corresponding article, and 3 words are newly introduced, then the Coverage metric for that summary would be calculated as $\textsc{Coverage}(A, S) = 0.7$.

Density Density measures how long the extractive fragments used by a summary are. When a summary utilizes many words directly from the article, it tends to have increased coverage, Nevertheless, by reorganising the words, the summary could express concepts that the article does not explicitly state. The $\textsc{Density}(A, S)$ denotes the mean length of the article text from which each word in the summary is extracted:

$$\textsc{Density}(A, S) = \frac{1}{|S|} \sum_{f \in \mathcal{F}(A,S)} |f|^2 \tag{2.4}$$

For example, a summary of 10 words derived from two extractive sequences of 3 and 4 words from the article would yield $\textsc{Coverage}(A, S) = 0.7$ and $\textsc{Density}(A, S) = 2.5$. Compared to coverage, density provides a greater overview of longer sequences which have been extracted from the article and the Newsroom authors suggest relying on density as the primary measurement of abstractiveness.

Match Ratio To measure the degree of a summary consisting of whole sentences extracted from an article, we propose a novel method based on Levenshtein Distances which we call Match Ratio to measure this. Coverage and Density measure the overlap of subsequences of text but are not able to understand that sequences with small differences are similar (e.g. Different pluralisations of a word). Levenshtein distance measures the number of operations needed to transform a sequence of text into another. Using this distance, we can use the pair-wise distances between sentences in the article and summary to measure if a sentence in the summary is extracted from the article.

Given a summary sentence a and a source sentence b, the Levenshtein distance lev can be calculated using the recursive formula:

$$\operatorname{lev}(a,b) = \begin{cases} |a| & \text{if } |b| = 0, \\ |b| & \text{if } |a| = 0, \\ \operatorname{lev}\big(\operatorname{tail}(a), \operatorname{tail}(b)\big) & \text{if } a[0] = b[0], \\ 1 + \min \begin{cases} \operatorname{lev}\big(\operatorname{tail}(a), b\big) \\ \operatorname{lev}\big(a, \operatorname{tail}(b)\big) \\ \operatorname{lev}\big(\operatorname{tail}(a), \operatorname{tail}(b)\big) \end{cases} & \text{otherwise} \end{cases} \tag{2.5}$$

where tail(x) denotes the string containing all characters of x except the first one, and $x[n]$ represents the n-th character of string x, counting from 0. In essence, the Levenshtein distance represents the lowest amount of edits needed to transform a into b, analogous to the shortest distance between them.

For example, consider the strings "kitten" and "sitting". The Levenshtein distance between these strings is 3, as demonstrated by the following series of edits:

1. **k**itten → **s**itten (substitution of "s" for "k")
2. sitt**e**n → sitt**i**n (substitution of "i" for "e")
3. sittin → sittin**g** (insertion of "g" at the end)

To evaluate the degree of similarity between summary and source sentences, we can normalize the Levenshtein distance by dividing it by the maximum possible distance (i.e., the length of the longest of the two sentences), resulting in a similarity score ranging from 0 (entirely dissimilar) to 1 (identical). To define that a sentence is extracted from the article, we can use a distance threshold (e.g. $\operatorname{lev}(a,b) > 0.95$). Using this, we define the Match Ratio (MR) as

$$MR = \frac{M}{N} \tag{2.6}$$

Where, M represents the number of nearly identical sentences (based on the defined threshold) in the summary and the source article, and N represents the total number of sentences in the summary.

2.2.4 Evaluating Generated Summaries

The evaluation of summarization models is an essential aspect of developing effective and reliable systems for automatic text summarization. Assessing the quality and performance of these models helps to identify their strengths and weaknesses, enabling researchers to refine and improve their algorithms. Furthermore, evaluation provides valuable insights into the practical impact of summarization models on real-world tasks, user experiences, and applications.

There are several methods used to evaluate summarization models, which can be broadly categorized into the extrinsic evaluation and intrinsic evaluation.

Extrinsic Evaluation

Extrinsic evaluation refers to assessing the performance of a summarization model by examining its impact on an external task [28]. The idea is to measure how well the generated summaries contribute to the overall goal of the application, such as improving user comprehension, saving time in information retrieval, or enhancing decision-making processes.

In the context of abstractive summarization, extrinsic evaluation can be performed by incorporating the generated summaries into a real-world application and measuring the effectiveness of the application with and without the summaries. For example, one could evaluate the summaries by conducting a user study in which participants are asked to perform a task using the summaries, and then compare their performance to a control group that uses the original text or a different summary method.

It is important to note that extrinsic evaluation is often more challenging and time-consuming than intrinsic evaluation, as it requires the design and execution of experiments that involve human participants and real-world tasks. However, it provides valuable insights into the practical impact of the summarization model on specific applications and user experiences.

Intrinsic Evaluation

Intrinsic evaluation, on the other hand, focuses on assessing the quality of the generated summaries independently of their impact on external tasks. This can be done by measuring various aspects of the summaries, such as relevance, coherence, conciseness, or fluency, using automated metrics or human evaluation.

Automated metrics for intrinsic evaluation typically compare the generated summaries to reference summaries, either created by humans or obtained from a dataset. Examples of automated metrics for intrinsic evaluation include ROUGE [29] and BLEU [30]. These metrics compare the overlap of n-grams between the generated and reference summaries, providing a quantitative score that reflects the degree of similarity. ROUGE and BLEU differ primarily as ROUGE is based on recall while BLEU measures precision. In the context of overlapping n-grams, precision refers to the proportion of n-grams in the generated summary that also appear in the reference summary, while recall measures the proportion of n-grams in the reference summary that are present in the generated summary.

The other approach to intrinsic evaluation on summaries is based on qualitative human evaluation, where humans rate the quality of generated summaries on a set of different pre-defined criteria such as informativeness or fluency.

ROUGE ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is an evaluation metric widely used for assessing the quality of generated summaries in summarization tasks. ROUGE is based on the idea of measuring the overlap between the generated summary and one or more reference summaries. It is a recall-oriented metric, as it primarily focuses on capturing the information content present in the reference summaries. The ROUGE metric was first proposed by Chin-Yew Lin in the paper titled "ROUGE: A Package for Automatic Evaluation of Summaries" in 2004 [29]. In this paper, a comprehensive evaluation was conducted, involving human evaluators who assessed the quality of generated summaries. The results demonstrated a strong correlation between the ROUGE scores and human evaluations across various ROUGE metrics, thereby substantiating the metric's effectiveness in reflecting human judgement.

The core concept behind ROUGE is the overlap of n-grams between the generated summary and the reference summaries. An n-gram refers to a series of n-words that appear in consecutive order. For example, unigrams are single words, bigrams are pairs of consecutive words, and trigrams are sequences of three consecutive words. By calculating the overlap of n-grams between the generated and reference summaries, ROUGE quantifies the extent to which the generated summary captures the salient information in the reference summaries.

Different variants of ROUGE metrics have been developed to address different aspects of summarization quality and offer a more comprehensive evaluation of the generated summaries. Each variant focuses on specific characteristics of the summaries, capturing different types of similarities and providing unique insights into the performance of summarization models. The necessity for different ROUGE measures arises due to the inherent complexity and variability of natural language, as well as the diverse ways in which information can be expressed and organized in text. Some of these variants include:

- **ROUGE-1**: Measures the overlap of unigrams (single words) between the generated summary and reference summaries. Higher ROUGE-1 scores indicate more common words between the summaries, leading to better performance. The score is computed as a recall measure, dividing the number of overlapping unigrams by the total number of unigrams in the reference summaries.
- **ROUGE-2**: Measures the overlap of bigrams (pairs of consecutive words) between the generated summary and reference summaries. Higher ROUGE-2 scores indicate more common bigrams between the summaries, reflecting better performance. The score is computed as a recall measure, dividing the number of overlapping bigrams by the total number of bigrams in the reference summaries.
- **ROUGE-L**: Based on the Longest Common Subsequence (LCS) between the generated summary and reference summaries, ROUGE-L computes recall as the length of the LCS divided by the total number of words in the reference summary. Unlike ROUGE-1 and ROUGE-2, ROUGE-L does not rely on fixed-length n-grams, making it more flexible and better suited for capturing longer paraphrases.
- **ROUGE-LSum**: A variant of ROUGE-L that computes the LCS at the sentence level instead of on the entire summary. It averages the scores of the LCS for each sentence in the generated summary and the reference summary. This approach may lead to differences in the results when compared to the original ROUGE-L metric, as it puts more emphasis on the local coherence between sentences.

While ROUGE is widely used for evaluating summarization models due to its simplicity and computational efficiency, it has some notable limitations. It is insensitive to semantics, focusing on the overlap of n-grams or subsequences without considering semantic similarity [31]. As a result, ROUGE may not recognize the semantic similarity between summaries with similar meanings but different words or phrasings. Moreover, the quality of reference summaries directly affects ROUGE scores, potentially misrepresenting the quality of generated summaries if reference summaries inadequately capture relevant information [32].

Given these limitations, a more comprehensive assessment of summarization models requires combining ROUGE with other evaluation methods, such as human evaluation.

Human evaluation Human evaluation involves assessing the quality of generated summaries by obtaining ratings from human judges. This approach is considered to be more reliable and accurate than automated metrics, as humans can better understand and appreciate the nuances of natural language, including semantics, coherence, and context. Human evaluators can provide qualitative feedback on various aspects of the summaries, such as informativeness and fluency, offering a comprehensive evaluation of the summarization model's performance.

However, this process also comes with certain challenges and limitations. Subjectivity, time, and resources required to obtain evaluations from human judges can be significant concerns. Moreover, the process can be slow, as it involves manual efforts from evaluators to carefully read and assess the generated summaries. To address these issues, a set of guidelines or criteria for each metric is typically provided to the evaluators [27], ensuring consistency and objectivity in the assessment process. Additionally, employing a diverse group of evaluators or using a blind evaluation process can help mitigate potential biases and subjectivity, while streamlining the evaluation process may help in reducing the time required for assessments.

While there are no single standards for human evaluators, some measurements have evolved to be commonly used. Grusky et al. in the Newsroom paper discussed in Section 2.2.3 evaluate summaries on four different categories: informativeness, relevance, fluency, and coherence [27], as described in Table 3.1. The informativeness dimension evaluates how well the summary captures the key points of the article, while the relevance dimension assesses whether the details provided by the summary are consistent with the details in the article. Fluency measures whether the individual sentences of the summary are well-written and grammatical, and coherence evaluates if the phrases and sentences of the summary fit together and make sense collectively. Evaluation setups may vary, involving single or multiple evaluators, which can impact the evaluation process and results.

2.3 Machine Translation

Machine Translation (MT) is a subfield of NLP that focuses on automatically translating text from one language to another. With an increasingly globalized world, the demand for efficient and accurate translation systems has grown exponentially. Machine translation systems aim to facilitate cross-lingual communication, improve access to multilingual information, and support various applications, such as e-commerce, customer support, and content localization.

Early machine translation systems relied on rule-based and statistical approaches, which faced challenges in handling the complexity and diversity of natural languages. However, with the advent of deep learning and advancements in NLP, more sophisticated and effective machine translation systems have been developed, including neural machine translation (NMT) models that have demonstrated state-of-the-art performance. In this section, we will discuss the evolution of machine translation, the major approaches, and their respective strengths and limitations.

2.3.1 Rule-Based Machine Translation

Rule-Based Machine Translation (RBMT) is one of the earliest approaches to MT, which relies on manually created rules to map the source language's grammar, syntax, and semantics to the target language [33]. These systems typically involve three main components: a source language analyzer, a transfer component, and a target language generator. The source language analyzer processes the input text and generates a syntactic and semantic representation. The transfer component applies a set of rules to convert the source language representation to the target language representation, and the target language generator then produces the translated text.

RBMT systems have some advantages, such as the ability to handle idiomatic expressions and domain-specific terminology by incorporating expert knowledge in the form of rules. However, they suffer from several limitations, including the time-consuming and labour-intensive process of creating and maintaining rules, the difficulty of handling linguistic ambiguities and variations, and the rigidity of the system, which often results in less fluent and natural translations.

2.3.2 Statistical Machine Translation

Statistical Machine Translation (SMT) is a data-driven approach that emerged in the late 1990s as an alternative to rule-based methods [34]. SMT models learn translation probabilities from large parallel corpora, which consist of aligned text pairs in the source and target languages. These models typically involve three main components: a translation model, a language model, and a decoding algorithm. The translation model captures the relationships between words or phrases in the source and target languages, while the language model estimates the probability of a sequence of words in the target language. The decoding algorithm then searches for the most probable target language sequence given the input text.

SMT models offer several advantages over RBMT, such as the ability to learn from data and generalize to unseen texts, more natural translations, and the potential for unsupervised and semi-supervised learning. However, SMT models have their limitations, including the reliance on large parallel corpora, the difficulty in handling long-range dependencies and complex linguistic structures, and the tendency to produce translations with errors and inconsistencies.

2.3.3 Neural Machine Translation

Neural Machine Translation (NMT) is a deep learning-based approach that has revolutionized the field of machine translation in recent years [35]. NMT models employ end-to-end learning to map the input text in the source language to the output text in the target language. These models typically use encoder-decoder architectures based on transformers, similar to those used in state-of-the-art abstractive summarization, as discussed in Section 2.7. The encoder processes the input text and generates a continuous vector representation, while the decoder generates the translated text by conditioning on this representation. NMT models leverage the power of neural networks to capture complex dependencies and relationships between words in the source and target

2.3.4 Evaluating Machine Translation with BLEU

Evaluating the quality of machine translation systems is crucial for measuring progress, comparing different approaches, and guiding future research. Human evaluation is considered the best way of measuring the quality of translations, as it takes into account factors such as fluency, adequacy, and meaning preservation. However, human evaluation is time-consuming, costly, and subject to inconsistencies due to subjective judgements. As a result, various automatic evaluation metrics have been developed to approximate human judgements, with the most widely used and influential metric being the Bilingual Evaluation Understudy (BLEU).

BLEU [30] is an automatic evaluation metric designed to measure the quality of machine-generated translations by comparing them to one or more human-generated reference translations. The main idea behind BLEU is to compute the similarity between the machine translation and the reference translations, based on the overlap of n-grams. N-grams are contiguous sequences of n words, and BLEU uses n-grams of different lengths (usually up to 4) to capture lexical and syntactic similarities at various granularities.

Evidence indicates a strong association between BLEU scores and human assessments of translation quality [30], making it a reliable proxy for human evaluation in many cases. It is computationally efficient and can be applied to large-scale evaluation tasks. Additionally, as a purely statistical metric, BLEU can be applied to any language pair without requiring language-specific knowledge or resources.

However, BLEU also has several limitations. It is based on n-gram overlap, which may not fully capture semantic and structural aspects of translation quality. Also, it is sensitive to the choice of reference translations and may not be as effective when dealing with languages that have flexible word order or complex grammar structures [36].

BLEU scores range from 0 to 1 but are often scaled to a range of 0 to 100, with 100 indicating a perfect match with the reference translations and 0 indicating no overlap. In practice, a perfect score of 100 is nearly impossible to achieve, as human translations can also differ from one another. A BLEU score above 30 is generally considered good, while a score above 40 is considered to be of high quality. It is important to note that the interpretation of BLEU scores can be dependent on the specific language pair and the domain of the translation task, and higher scores do not always guarantee that the translation will be perceived as better by human evaluators due to the limitations mentioned above.

BLEU and ROUGE differences

While BLEU is designed for evaluating machine translation, the ROUGE metric is used for evaluating Automatic Text Summarization. Both metrics are based on the comparison of a generated output (translation or summary) to one or more human-generated reference texts. However, ROUGE primarily focuses on recall, while BLEU emphasizes precision. This means that ROUGE measures the extent to which the important information in the reference texts is captured by the generated summary, while BLEU measures how well the generated translation matches the reference translations.

2.4 Machine Learning

Machine learning is a subfield of artificial intelligence that aims to develop algorithms and models that enable computers to learn from data and make predictions or decisions based on that knowledge. In contrast to traditional computer programming, where explicit rules and instructions are written by humans to solve specific problems, machine learning allows computers to learn from data and improve their performance over time without being explicitly programmed.

A more formal definition of machine learning provided by Tom Mitchell in ”Machine Learning” [37] is widely accepted and commonly cited in the field. It provides a formal framework for understanding the key elements of machine learning:

”A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

In other words, machine learning involves improving a computer program’s ability to perform a specific task by exposing it to relevant data (experience E) and measuring its performance on that task (performance measure P). The program is said to have learned when its performance on the task improves as a result of the experience it gains from the data.

This definition highlights the central role of data in machine learning and emphasises the importance of selecting appropriate performance measures to evaluate the effectiveness of the learning algorithm. It also underscores the goal of developing models that can generalize from observed data to make accurate predictions or decisions on new, unseen data.

The diverse ways in which a computer program can enhance its performance on a task can be classified into three distinct learning paradigms. These paradigms showcase the variety of approaches employed by machine learning models to optimize their performance measures.

2.4.1 Learning Paradigms

In machine learning, three main different paradigms for learning have evolved: supervised learning, unsupervised learning, and reinforcement learning. Each of the three paradigms uses a different approach to learn from experience and improve its performance measure.

- **Supervised learning** is the most common type of machine learning, where the goal is to learn from mapping between input and output pairs using labelled data. Labelled data refers to a dataset where each input example is paired with its corresponding output or target, often referred to as a label or annotation. The input data is presented to the model along with its corresponding output labels, and the model is trained to generalize to new, unseen data. In other words, the model learns to predict the output label for a given input by learning from a set of examples where the correct output is known. Examples of tasks where supervised learning is used are image classification or sentiment analysis. Common supervised algorithms used for these tasks include convolutional neural networks (CNN) [38] or support vector machines (SVM) [39].
- **Unsupervised learning** involves learning patterns and relationships in the input data without explicit output labels. The goal is to discover the underlying structure of the data and to find patterns that can be used to make predictions or solve problems. Examples of unsupervised learning include clustering using the K-means algorithm [40], dimensionality reduction with principal component analysis (PCA) [41], and generative modelling using variational autoencoders (VAEs) [42].
- **Reinforcement learning** involves learning through trial and error in an environment, where the goal is to maximize a reward signal. The model learns to take actions that lead to higher rewards by exploring the environment and receiving feedback in the form of a reward or punishment. Reinforcement learning has been used in applications such as robotics with Deep Q-Networks (DQN) [43], playing games using AlphaGo and winning against human agents [44], and recommendation systems employing Proximal Policy Optimization (PPO) [45].

Each of these paradigms possesses its unique strengths and weaknesses, making them suitable for addressing various types of problems. Supervised learning is particularly effective when labelled data is available, while unsupervised learning excels at uncovering patterns and relationships within large, unstructured datasets. Reinforcement learning is well-suited for scenarios where a reward signal can be established, and exploration is necessary for identifying optimal solutions. In practice, many machine learning applications employ a blend of these paradigms to tackle intricate challenges. One such example is ChatGPT [46], which initially undergoes training to comprehend the nuances of natural language structure using unsupervised learning. Subsequently, it is fine-tuned with reinforcement learning to align with human expectations of how a chatbot should interact, effectively combining various learning approaches to achieve a more sophisticated and user-friendly conversational AI experience.

Transfer Learning

Transfer learning is a powerful technique in machine learning where a model pre-trained on one task or dataset is fine-tuned to perform a different but related task or to work with a different dataset. This approach leverages the knowledge acquired by the pre-trained model to achieve better performance and reduced training time on the new task. Transfer learning has proven particularly effective in deep learning and natural language processing (NLP) applications, where large and complex models are trained on massive datasets.

The primary motivation behind transfer learning is the observation that, for many tasks, it is inefficient to train a model from scratch, particularly when a related task has already been solved using a similar model. Instead, the learned features and representations can be transferred from the pre-trained model to the new task, allowing the model to build upon existing knowledge and adapt to the new problem more effectively.

The full process of training a model with transfer learning can be divided into two main stages:

1. **Pre-training:** During this phase, the model is trained on a large-scale dataset, usually containing diverse samples. The goal of this stage is to learn general patterns or features that are relevant to a wide range of tasks [47]. These models, often referred to as "foundation models," serve as a base for further fine-tuning on specific domains. For example, in computer vision, a model might be pre-trained on the extensive ImageNet dataset [48], where it learns to recognize various object categories, textures, and shapes. It is worth noting that pre-training is not always necessary for every application, as many foundation models are publicly shared and available for use. This allows researchers and practitioners to directly fine-tune these models on their specific tasks, saving both time and resources.
2. **Fine-tuning:** In this stage, the pre-trained model is further trained on the target task's dataset, typically smaller than the pre-training dataset [49]. The objective is to adapt the model's knowledge to the specific task by refining its parameters, enabling it to perform well even with limited labelled data [47]. Continuing with the computer vision example, the model pre-trained on ImageNet could be fine-tuned on a smaller dataset like CIFAR-10 [50] to classify specific object categories with high accuracy.

Transfer learning offers several advantages over training from scratch :

- **Reduced training time:** The pre-trained model has already learned fundamental patterns or features in the data, so fine-tuning requires fewer training iterations compared to training from scratch [51]. As a result, transfer learning often leads to faster convergence and reduced training time [47].
- **Improved performance:** Transfer learning often results in better performance on the target task, as the model can leverage the knowledge gained during pre-training, which may not be easily obtainable from the smaller target dataset [47]
- **Domain adaptation:** Transfer learning can facilitate domain adaptation, where the model is trained on a source domain but applied to a related but different target domain [49]. By fine-tuning the pre-trained model on a small amount of labelled data from the target domain, it can effectively adapt to the new domain, mitigating the effects of distribution shifts and enhancing performance on the target task [52].

Transfer learning has played a significant role in the advancements of NLP in recent years, particularly with the introduction of large-scale pre-trained models like BERT [9] and T5 (section 2.7.4). These models have been pre-trained on massive text corpora, enabling them to learn rich language representations that capture diverse linguistic patterns, syntactic structures, and semantic relationships. The pre-trained models can then be fine-tuned on various downstream NLP tasks, such as sentiment analysis, question answering, named entity recognition, and machine translation, resulting in state-of-the-art performance across a wide range of benchmarks

[52].

2.4.2 Training, Validation, and Testing stages

Training, validation, and testing are essential stages in the development of machine learning models. The training phase involves teaching the model on a labelled dataset, aiming to minimize the discrepancy between predicted and actual labels. After training, the model is validated using a separate dataset to assess its performance on unseen data. Performance metrics obtained during validation are utilized to adjust model parameters and compare various models to enhance performance. Testing is the final stage of model development, where the model's performance is assessed on a previously unseen portion of the dataset to estimate its effect on new data.

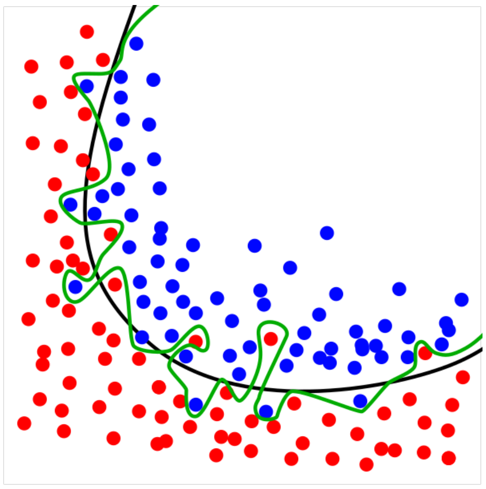


Figure 2.4: An illustration of overfitting (green line) and a well-balanced model (black line). Modification of [53].

Overfitting can occur when a model becomes excessively adapted to the training data, capturing noise or random fluctuations and reducing its ability to generalize effectively to new, unseen data. To prevent overfitting, it is essential to ensure that the training and validation datasets are representative of the unseen data the model will encounter in the test set and real-world applications. In contrast, underfitting may take place when a model fails to learn enough from the training data, resulting in suboptimal performance. In Figure 2.4, the black line represents an appropriate balance between underfitting and overfitting, while the green line demonstrates an overfit model that has not generalized well.

A common practice is to split the dataset into 60% training, 20% validation, and 20% test subsets. However, the precise proportions depend on the dataset size and the specific problem being addressed.

One challenge with dividing the data into fixed subsets is that the evaluation might be sensitive to the particular selection of training and validation data. To address this issue, a technique called k-fold cross-validation can be employed [54]. In k-fold cross-validation, the training set is partitioned into k equally-sized subsets, or "folds." The model is trained k times, with each iteration using k-1 folds for training and the remaining fold for validation. The performance measure is then averaged across the k iterations. By averaging the performance over multiple training-validation splits, cross-validation offers a more reliable estimate of the model's generalization capabilities.

2.5 Artifical Neural Networks

Inspired by the workings of the human brain, artificial neural networks (ANNs) seek to replicate its complex processes through computational means [55]. The brain consists of billions of interconnected neurons, which transmit and process information via electrical and chemical signals. ANNs attempt to model these connections and interactions through a series of nodes and layers, designed to recognize patterns and make decisions based on input data.

At the core of ANNs are artificial neurons, or nodes, which receive input from various sources, process the information, and pass it along to other nodes within the network. These nodes are organized into layers, with an input layer receiving the initial data, hidden layers performing complex computations, and an output layer producing the final result or prediction. When each node in the previous layer is connected to every node in the next layer, the ANN is often referred to as a fully-connected neural network or multilayer perceptron (MLP).

Figure 2.5 shows a small MLP with two inputs, two hidden layers with 4 neurons in each layer and one output neuron. The black arrows depict the weights for each neuron in the MLP. Each neuron in the MLP has an activation function inside it which adds non-linearity to the values, making it possible for the network to learn non-linear and more complex relationships. For each layer, there is also a bias vector, which is an additional parameter that gets added to the output of the neurons. The bias vector helps shift the values in the activation function along the horizontal axis similar to the effect of the intercept (b) in the equation for a straight line: $y(x) = mx + b$.

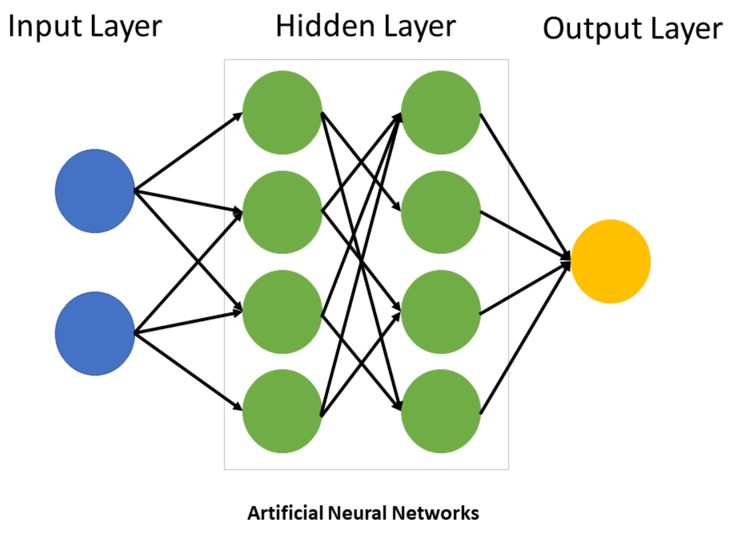


Figure 2.5: A simple artificial neural network with 2 inputs, 2 hidden layers with 4 neurons each, and a single output neuron. Note that bias nodes are not depicted here. Modification of [56].

In the process of passing information between layers, the weights come into play. Each connection between a node in one layer and a node in the next layer has a weight associated with it. To determine the input value for a node in the hidden layer, the values from the previous layer's nodes are multiplied by their respective connection weights and then summed together. This process can be represented mathematically as:

$$\mathbf{a}^{(l)} = f\left(\mathbf{W}^{(l-1)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}\right) \tag{2.7}$$

where $\mathbf{a}^{(l)}$ is the vector of activation values for all neurons in layer l, f is the activation function for the neurons in layer l (different activation functions commonly used are covered in Section 2.5.2, $\mathbf{W}^{(l-1)}$ is the weights connecting layer $l-1$ (the previous layer) and $\mathbf{b}^{(l)}$ is the bias vector for the neurons in layer l.

The weights and biases are the adjustable elements of an MLP, which are adjusted during the training process to optimize the network's performance on a given task. The goal of training is to minimise the discrepancy between the network's predictions and the true target values, usually measured by a loss function which measures the difference between the MLPs output and the correct values. To achieve this, the network must learn how to adjust its weights and biases in response to the input data, such that the error between its predictions and the target values is minimised. One of the most widely used algorithms for this purpose is the backpropagation algorithm, which aims to do this by iteratively computing the gradients of the loss function with the goal of minimizing it.

2.5.1 Backpropagation

The backpropagation algorithm is a powerful and efficient method for training artificial neural networks, specifically MLPs [57]. It is a supervised learning technique that aims to minimise the error between the network's predictions and the true target values by adjusting the weights and biases of the network. The key idea behind backpropagation is to compute the gradient of the loss function with respect to each weight and bias by applying the chain rule for the derivatives. The computed gradients are then used to update the network parameters in a way that reduces the overall error.

The backpropagation algorithm consists of two main phases: the forward pass and the backward pass. In the forward pass, the input data is propagated through the network, layer by layer, using the formula in Equation 2.7. Once the output values are obtained, a suitable loss function is chosen based on the specific problem being solved and the nature of the target values. Common loss functions include the mean squared error (MSE) for regression tasks [58, pg. 37] and the cross-entropy loss for classification tasks [58, pg. 471]. The selected loss function quantifies the discrepancy between the network's predictions and the true target values.

In the backward pass, the gradient of the loss function with respect to each weight and bias is calculated. This process starts from the output layer and moves towards the input layer, applying the chain rule for derivatives, which allows the gradient to be computed efficiently through a series of local computations. Once the gradients are obtained, they are used to update the weights and biases of the network according to an optimization algorithm, such as gradient descent or a variant thereof. These updated parameters are adjusted by a factor known as the learning rate, which determines the step size in the optimization process. Choosing a suitable learning rate is crucial, as it ensures that the network converges to a solution without overshooting or getting stuck in local minima.

Despite not being covered in this section, the same principles of backpropagation, including the forward and backward passes, also apply to more advanced neural network architectures such as Recurrent Neural Networks (RNNs) and Transformers.

Batch Sizes and Mini-batch Gradient Descent

When training neural networks with backpropagation, multiple samples of data can be passed through the network at the same time using mini-batches, which are small subsets of the training dataset. Mini-batch gradient descent offers a balance between the stability of computing gradients with batch gradient descent, which uses the entire training dataset for a single parameter update, and the computational efficiency of stochastic gradient descent, which updates model parameters with a single training example at a time. By processing multiple samples at once, mini-batch gradient descent takes advantage of the averaging effect, leading to more stable convergence and allowing modern hardware, such as GPUs, to be efficiently utilized for faster training time [59].

One of the primary advantages of using mini-batches is the reduction of noise in gradient estimates. While stochastic gradient descent updates model parameters based on a single training example, introducing high variance in the gradient estimates, mini-batch gradient descent computes the average gradient over a small subset of samples. This averaging effect results in more stable and accurate gradient estimates, promoting smoother convergence and reducing the chances of getting stuck in local minima or saddle points [60].

Moreover, mini-batch gradient descent can potentially strike a balance between optimization and generalization performance. Smaller batch sizes expose the model to a greater variety of training examples with each update, which may lead to better generalization and faster convergence. On the other hand, larger batch sizes yield more stable gradient estimates and can utilize computational resources more efficiently. However, it is important to avoid excessively large batch sizes, as they can negatively impact the model's generalization capabilities due to over-optimization on the training set [59].

In practice, to find an optimal batch size that strikes a balance between training speed and generalization performance, a common approach, as recommended in Google's fine-tuning playbook [61], is to choose the largest batch size that can be accommodated within the memory constraints of the hardware being used. Higher batch sizes lead to increased memory consumption because more training examples are processed simultaneously, which requires additional memory to store intermediate values, such as activations and gradients, during both the forward and backward passes of the neural network. As a result, choosing a larger batch size can maximize computational efficiency and take advantage of the parallel processing capabilities of modern hardware, such as GPUs.

2.5.2 Activation Functions

Activation functions play a crucial role in artificial neural networks, as they introduce non-linearity into the network, allowing it to learn complex and non-linear relationships between input and output variables. Without activation functions, the network would essentially be a linear model, which would limit its ability to learn more intricate patterns in the data.
The sigmoid function is a widely used activation function, especially in earlier ANN models [62]. The sigmoid function is defined as:

$$f(x) = \frac{1}{1+e^{-x}}, \tag{2.8}$$

where x is the input to the function. The sigmoid function maps the input to a value between 0 and 1, making it suitable for binary classification problems. However, it suffers from the vanishing gradient problem, which can slow down the learning process during backpropagation, as the gradient becomes very small [63].

The rectified linear unit (ReLU) [64] is another popular activation function, defined as:

$$f(x) = \max(0, x). \tag{2.9}$$

The simplicity of the ReLU function allows for efficient computation and addresses the vanishing gradient problem, as its gradient is either 0 or 1. However, the ReLU function can lead to dead neurons, which can hinder the learning process.

The Gaussian Error Linear Unit (GELU) [65] is a more recent activation function that has gained popularity. It is the activation function used by modern NLP models such as T5 [3]and GPT-2 [18]. The GELU function is defined as:

$$\text{GELU}(x) = \frac{1}{2}x\left(1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right) \tag{2.10}$$

where x is the input to the function and erf represents the Gaussian Error function. GELU is a smooth approximation of the rectifier function and has been shown to perform well in deep learning models. Its smoothness allows for better gradient flow, which can be beneficial in the training process.

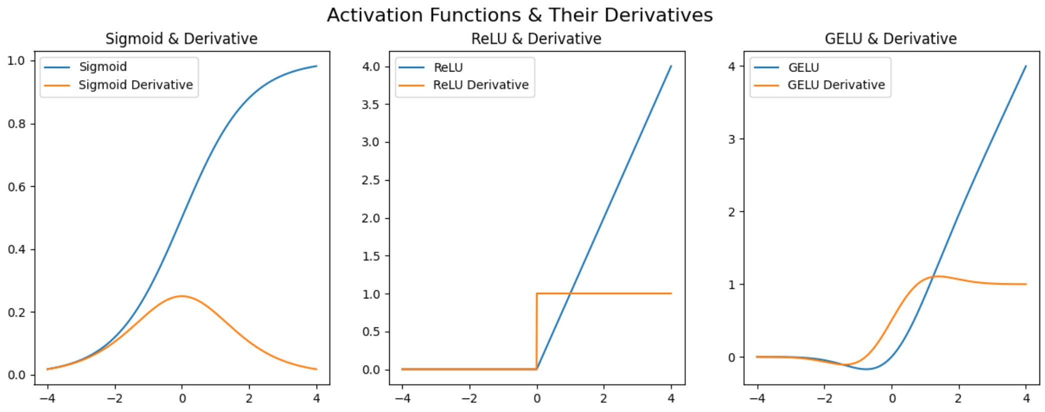


Figure 2.6: Plots of the sigmoid, ReLU, and GELU activations, along with their corresponding derivatives. The x-axis represents the input to the activation function, while the y-axis represents the output.

Figure 2.6 shows plots of the sigmoid, ReLU, and GELU functions, along with their corresponding derivatives. The x-axis represents the input to the activation function, while the y-axis represents the output. These plots provide a visual representation of how each function transforms the input into its corresponding output. The derivatives of these functions represent the rate of change of the output with respect to the input and are essential for the backpropagation algorithm during training.

Softmax

For multi-class classification problems, where the goal is to assign a probability to one of several possible classes, the softmax function [66] is commonly used as the activation function for the neurons in the output layer of the neural network. The softmax function transforms a vector of input values into a probability distribution over the possible classes, where each class is assigned a probability between 0 and 1, and the sum of all probabilities is equal to 1. This allows the neural network to output a probability for each possible class, which can be used to make predictions.

The softmax function takes a vector of input values $\mathbf{x} = (x_1, x_2, ..., x_K)$ and produces a vector of output values $\mathbf{y} = (y_1, y_2, ..., y_K)$, where N is the number of classes. The output values are calculated as follows:

$$y_i = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}, \tag{2.11}$$

where i indexes the individual elements of the input vector. The denominator in the equation represents the sum of the exponential values of all the input elements. The numerator is the exponential value of the i-th input element. The softmax function essentially normalizes the exponential values of the input elements to obtain a probability distribution over the classes.

2.5.3 Dropout

Dropout is a widely used regularization technique in deep learning models, particularly in neural networks [67]. It serves as an effective method to mitigate the overfitting issue, leading to improved generalization on unseen data.

During the training phase, the dropout technique functions by randomly setting a proportion of neurons in a layer to zero, effectively "dropping out" their contribution to the forward and backward passes. This process is visualized in Figure 2.7, where neurons with red crosses are dropped out. This approach introduces noise to the model's training, encouraging it to learn more robust representations of the data and preventing it from becoming overly reliant on any single neuron. Dropout is only applied during training, and all neurons are retained during the inference phase, ensuring the model's full capacity is utilized when making predictions on new data.

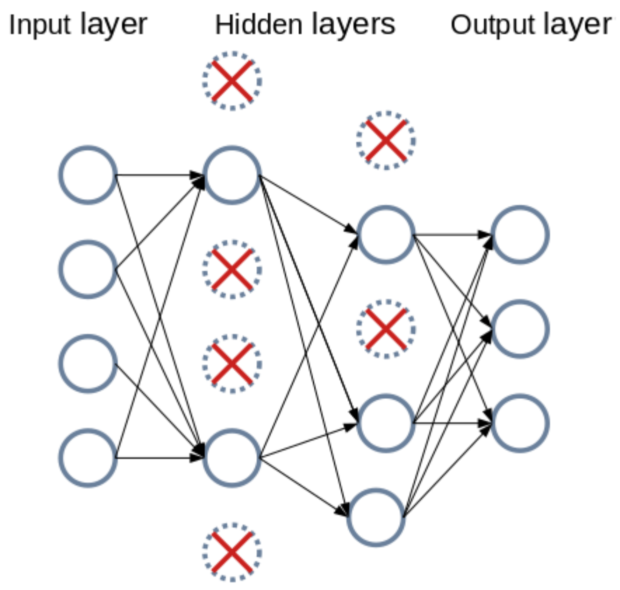


Figure 2.7: Visualization of dropout. Neurons with red crosses represents neurons that have been dropped (values set to zero). Modification of [68]

The dropout rate, which determines the proportion of neurons to be dropped out, is a critical hyperparameter that can be tuned for optimal performance. Typically, dropout rates between 10% and 50% are found to be effective in practice [69]. However, it is essential to experiment with different dropout rates to find the optimal value for a specific task or dataset.

Dropout is particularly effective when used in conjunction with other regularization techniques, such as weight decay [70] and early stopping [71, pg. 239]. The combination of these techniques can significantly improve the generalization capability of deep learning models, resulting in better performance on unseen data. Moreover, dropout has been shown to have an implicit ensemble effect, as it can be interpreted as training an ensemble of sub-networks with shared weights, where the final prediction is essentially an averaging of the predictions made by these sub-networks.

2.5.4 Layer Normalization

Layer normalization [72]is a technique used to improve the performance and training stability of neural networks, especially deep models with many hidden layers. It helps alleviate the internal covariate shift problem, which occurs when the distribution of input values to a given layer changes during training. Layer normalization can be applied independently to each hidden layer in the network, usually after the linear transformation and before the activation function, ensuring a consistent distribution of input values to the activation function. This facilitates faster learning and improved generalisation.

The layer normalization operation is defined as follows. Given an input vector **x**, the output **y** is computed as:

$$\mathbf{y} = \frac{\mathbf{x} - \mathbf{E}[\mathbf{x}]}{\sqrt{\mathbf{Var}[\mathbf{x}] + \epsilon}} * \gamma + \beta \tag{2.12}$$

The ϵ term is a small positive constant added to the variance to ensure numerical stability and γ and β are learnable scaling and shifting parameters, which allow the model to adjust the output distribution during training. $\mathbf{E}[\mathbf{x}]$ and $\mathbf{Var}[\mathbf{x}]$ are the expected value and variance of $\mathbf{x}$.

Layer normalization is conceptually similar to batch normalization [73], which is another normalization technique used in deep learning. The key difference between the two is that while batch normalization normalizes the input values across a batch of samples, layer normalization normalizes the input values for each sample independently. This makes layer normalization more suitable for tasks with variable batch sizes or where data dependencies within a batch are not desirable, such as in recurrent neural networks (RNNs).

2.5.5 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) [74] were developed to address the limitations of traditional feedforward neural networks in processing sequential data. While ANNs, like MLPs, can efficiently model complex relationships between input and output variables, they lack the ability to capture temporal dependencies or the order of the input data. This limitation hinders their effectiveness in tasks involving time-series data, natural language processing, and other applications that rely on the sequential nature of the input.

RNNs were introduced as a solution to this problem by incorporating a memory mechanism into the network (often referred to as the hidden state), allowing it to maintain information about the previous input states. This memory enables RNNs to capture the temporal dependencies and the order of the input data, making them suitable for a wide range of applications involving sequential data.

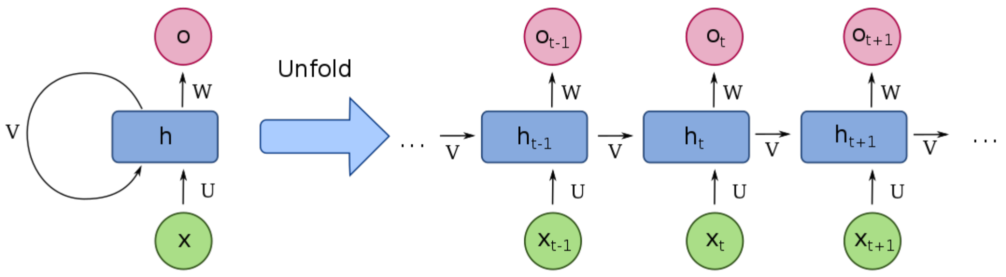


Figure 2.8: A depiction of a single-unit RNN. The progression from bottom to top represents input state, hidden state, and output state. The network weights are denoted by U, V, and W. The left side features a compressed diagram, while the right side illustrates the unfolded version of the network. Modification of [75].

The architecture of an RNN can be visualized as a series of repeating modules, where each module represents a time step in the sequence, as shown in Figure 2.8. In the compressed diagram on the left, the circular unit represents the hidden state, which maintains information about the order and temporal dependencies of the input data. The unfolded version of the RNN on the right illustrates the process for each input in the sequence. At each time step, the module takes the current input (bottom), along with the hidden state from the previous time step, and produces an output (top) and an updated hidden state. The weights of the network, represented by U, V, and W, are shared across all time steps, allowing the network to learn patterns and dependencies in the sequence data.

Challenges and Variants of RNNs

Despite their ability to model sequential data and capture temporal dependencies, RNNs face several challenges and drawbacks that limit their performance and applicability in certain scenarios. One of the primary challenges encountered in training RNNs is the vanishing and exploding gradient problem [63]. This issue occurs due to the repeated application of weight matrices during backpropagation through time (BPTT), which can cause gradients to either vanish (approach zero) or explode (grow exponentially). Vanishing gradients make it difficult for the network to learn long-range dependencies in the input data while exploding gradients can lead to instability in the training process and poor generalization.

Although RNNs are designed to capture temporal dependencies in sequences, they often struggle with learning long-range dependencies due to the vanishing gradient problem. This limitation makes it difficult for RNNs to model complex relationships and contextual information in tasks involving long input sequences, such as machine translation or long document summarization.

The recurrent nature of RNNs inherently limits their ability to leverage parallel computation. As the hidden state of each time step depends on the previous time step, the network must process the input sequence sequentially, which can result in slower training and inference compared to feedforward architectures like Convolutional Neural Networks (CNNs) or Transformer-based models.

Additionally, the hidden state in RNNs serves as the primary memory mechanism, enabling the network to retain information about previous inputs. However, this memory capacity is limited by the size of the hidden state and the network's ability to maintain information across long sequences. As a result, RNNs can struggle with tasks that require the retention of a large amount of contextual information or involve the processing of long input sequences.

To address these challenges, researchers have proposed several variants and modifications of the basic RNN architecture. Some of these include the Long Short-Term Memory (LSTM) [76] and Gated Recurrent Unit (GRU) [77] networks, which introduce gating mechanisms to alleviate the vanishing gradient problem and improve the learning of long-range dependencies. Additionally, the development of the Transformer architecture [7] and its derivatives has further advanced the field of sequence modelling by introducing attention mechanisms that enable more efficient handling of long-range dependencies and parallel computation, surpassing the capabilities of traditional RNNs.

Encoder-Decoder Models

A considerable number of RNNs, particularly those utilized in sequence-to-sequence tasks, make use of the encoder-decoder architecture [77]. This architecture adeptly addresses the challenges associated with tasks that involve input and output sequences of varying lengths or structures, as seen in machine translation and summarization. The primary objective of using encoder-decoder models is to enable the learning of a fixed-length representation, called the Context Vector, from variable-length input sequences, and subsequently decode this representation to generate variable-length output sequences. To achieve this, the model is segmented into two main components: the encoder and the decoder. Both components typically comprise RNN layers, such as LSTMs or GRUs.

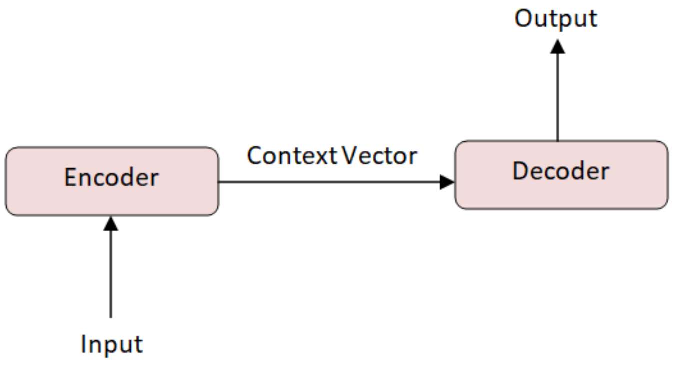


Figure 2.9: A basic encoder-decoder model [78]. The input sequence is processed by the encoder, resulting in a context vector that is then used by the decoder to generate the output sequence.

Encoder

The encoder is responsible for processing the input sequence and compressing it into a fixed-length latent representation, also known as the context vector. This context vector is designed to capture the essential information from the input sequence, allowing the decoder to generate an appropriate output sequence based on it.

As illustrated in Figure 2.9, the encoder receives the input sequence and processes it to create a context vector. This is typically done by processing the sequence one element at a time using RNN layers, though this detail is abstracted away in the figure.

Decoder

The decoder, also depicted in Figure 2.9, is responsible for generating the output sequence based on the context vector provided by the encoder. It starts with an initial hidden state, usually set to the context vector. At each step, the decoder generates an output element and updates its hidden state based on the current input and hidden state.

The output generated by the decoder at each step is typically passed through a softmax function, which converts it into a probability distribution over all possible output elements. The element with the highest probability is then chosen as the predicted output for the current step. This process continues iteratively until either a predefined maximum output length is reached, or a special end-of-sequence token is produced, signalling that the model should cease generating further output.

The encoder-decoder architecture allows RNNs to handle complex sequence-to-sequence tasks by learning to map input sequences to output sequences through a fixed-length latent representation. This approach has been highly successful in a wide range of applications and has been further enhanced by the incorporation of attention mechanisms, which enable the model to dynamically focus on different parts of the input sequence during the decoding process, improving its ability to capture long-range dependencies and handle complex input-output relationships.

2.6 Token Decoding

When generating text for NLP tasks such as machine translation or summarization, the output of the model is normally a token, where the token is an integer that represents a word, sub-word, or character (Section 2.1.1). The token is then generally concatenated to the input and used for generating the next tokens in the output sequence. This loop continues until the model itself decided it has reached the end when it outputs a stop token. Models which generate sequences like this where models predict the next element in a sequence by conditioning on the previously generated elements are often referred to as autoregressive models [79, pg. 117] and the process of selecting tokens and converting tokens back into human-readable characters or words is known as token decoding.

Figure 2.10 illustrates an example of an autoregressive model translating the English sentence ”I am driving a car” into Norwegian. The generation proceeds until the model outputs a stop token, which is often ”EOS” (End Of Sequence) or ”</s>”. While the example features an encoder-decoder model, similar to the T5 model used in this thesis, the principles of token generation apply to decoder-only models like GPT-2 [18] as well.
The final layer of the decoder in these generative models is normally a softmax function. The softmax function in the output returns a vector with the probability of each token in the vocabulary. When selecting the token at each generation step, the easiest thing to do is to choose the token with the highest probability. This technique of picking tokens is called greedy decoding [26].

2.6.1 Greedy Decoding

Greedy decoding, while computationally simple and efficient, has several drawbacks that can limit the quality and diversity of the generated text [80]. These shortcomings are mainly due to its deterministic nature and focus on short-term rewards rather than long-term coherence.

One significant issue with greedy decoding is the lack of diversity in the generated sequences. Since it always selects the token with the highest probability at each generation step, the pro-

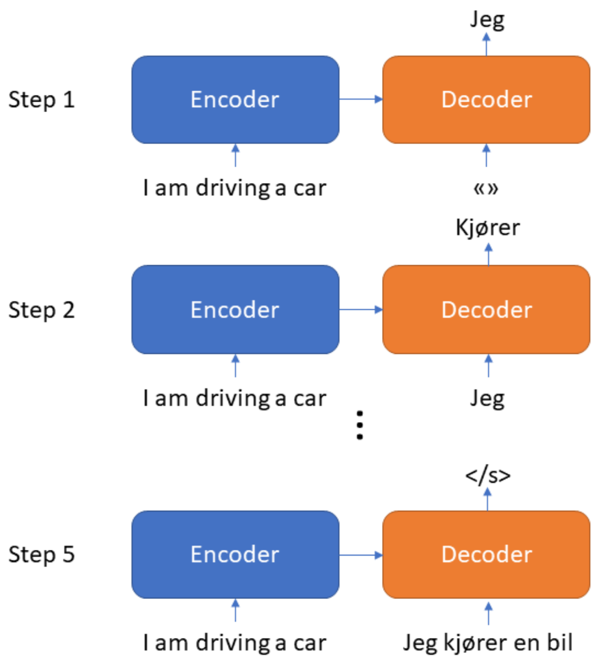


Figure 2.10: Example of autoregressive translation of the sequence ”I am driving a car” from English to Norwegian using an encoder-decoder model.

duced sequences tend to be repetitive and lack variety. This can result in monotonous and predictable outputs, which might not adequately capture the richness and complexity of the source material or the intended message.

Moreover, greedy decoding is a local optimization method, meaning it focuses on maximizing the immediate reward rather than considering the overall quality of the generated sequence. As a result, it can sometimes lead to suboptimal solutions, where the generated text may appear coherent in the short term but fail to accurately convey the intended meaning or structure of the original content.

Another limitation of greedy decoding is its inability to recover from errors or explore alternative paths once a token has been selected. If a poor choice is made early in the generation process, the subsequent tokens will be generated based on that choice, potentially leading to a chain of errors and a poor-quality output.

To address these drawbacks, various alternative decoding strategies have been developed, such as beam search, top-k sampling, and nucleus sampling. These methods introduce an element of exploration and randomness into the token selection process, aiming to improve the diversity, coherence, and overall quality of the generated text by considering multiple potential sequences and balancing the trade-off between exploration and exploitation.

2.6.2 Advanced Token Decoding

These methods introduce a balance between deterministic and stochastic approaches in the token selection, exploring alternative sequences, and mitigating common issues such as repetition and suboptimal outputs. By leveraging these advanced techniques, researchers and practitioners can enhance the performance of sequence-to-sequence models across a wide range of natural language processing tasks and applications. Some of these different techniques commonly utilized in machine learning research include:

Beam search

Beam search [81] is a heuristic search algorithm used in sequence-to-sequence models for finding the most likely output sequence. Beam search can be adapted to impose additional constraints on the generated sequences, such as limiting the length of the output, avoiding repeating n-grams, and filtering out undesirable words while still being able to generate coherent sequences.

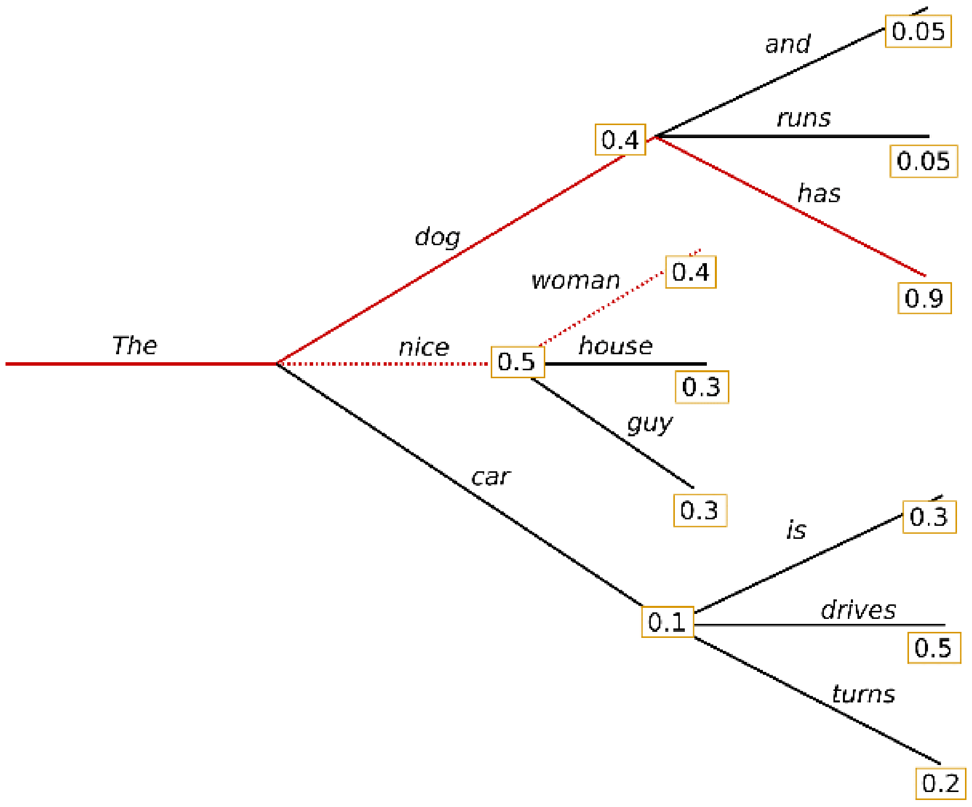


Figure 2.11: Example of the process of a beam search with a beam width of 3. The dotted line represents the sequence greedy decoding would generate and the solid red line is the most likely sequence generated with beam search. Reprinted from [82] with permission from Patrick von Platen.

Figure 2.11 clearly illustrates beam search's advantage over greedy decoding in generating a more likely sequence. In this example, greedy decoding, represented by the dotted red line, generates the sequence *The nice woman* with a conditional probability of $P(nice|The) \cdot P(woman|The\ nice) = 0.2$. In contrast, beam search, depicted by the full red line, produces the sequence *The dog has* with a higher conditional probability of $P(dog|The) \cdot P(has|The\ dog) = 0.36$. This demonstrates the advantage of beam search over greedy decoding, as it considers multiple candidate sequences and ultimately selects the one with the highest overall likelihood.

The number of candidates beam search considers at each step is known as the beam width or the number of beams. As the beam width increases, the algorithm explores a larger portion of the search space, which can lead to better solutions. However, this increased exploration comes at the cost of higher computational complexity, as more candidate sequences need to be evaluated at each time step. In practice, selecting an appropriate beam width involves balancing the need for high-quality output sequences with the available computational resources and time constraints.

2.6.3 Sampling

Sampling-based decoding techniques provide an alternative approach to token selection in sequence-to-sequence models by incorporating stochasticity into the decision-making process. Instead of deterministically selecting the most likely token at each time step, as in greedy decoding, or exploring a fixed number of candidate sequences, as in beam search [81], sampling-based methods generate tokens by drawing from a probability distribution over the vocabulary [83]. This probabilistic approach allows the model to explore a more diverse range of output sequences, potentially capturing unexpected or creative solutions that deterministic methods might overlook [18]. Moreover, sampling-based techniques can alleviate issues such as repetitive output and overconfidence by introducing variability and uncertainty into the generated sequences [84].

While traditional sampling decoding works by sampling from the softmax probabilities across all of the tokens in the vocabulary, recent developments have shown that by modifying the probability distributions, even better sequences can be generated. Some of these advanced sampling techniques include:

- **Top-k**: Top-k sampling [84] is a decoding strategy in which, at each time step, the model selects the next token by sampling from the k most probable tokens. It introduces some randomness in the generated output, promoting diversity while still constraining the search space to more likely token choices.
- **Nucleus sampling**: Nucleus sampling, or top-p sampling [84], is a decoding method that selects tokens from the top-p most probable tokens, where p is a probability mass threshold. This approach balances flexibility and determinism, allowing for diverse and creative outputs while maintaining control over the generated text's quality.
- **Temperature**: Temperature [84] is a hyperparameter in probabilistic sampling techniques that controls the degree of randomness in token selection. A high temperature leads to more random token selection, while a low temperature makes the model more deterministic, favouring tokens with higher probabilities.
- **Repetition penalty**: Repetition penalty [85] is a technique used during the decoding process to penalize the model for generating repetitive sequences. It involves modifying the probability distribution over tokens at each time step to decrease the likelihood of selecting tokens that have already appeared in the generated output, thus encouraging more diverse and non-repetitive text.
- **Eta cutoff**: Eta cutoff is an advanced token selection technique that aims to improve the quality of generated text by estimating a subset of the support of the true distribution in a language model [86]. It operates by setting an entropy-dependent probability threshold, which helps identify a more appropriate set of tokens to consider during the decoding process. Compared to other truncation sampling methods, eta cutoff generates more plausible and diverse long text sequences, effectively breaks out of repetition, and demonstrates more reasonable behaviour on a range of test distributions.

The various token sampling techniques discussed in this section offer numerous possibilities for enhancing decoding strategies in sequence-to-sequence models. For instance, combining beam search with temperature scaling can strike a balance between exploration and exploitation during the search process. Additionally, incorporating repetition penalties into top-k or nucleus sampling can reduce redundancy and improve sequence diversity.

However, it is essential to recognise that the optimal decoding parameters may vary depending on the specific task. Repetition is not always undesirable, as some tasks may benefit from recurring patterns or phrases. Moreover, not all applications require creative outputs, with some prioritising accuracy and consistency over novelty. By customising the combination of token sampling techniques to suit each task's unique requirements and carefully integrating these methods, researchers and practitioners can achieve an equilibrium between output quality, diversity, and computational efficiency. This tailored approach enables the development of versatile models that cater to a wide range of natural language processing tasks and applications.

Furthermore, it is important to note that during the training phase of sequence-to-sequence models, loss values are calculated at the individual word level rather than considering the entire sequence. This means that the model evaluates the correctness of its predictions for each token independently, without accounting for the overall coherence or structure of the generated sequence. As a result, using advanced decoding techniques like beam search or sampling-based methods during training would not be compatible with the loss calculation approach, since these methods focus on generating sequences from a more global perspective. Instead, the optimal token decoding parameters can be optimized after the model has been trained.

2.7 Transformers

Transformers represent a groundbreaking deep learning architecture first introduced by Vaswani et al. in their 2017 paper, "Attention is All You Need" [7]. Since its inception, the transformer model has become a widely adopted architecture for numerous NLP tasks. One of its main strengths lies in its ability to handle input and output sequences of varying lengths, which is especially valuable in tasks such as abstractive summarization and machine translation. Transformers have been successful in addressing some of the limitations and challenges associated with RNNs, particularly in capturing long-range dependencies and enabling parallel computation.

In recent years, the success of transformers has extended beyond natural language processing and into the realm of computer vision with the development of Vision Transformers (ViTs). ViTs, proposed by Dosovitskiy et al. in their 2020 paper "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale" [87], adapt the transformer architecture to accommodate images by dividing them into fixed-size, non-overlapping patches and linearly embedding each patch into a flat vector, which then serves as input for the transformer model.

While there are notable similarities between the structures of Vision Transformers used in computer vision tasks and those employed in NLP, this section will concentrate on the application of transformer models within the context of NLP tasks.

The cornerstone of the transformer architecture is the self-attention mechanism, which empowers the model to assess the relative significance of each word within a sequence. This mechanism facilitates the model's ability to efficiently capture long-range dependencies and contextual information, outperforming traditional recurrent neural networks in this regard. By leveraging the self-attention mechanism, transformers are able to mitigate the vanishing gradient problem that plagues RNNs, enabling more effective learning of long-range dependencies and improving the overall performance of various NLP tasks. Additionally, the transformer architecture is highly parallelizable, allowing for faster training and inference compared to the inherently sequential nature of RNNs.

2.7.1 Attention

The attention mechanism, a central component of the transformer architecture, empowers models to assess the importance of different words within a sequence by considering the context. Attention weights represent the relative significance of each word, influencing their contribution when generating the output. By emphasizing words with higher attention scores, the model is steered to focus on input segments that are most relevant to the task. This mechanism enables the model to capture long-range dependencies and contextual information more effectively than traditional RNNs.

The evolution of attention mechanisms in neural networks can be traced back to content-based attention from Neural Turing Machines [81]. In this early work, attention was used to access memory based on the cosine similarity between a query and memory content. The concept was further developed and introduced into RNNs with additive attention, also known as Bahdanau attention, in a groundbreaking paper by Bahdanau et al. [35]. This innovative mechanism allowed models to focus on specific parts of the input sequence when generating output by calculating the alignment between the hidden states of the encoder and decoder. The attention landscape advanced further with the introduction of the Transformer architecture by Vaswani et al. in "Attention is All You Need" [7], proposing the scaled-dot-product attention mechanism. This highly efficient approach facilitated the development of large-scale models capable of handling long-range dependencies and complex patterns across various NLP tasks, laying the foundation for modern language models like GPT and BERT.

The attention weights are computed using the scaled dot-product attention function, which involves three parameters: queries Q, keys K, and values V. Instead of being derived through backpropagation, these parameters consist of values from the previous layer in a neural network. Although Q, K, and V are identical, their distinct roles in scaled dot-product attention allow them to represent different aspects. Q enable the model to identify the parts of the input sequence it should focus on, essentially acting as questions posed by the model to uncover pertinent input information. K, conversely, help the model match queries with corresponding parts of the input sequence, serving as labels or tags for different segments of input data. Lastly, V hold the actual information of the input sequence, and when a query finds a matching key, the model uses the corresponding value to represent that section of the input sequence in the output. With the three parameters in place, along with d_k representing the dimensions of K, scaled dot-product attention is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \tag{2.13}$$

As illustrated in Figure 2.12, the scaled dot-product attention function computes the compatibility between each query and key in the given sequence. Taking the dot product of the query and key and dividing it by the square root of the key's dimension d_k normalizes the magnitude of the values, improving gradient flow during training. The resulting compatibility scores are then processed through a softmax function, converting them into attention weights that indicate the relative importance of each word in the sequence. These attention weights are multiplied by the value vectors (V), and the resulting weighted sum serves as the output of the self-attention mechanism, effectively capturing context-aware representations of the input sequence.

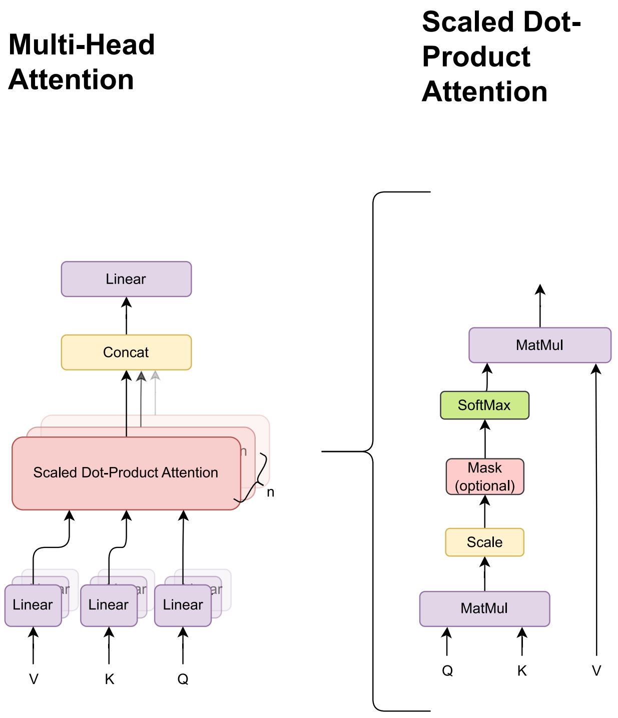


Figure 2.12: Illustration of the multi-head attention block (left) and scaled dot-product attention (right), highlighting their connections and substructures within the transformer architecture. Illustration with modification from [7].

In the transformer architecture, an advanced version of the attention mechanism known as multi-head attention is utilized. Multi-head attention consists of h parallel scaled dot-product attention layers, or ”heads,” each with its unique learned linear projections for queries, keys, and values. This mechanism can be seen on the left side of Figure 2.12.

For each $head_i$, the scaled dot-product attention function is applied to the linearly projected queries, keys, and values. We can represent the computation of each $head_i$ as follows:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.14}$$

In this equation, W_i^Q, W_i^K, and W_i^V are the learned weight matrices responsible for projecting the input queries, keys, and values, respectively. The multi-head attention is then defined by concatenating all the $head_i$ and applying an additional learned linear projection:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{2.15}$$

Here, W^O is the learned weight matrix responsible for projecting the concatenated output.

Multi-head attention, compared to single-head attention, offers several advantages in capturing various aspects of the input sequence and enhancing the expressiveness and flexibility of the attention mechanism:

- Diversity in representation: Multi-head attention allows the model to learn different attention patterns across multiple heads. This enables the model to capture diverse aspects of the input sequence, such as syntactic and semantic information, at various positions, leading to a richer and more versatile representation.
- Parallelism: Since multi-head attention operates on several heads in parallel, the model can efficiently learn and process different attention patterns simultaneously. This can lead

to better computational efficiency, especially when implemented on parallel hardware, such as GPUs or TPUs.

2.7.1.1 Scalability Challenges in Attention Mechanisms

The attention mechanism has been widely adopted for its ability to capture long-range dependencies effectively. However, its scalability with longer context windows (i.e., longer input sequences consisting of more tokens) is an area of concern due to its quadratic complexity in terms of time and memory.

The original implementation of the attention mechanism calculates attention scores for each pair of positions in the input sequence, which means that the number of attention scores increases quadratically with the length of the input sequence. Specifically, for an input sequence of length n, there are n^2 pairwise attention scores. Consequently, both the time complexity and memory complexity of the attention mechanism is $O(n^2)$.

This quadratic complexity presents challenges when processing long input sequences, as it can lead to increased computational time and memory usage. This limitation has motivated the development of various techniques to address the scalability issue while maintaining the ability to capture long-range dependencies effectively.

These techniques are based on the idea that not every single token in the attention window is as relevant at every step. Longformer [88] introduces a sliding window self-attention mechanism that reduces the complexity of self-attention from $O(n^2)$ to $O(n \cdot w)$, where w is the average window size. It also employs global attention for a few selected tokens, which allows the model to capture long-range dependencies. A different approach, Linformer [89] employs low-rank approximation to the self-attention weights to reduce the computational complexity to $O(n)$.

Both Longformer and Linformer offer a unique approach to addressing the quadratic complexity challenge posed by the original attention algorithm. However, it is essential to note that the reduction in time complexity may come at the expense of overall performance or model quality. In response to this trade-off, a more recent development, FlashAttention [90], has emerged as a highly popular attention implementation due to its optimized, highly performant nature based on optimizing write operations to the GPU memory. The implementation is lossless while maintaining the original $O(n^2)$ complexity. The paper reports speedups of 15% on BERT with a sequence length of 512 compared to the state-of-the-art and a 3x speedup on GPT-2 [18] with a sequence length of 1K.

2.7.2 Transformer Architecture

The Transformer's overall architecture is relatively straightforward. A visual representation of the transformer architecture is displayed in Figure 2.13. The model comprises N encoders and N decoders stacked on top of each other, as indicated by "N" in the figure. Several transformer models that have gained popularity since their release share the same building blocks, but differ in architecture. For instance, while BERT [9] only employs the encoder part, GPT models [2] use only the decoder. However, in this section, the focus will be on the original encoder-decoder architecture [7].

Before the inputs and outputs get sent into the encoder and decoder, the tokens get transformed into dense vectors by a word embedding layer. After the input sequences are transformed into dense vectors, the next step is to apply positional encoding. Positional encoding is a technique used to add positional information to the input sequences in a sequence-to-sequence model. It

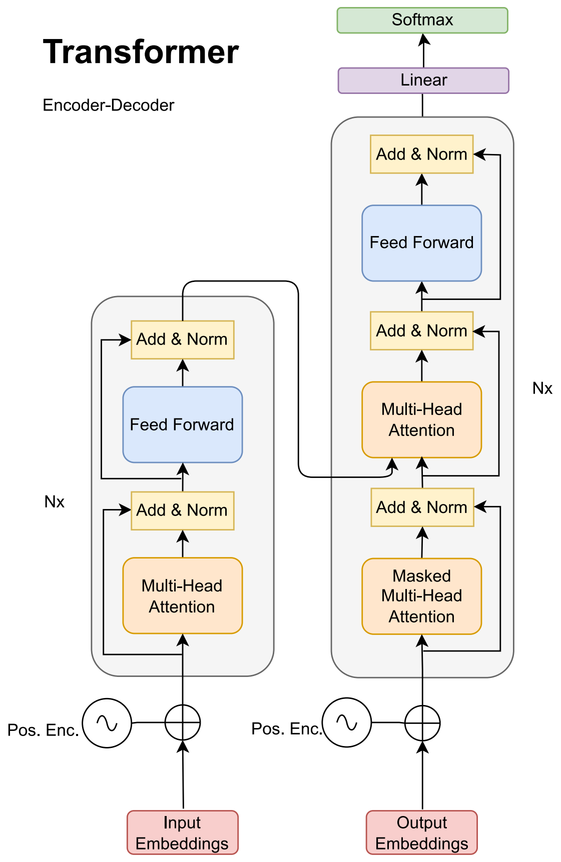


Figure 2.13: Visualization of a simplified Transformer model. N represents the number of encoder and decoder blocks stacked on top of each other. Illustration with modification from [7].

involves computing sinusoidal functions with different frequencies and phases, which are added to the input vectors to represent the position of each token within the sequence. This helps the model distinguish between tokens based on their position.

Encoder

Each encoder comprises a multi-head attention encoding layer and a feed-forward neural network (MLP). The feed-forward layer is a normal feed-forward neural network which usually uses ReLU or GELU as its activation function. At the beginning of the first encoder block, the outputs from the positional encoding layer are used as the queries Q, keys K, and values V for the multi-head attention layer. For the subsequent encoder blocks, the output from the previous encoder block is used as Q, K, and V. The multi-head attention layer encodes the input, which is then added to the original input through a residual connection surrounding the multi-head attention. The combined input is subsequently normalized using layer normalization, as depicted by the "Add & Norm" box. Not shown in the figure is that dropout is applied to a set percentage of the neurons after normalization is applied. The aggregated input is then passed through an MLP, and its output is also added and normalized. This constitutes a single encoder. Multiple instances of this encoder architecture are stacked N times, with each encoder taking the output of the preceding encoder as input.

Decoder

The decoder stack resembles the encoder stack but with some modifications. The first distinction is the masking of the initial multi-head attention layer, which processes the output embeddings generated by the model itself. Masking implies setting attention values to zero for words that have not been generated yet, allowing the decoder to attend only to words it has already produced. The second difference is that the subsequent multi-head attention layer processes both the aggregated and normalized output of the masked multi-head attention layer and the encoder stack's outputs using a process often referred to as Cross-Attention. As the inputs for this layer, Q comes from the output of the final encoder layer, while K and V comes from the Masked Multi-Head Attention layer in the decoder. This means that the entire input sequence is first encoded by the encoders and then utilized as a static input for all decoder blocks.

A linear layer at the end of the decoder stack maps the decoder's real-valued output to a vector of the desired size. For example, this output vector's size could be the number of classes if the Transformer is employed for classification, or the same size as the output vocabulary if used for machine translation or other sequences. The final output is then passed through the softmax function to generate a probability for each class or the probability of each token in the target vocabulary.

2.7.3 Learning objectives

This section focuses on the learning objectives for Transformer models, specifically the two main pre-training objectives: Masked Language Modeling (MLM) and Causal Language Modeling (CLM). The learning objectives provide a set of tasks for the model to perform, guiding its training process. They are instrumental in developing the model's language understanding and generation capabilities, which in turn result in improved performance for downstream tasks, such as abstractive summarization.

Masked Language Modelling

Masked Language Modeling (MLM) is a self-supervised learning task employed for pre-training transformer models. The objective of MLM is to predict randomly masked words in a sequence based on the context provided by the surrounding words. The model is trained to reconstruct the original sequence, constrained by not being able to rely on the masked words. This task is beneficial as it necessitates the model to learn a rich representation of the input text, capturing the meaning of the text, and enhancing performance on downstream tasks such as automatic summarization. The self-supervised nature of MLM allows the model to leverage large amounts of unlabelled text data, making it more scalable and effective at learning complex language patterns.

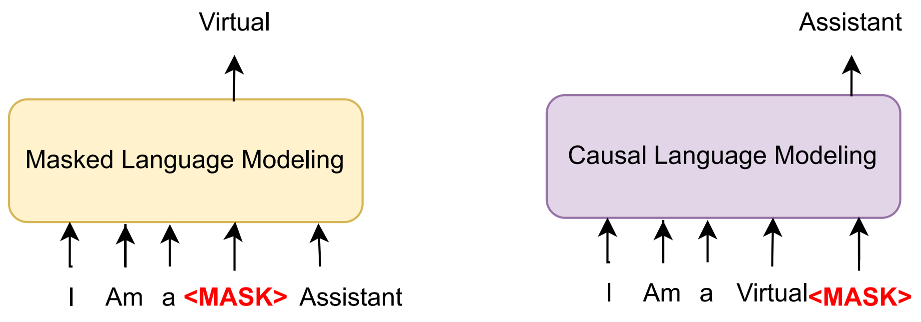


Figure 2.14: Examples of the MLM and CLM learning objectives

Figure 2.14 shows the MLM procedure used for training BERT [9]. The middle token gets masked with a "<MASK>" token. After the inputs get passed through the model, the model tries to predict which word the masked token should be replaced by. In the original paper, 15% of all tokens in the input were masked, but more recent research has shown that it might be beneficial to mask up to 40% percent of the tokens [91].

BERT (Bidirectional Encoder Representations from Transformers) is a prominent example of a transformer model that employs MLM as its pre-training objective [9]. BERT revolutionized natural language processing by introducing the concept of bidirectional context representation, which allows the model to understand the relationships between words in both forward and backward directions. By masking a percentage of the input tokens and requiring the model to predict the masked tokens based on the unmasked ones, BERT successfully captures a deeper understanding of the input text. This innovation led to significant improvements in various downstream tasks, setting new state-of-the-art benchmarks across multiple natural language processing domains.

Causal language modelling

Causal Language Modeling (CLM) is another widely used pre-training objective for Transformer models, and more specifically, autoregressive transformer models. As described in Section 2.6, autoregressive models generate text by predicting one word at a time, conditioning on the previously generated words. CLM follows a similar approach for pre-training, where the model learns to predict the next word in a sequence, given all the preceding words.

The objective of CLM is to maximize the likelihood of a target word, given the context words in the sequence. By training the model in such a manner, it learns the inherent structure and patterns within the language, enabling it to generate coherent and contextually relevant text. This is achieved by minimizing the cross-entropy loss between the predicted probabilities and the true word distribution.

Compared to MLM, CLM allows the model to learn only unidirectional context representation, as it predicts words based solely on the words that precede them. However, this unidirectional approach aligns well with the text generation process, where words are generated sequentially. Models like GPT-2 and GPT-3 (Generative pre-trained transformer) [2, 92] employ CLM as their only training objective, achieving state-of-the-art performance in various natural language processing tasks, including translation and summarization. While CLM has some limitations compared to MLM, such as the inability to leverage bidirectional context, it has proven to be highly effective for many downstream tasks, particularly those related to text generation.

2.7.4 T5: Text-to-Text Transfer Transformer

The T5 models were first introduced by Raffael et al. [3] in 2019, are a series of Transformer-based models designed to work as unified text-to-text models for multiple tasks. These models are pre-trained to convert any input text into a target text by framing a wide range of natural language processing tasks as sequence-to-sequence (seq2seq) problems. The seq2seq framework is well-suited for handling tasks such as translation, summarization, and question answering, as they all involve generating coherent output text based on the input text while preserving the underlying meaning and context.

While maintaining the overall encoder-decoder structure, T5 uses a simplified version of layer normalization, where only rescaling is applied, and no additive bias is introduced. In addition, layer normalization is applied outside the residual path. Another significant change in the T5 models is the use of relative position embeddings instead of the fixed sinusoidal position signals or learned position embeddings found in the original Transformer. This approach produces different learned embeddings based on the offset between the ”key” and ”query” in the self-attention mechanism. Moreover, the position embedding parameters are shared across all layers in the model, but within a given layer, each attention head uses a different learned position embedding. This modification allows the model to better capture positional relationships between words in a sequence. Despite these architectural alterations, the T5 models remain conceptually close to the original Transformer architecture, with the changes mainly aimed at enhancing performance and transfer learning capabilities.

The original T5 models were trained on a multi-task learning setup, which allowed them to perform multiple tasks concurrently by learning from diverse datasets and tasks. In this training approach, the model's parameters are shared across different tasks, allowing the model to generalize and transfer knowledge from one task to another. This multi-task training approach enabled the models to leverage shared representations and learn more efficiently, resulting in better performance across tasks.

The T5 models were initially pre-trained using a masked language modelling (MLM) objective, similar to BERT, which allowed them to learn meaningful representations of words and their contexts. After this pre-training stage, the models underwent further fine-tuning using a causal language modeling (CLM) objective. This step transformed the T5 models into autoregressive language models, which significantly improved their performance in text generation tasks by leveraging the knowledge and representations acquired during the MLM pre-training phase.

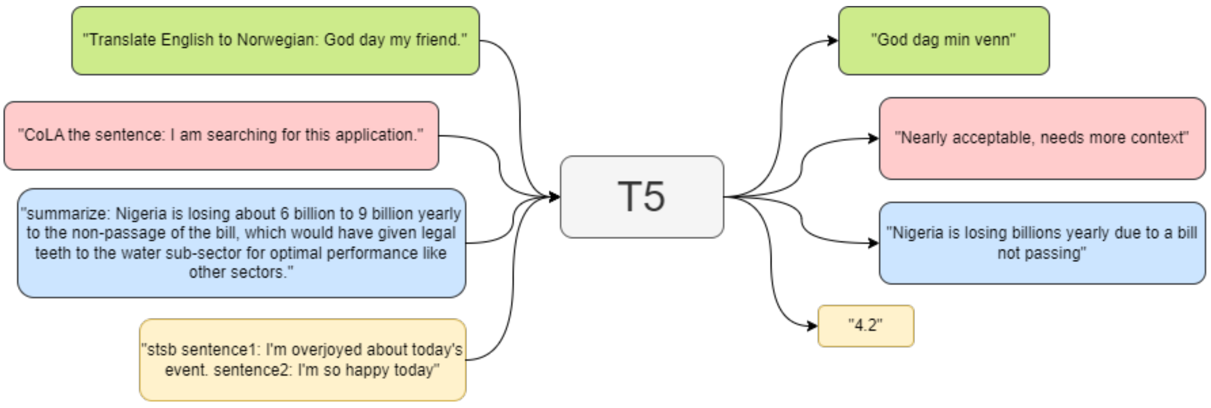
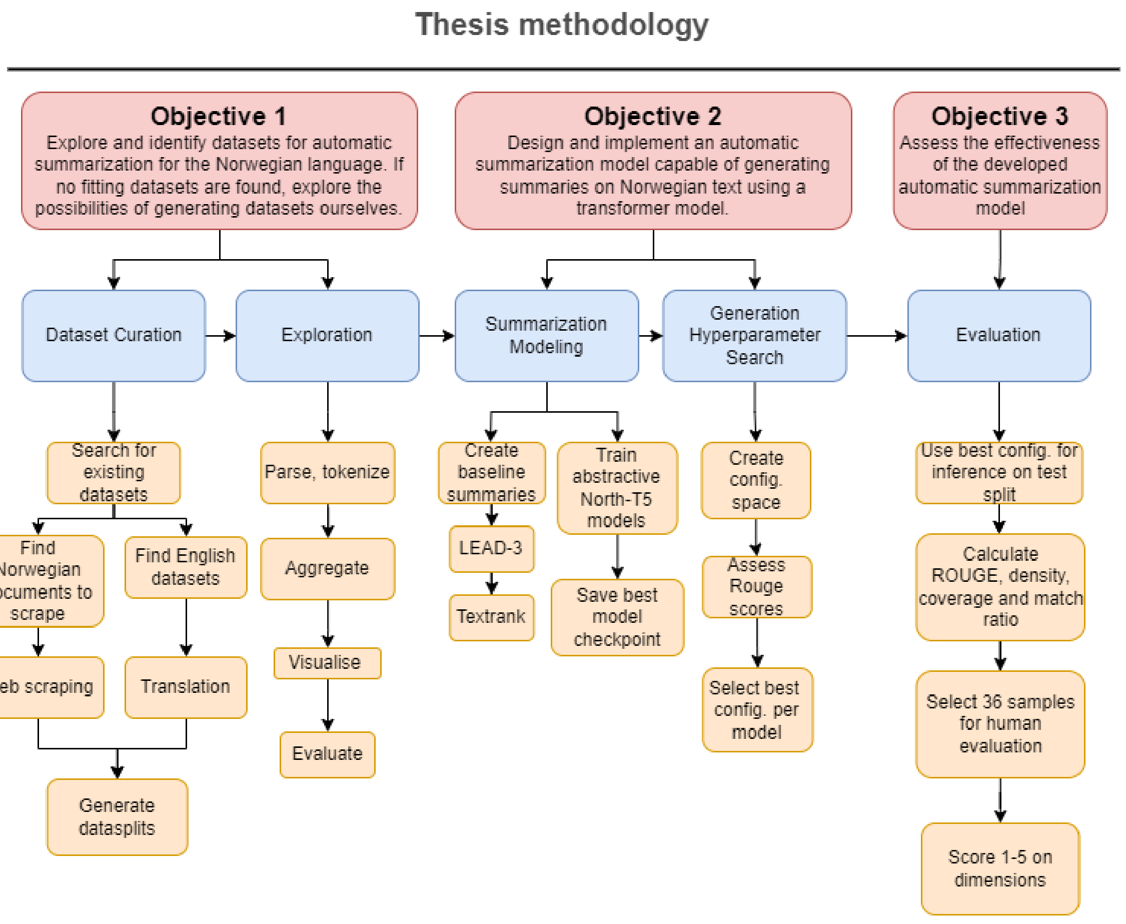

"Translate English to Norwegian: God day my friend."
"CoLA the sentence: I am searching for this application."
"summarize: Nigeria is losing about 6 billion to 9 billion yearly to the non-passage of the bill, which would have given legal teeth to the water sub-sector for optimal performance like other sectors."
"stsb sentence1: I'm overjoyed about today's event. sentence2: I'm so happy today"
T5
"God dag min venn"
"Nearly acceptable, needs more context"
"Nigeria is losing billions yearly due to a bill not passing"
"4.2"

Figure 2.15: T5 - Multi-Task training and fine-tuning visualized. CoLA measures whether a sentence is linguistically acceptable or not. Stsb scores two sentences based on how similar they are.

Figure 2.15 provides a visual representation of how T5 handles multiple tasks, such as translation, summarization, measuring linguistic acceptability with CoLA [93] or sentence similarity with stsb [94]. The model can process these diverse tasks concurrently by converting them into a unified text-to-text format. By incorporating a diverse set of tasks during pre-training, T5 models develop a more comprehensive understanding of language patterns and structures. This robust linguistic knowledge allows the models to effectively adapt to a variety of downstream tasks with minimal fine-tuning. Moreover, the multi-task learning setup encourages the models to learn task-agnostic representations, which are beneficial for transfer learning and enable the models to perform well on unseen tasks.

CHAPTER 3 METHOD

In this study, we follow a comprehensive approach that consists of five main stages: dataset curation, dataset exploration and visualization, modelling, generation hyperparameter tuning, and validation. These stages are supported by the theoretical underpinnings presented in Chapter 2.



Thesis methodology
Objective 1
Explore and identify datasets for automatic summarization for the Norwegian language. If no fitting datasets are found, explore the possibilities of generating datasets ourselves.
Objective 2
Design and implement an automatic summarization model capable of generating summaries on Norwegian text using a transformer model.
Objective 3
Assess the effectiveness of the developed automatic summarization model
Dataset Curation
Exploration
Summarization Modeling
Generation Hyperparameter Search
Evaluation
Search for existing datasets
Find Norwegian documents to scrape
Find English datasets
Web scraping
Translation
Generate datasplits
Parse, tokenize
Aggregate
Visualise
Evaluate
Create baseline summaries
LEAD-3
Textrank
Train abstractive North-T5 models
Save best model checkpoint
Create config. space
Assess Rouge scores
Select best config. per model
Use best config. for inference on test split
Calculate ROUGE, density, coverage and match ratio
Select 36 samples for human evaluation
Score 1-5 on dimensions

Figure 3.1: Flowchart of the methodology and steps utilized in this thesis.

This section begins by outlining the challenges encountered in identifying suitable Norwegian summarization datasets. We then discuss the two methods employed to address these challenges: web scraping and translation. We generate two datasets for further use, a web scrape of

Store Norske Leksikon (a Norwegian Encyclopedia) and a translated version of the CNN/Daily Mail dataset. Further, we discuss some extra datasets generated by us, which we do not focus on due to limited computational resources.

Subsequently, we describe the visualization techniques used for analyzing the curated datasets, elaborating on the rationale behind these visualizations and the software employed for their generation.

Following the data visualization, we first generate two different baselines using LEAD-3 and TextRank. Then, we train multiple T5 models of varying sizes on two of the datasets detailed in the first section. Additionally, we discuss the training strategies, hardware, and software utilized.

After training the models, we describe the optimization process for the sequence generation hyperparameters based on a set of manually created token generation configurations, as explained in Section 2.6, using the validation set.

Finally, we outline the systematic evaluation of our trained models on the test set from the datasets generated earlier. We evaluate using both ROUGE scores and human evaluation on a wide range of different criteria. Additionally, we perform human evaluation on a set of out-of-domain samples to better understand the model's behaviour.

3.1 Dataset Curation

During the initial stages of dataset curation, our primary objective was to locate existing summarization datasets in Norwegian. However, after conducting an extensive search, we found no publicly available datasets suitable for our purpose. We were able to find a master thesis [95] which was written in collaboration with Schibsted, an international media group headquartered in Oslo. The thesis created systems for extractive summarization using BERTSum [23] on a private dataset of news articles from Aftenposten and a shortened version (Aftenposten Oppsummert). We reached out to the supervisor from Schibsted, but we were not able to get access to the data.

Having exhausted the search for existing Norwegian summarization datasets, we turned our focus to well-established English summarization datasets, such as CNN/Daily Mail [96]. By closely examining these datasets, we identified that these datasets were often generated by scraping websites offering a more accessible and shorter version of the content. Being inspired by this, we realised that articles on Store Norske Leksion (SNL), a Norwegian Encyclopedia had a clear writing style where the lead paragraph (the first paragraph in an article) acted as an ingress/summary where the main parts of the article were highlighted. As a result, we generated a dataset based on articles and lead paragraphs from SNL. Additionally, we acknowledged the potential value of translating English summarization datasets into Norwegian to supplement our dataset acquisition. To this end, we translated the extensively researched CNN/Daily Mail dataset using a pre-trained transformer model specifically designed for English to Norwegian translation. In the following section, we provide more details on the methods and procedures employed to accomplish these tasks.

3.1.1 Web Scraping

Web scraping is an effective method for obtaining information and data from websites. By programmatically parsing and navigating web page structures, desired content can be accessed with ease. In the field of NLP, web scraping has become an essential tool, allowing researchers to gather vast quantities of text data from a wide range of online sources.

SNL

Store Norske Leksikon (SNL) is a peer-reviewed encyclopedia in Norwegian, owned and maintained by Norwegian universities and public organisations. Its goal is to make high-quality knowledge publicly available for Norwegians. The articles are structured so that the first paragraph (the lead) acts as a summary of the entire article. This made SNL an excellent source for creating a summarization dataset, with the lead paragraph serving as a summary and the remaining content as the full-length version.

We couldn't find any existing datasets containing SNL data, so we decided to create our own by scraping articles from SNL.no. The first step involved gathering a list of all article URLs on the site. We extracted the URLs from the sitemaps and retained only those following the format "https://snl.no/name_of_article" to avoid non-article pages.

Next, we scraped the URLs with multiple threads downloading articles at the same time using the Python module *grequests* and parsed the received HTML using *beautifulsoup4*. We extracted the text from the lead and the rest of the article text, joining the latter while removing any whitespace. Additionally, we saved metadata such as URLs, headlines, and categories for each article. To filter out very short articles, we set criteria for keeping an article: the lead had to be at least 100 characters long, and the rest of the article had to be longer than 400 characters.

Finally, we split the dataset using an 84%/6%/10% split for the train/validation/test sets. This division was chosen to ensure a sufficient amount of data for training our models while still providing an adequate sample size for validation and testing. By allocating a larger portion (84%) of the data for training, our goal was to optimize the model's learning process. We allocated 6% of the data for validation, which was intended to help fine-tune the model and its hyperparameters, while the remaining 10% was designated for the final evaluation of our model's performance on unseen data in the test set.

3.1.2 Machine Translation

In this subsection, we explore the process of translating English summarization datasets into Norwegian to expand our resources for training and evaluating our models. We discuss the challenges we faced in finding affordable and accurate translation solutions and present the machine translation model we selected for this task. Furthermore, we detail the translation process and the steps involved in adapting the CNN/Daily Mail dataset for our study.

In our quest for cost-effective and accessible translation solutions, we explored a variety of commercial services, including Google Translate, Azure Translator, and deepl.com. While these services excel at English-to-Norwegian translation, their pricing structures, usually determined by the number of characters or tokens generated, presented a considerable financial burden. Similarly, commercially hosted language models highly capable of translation, such as OpenAI's GPT-3.5, were also deemed too expensive due to the costs associated with their usage. During our development process, the more affordable ChatGPT API was released, offering translation capabilities at a fraction of GPT-3's cost. However, the timing of its release rendered it an unviable option for our project as we had already moved on to exploring different approaches. As a result, we decided to explore alternative avenues and sought a publicly available pre-trained model to perform machine translation ourselves, balancing quality and cost-efficiency in the context of our study.

Our search led us to the discovery of the OPUS-MT models from Helsinki NLP. These models provide a diverse range of over 1000 pre-trained translation models, including multiple options tailored for English to Norwegian translation. By utilizing the OPUS-MT models, we were able to capitalise on their capabilities, achieving a satisfactory balance between translation quality and cost efficiency.

OPUS-MT Model

The OPUS-MT models [97] leverage the extensive OPUS parallel corpus [97], which encompasses a wide variety of multilingual texts from diverse domains such as Wikipedia, movie subtitles, and EU legislation. These models are specifically tailored for low-resource languages, demonstrating competitive performance in numerous translation tasks.

Performance evaluation of these models takes place on the Tatoeba Challenge Datasets [98]. Tatoeba is a comprehensive, open, and collaborative multilingual database containing sentences and their translations in a vast array of languages, including low-resource and minority languages. This dataset consists of high-quality translations and diverse language pairs, contributed by a global community.

Built on a standard transformer setup, the architecture of OPUS-MT models features six self-attentive layers in both the encoder and decoder networks with eight attention heads per layer, where one of the heads is trained to learn alignment between words [97, 99]. This design enables the models to capture intricate relationships between source and target languages, yielding more accurate and coherent translations. For tokenisation, SentencePiece is employed with a vocabulary of 54 215.

The OPUS-MT model specifically tailored for English to Norwegian translation [100] achieved an impressive BLEU score of 56.4 on the Tatoeba English to Norwegian Bokmål test set. The model also works for Nynorsk, where it scored 40.3. Based on this performance, we selected this model for our dataset translation.

Most OPUS-MT models are readily available on the HuggingFace Hub, facilitating experimentation. However, our chosen model was not accessible there for unknown reasons. To address this, we downloaded the official weights from Helsinki NLP, converted the model to the appropriate format, and shared it on the Huggingface Hub [101].

CNN/Daily Mail

To create a dataset suitable for training a Norwegian text summarization model, we used the CNN Daily Mail dataset (referred to as CNN/DM in the remainer of the thesis), which consists of English news articles and their corresponding summaries. The dataset has been widely used for text summarization research, particularly for abstractive summarization tasks. The samples are extracted from CNN and Daily Mail articles and their associated bullet point descriptions which have been concatenated to create a summary for the whole article. These bullet points have been shown to be highly extractive [27, 102] with the most important information located in the first few sentences [103]. The summaries have also been shown to rarely contain the same trigram more than once [104], suggesting that token decoding techniques that avoid generating the same n-grams more than once might be effective when generating summaries using this dataset.

To translate the dataset from English to Norwegian, we employed the OPUS-MT model as described above with the following process.

Dataset Translation Process

To translate the dataset using the pre-trained model, we followed these steps:

1. First, we preprocessed the English dataset by removing any special characters, HTML tags, and extra spaces to ensure a clean input for the translation model.
2. We then divided the dataset into smaller sentences to facilitate parallel processing and efficient translation. This approach allowed us to speed up the translation process and reduce the memory requirements of the translation model.
3. For each chunk, we used the pre-trained OPUS-MT model to translate the English articles and summaries into Norwegian. We fed the model with the preprocessed text and generated the translated text as output.
4. After translating all the chunks, we combined the translated Norwegian text to form a complete translated dataset. We ensured that the translated articles and summaries were correctly aligned and maintained the same structure as the original dataset.
5. Finally, we post-processed the translated dataset by correcting any tokenization issues, ensuring proper punctuation, and restoring any special characters that were removed during the initial preprocessing step.

3.1.3 Additional Datasets Not Utilized Further

In the course of our research, we developed several additional datasets that were not ultimately incorporated into the thesis due to the decision to narrow our focus primarily due to limited computational resources and slow training times. Nevertheless, we believe these datasets may be of interest to others, and we have made them available on the Huggingface Hub for exploration and use. In this section, we provide a brief overview of these datasets, their sources, and the methodologies used in their creation.

Among the unused datasets are Norwegian translations of two well-known English summarization datasets, XSUM [105] and SAMSUM [106]. We employed the OPUS-MT model to translate both datasets. XSUM consists of single-sentence summaries of BBC news articles, while SAMSUM comprises abstractive summaries of dialogue-based conversations.

We also generated a dataset by extracting Norwegian Wikipedia articles and their associated lead paragraphs. The methodology used for this dataset closely resembles the approach taken in the creation of the Store Norske Leksikon (SNL) dataset.

Lastly, we created a dataset using articles from the Norwegian newspaper VG. The source of these articles was the Norsk Aviskorpus (Norwegian newspaper corpus) [107]. This corpus includes articles from Norway's largest newspaper from 1998 to 2019. In this dataset, we used the first paragraph (lead) of each article as its summary, similar to the approach taken with the SNL dataset.

3.2 Data Exploration

Our primary motivation for exploring the data was to understand the different distributions of the sizes of the documents and reference summaries as well as understand the abstractiveness of the reference summaries.

3.2.1 Token Distributions

In Section 3.3.3, we discuss the rationale behind truncating documents to the first 512 tokens for abstractive model summarization. To investigate the extent of document truncation, we utilized the subword SentencePiece tokenizer specifically designed for the North-T5 models employed in our study (covered in Section 3.3.2). We tokenized each document and calculated the number of tokens present in each article and reference summary. A histogram was created to visualize the distribution of token counts across the documents for each of the two datasets. A vertical line at 512 tokens was added to the plot, marking the truncation point. This graphical representation allowed us to better understand the proportion of documents affected by the truncation process and its potential impact on the model's summarization performance.

We aimed to compare the lengths of articles and summaries as well as the compression rate for the translated version of the CNN/DM dataset against the original English version. The multilingual SentencePiece tokenizer used by the North-T5 models has been exposed to more English than Norwegian during its training [108]. Consequently, it's expected to tokenize Norwegian sequences and words into smaller subunits, resulting in a higher token count for a Norwegian sentence compared to its English counterpart [109]. For this reason, a comparison based on the number of tokens might not yield meaningful results. Instead, we chose to compare the average number of characters in the datasets (including special characters), as this approach bypasses the potential misleading results introduced by the tokenizer.

3.2.2 Exploring Abstractiveness of Summaries

To better understand the abstractiveness of the reference summaries in our dataset, we employed various techniques and metrics, such as coverage, density and our own novel metric, match ratio. These metrics helped us measure the text overlap between the summaries and the source documents, as well as the rate at which the conveyed information is compressed.

We used the definitions from Grusky et al. [27] to calculate the extractive fragment coverage, extractive fragment density, and compression ratio. The coverage measures the proportion of words in the summary that are part of an extractive fragment in the article, while the density measures the average length of the extractive fragment to which each word in the summary belongs. As density measures overlaps between longer fragments, it should be used as the primary metric to measure extractiveness. The compression ratio is the word ratio between the article and its summary. To compute the metrics, we first tokenised the articles and summaries into words using NLTK [110] and calculated the coverage and density.

We also calculated Match Ratio (MR), which is our own novel contribution based on string distances calculated using Levenshtein distances [111]. The Levenshtein distance calculates the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one string into another. As a threshold for classifying two sentences as equal, we used a threshold of 0.95 on the normalized distances between sentences.

Visualizing Abstractiveness Metrics

To display the calculated metrics' distributions, we employed *Seaborn*, a Python library dedicated to data visualization. We generated kernel density estimate (KDE) plots for both density and coverage. KDE illustrates the data through a continuous probability density curve in one or multiple dimensions. Although comparable to a 2D histogram (heat map), KDE can create a less cluttered and more easily understandable plot, particularly when visualizing multiple distributions.

These visualizations helped us gain insights into the quality, originality, and effectiveness of the summarization datasets, and later in our summarization technique, used in our dataset and provided a better understanding of the degree to which the dataset and models are extractive or abstractive.

3.3 Summarization Modelling

In this section, we delve into the methodology adopted for the summarization process, highlighting the use of pre-trained Norwegian T5 models and their fine-tuning to achieve desired outcomes. The North-T5 models, varying in size and complexity, are built upon the foundation of the multilingual T5 model, trained to encompass 101 languages, including Norwegian. By leveraging these models, we bypass the computational challenges associated with training language models from scratch. To provide a comparative perspective, we also include baseline methods, previously discussed in the theory section, such as LEAD-3 and TextRank.

This section further details the training procedure, which employs key techniques such as early stopping, gradient accumulation, and half-precision training to optimize performance and efficiency. Additionally, we discuss the software and hardware choices that were made to facilitate seamless model training and experimentation.

3.3.1 Baselines

In order to evaluate the performance of our North-T5 models, we compare them against two commonly used baselines in the field of extractive summarization, LEAD-3 and TextRank. These baselines serve as a reference point to gauge the effectiveness of our models and assess whether they yield improvements in summary quality.

LEAD-3

As previously discussed, LEAD summarization methods select a certain number of the initial sentences of the document as the summary. In our evaluation, we use the LEAD-3 approach, which selects the first three sentences of the input text as the summary. By including LEAD-3 as a baseline, we aim to measure the extent to which our models can surpass this rudimentary yet efficient approach.

TextRank

TextRank, an unsupervised graph-based extractive summarization method, constructs a graph with sentences as nodes and edges between nodes based on the similarity between sentences. It iteratively assigns scores to each sentence node, and upon convergence, the top-ranked sentences are selected and combined in their original order from the source text, resulting in a summary. We used the Sumy library for the TextRank implementation [112]. By incorporating TextRank as a baseline, we aim to determine if our models can outperform this more sophisticated method in terms of coherence and structure.

3.3.2 North-T5 models

The decision was made at an early stage that training a Norwegian language model (LM) from the ground up would be beyond the scope of this thesis due to the computational expense involved in training such models from the ground up. Fortunately, a series of Norwegian T5 models, referred to as North-T5 in this thesis, have been trained and made publicly available by Per Egil Kummervold [113]. These models have been privately trained by Kummervold and are currently in use at the National Library of Norway.

Instead of training the models from scratch, the North-T5 models used in this thesis were built upon a variant of the original T5 model named mT5 [108]. The mT5 model is a multilingual adaptation of the T5 architecture, which has been trained using a similar methodology as described in the original T5 paper. The distinction lies in the expansion of the training data to encompass 101 languages, including Norwegian.

While no thorough evaluation of the models has been performed by either us or Kummervold, they have both been fine-tuned on classifying which political party Norwegian political transcripts belong to [114]. Kummervold reports an F1 score of 73.2 for the mT5-base compared to 85.3 for the North-T5-base [113]. This gave us the impression that the North-T5 models are more capable of modelling the Norwegian language than mT5 and that they should act as a solid basis for our summarization task.

For tokenization, the models employ the sub-word SentencePiece tokenizer described in Section 2.1.1 which is commonly used for non-English and multilingual models. A vocabulary size of 250 112 tokens is used for all of the models.

The North-T5 models were fine-tuned on the Norwegian Colossal Corpus (NCC), a compilation of multiple Norwegian corpora released by the AI-lab of the National Library of Norway [115]. The majority of the training data (49GB in total) consists of scanned newspapers, transcripts from the Norwegian parliament, and books. In addition to the NCC, supplementary data was included from Common Crawl and English Wikipedia to enhance the multilingual capabilities of the models [113]. The different steps from mT5 to our fine-tuned summarization models are shown in Figure 3.2.

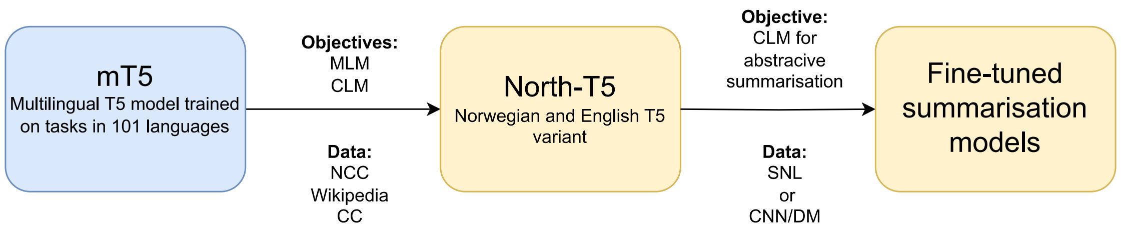

mT5
Multilingual T5 model trained on tasks in 101 languages
Objectives: MLM CLM
Data: NCC Wikipedia CC
North-T5
Norwegian and English T5 variant
Objective: CLM for abstracive summarisation
Data: SNL or CNN/DM
Fine-tuned summarisation models

Figure 3.2: Visual explanation of the different training stages and data used to generate create our final models based on mT5 and North-T5.

Five unique models of different sizes were introduced (small, base, large, XL, XXL). The tiniest model, t5_small_NCC_lm, comprises 300M parameters, whereas the most substantial, t5_xxl_NCC_lm, consists of 13B parameters. The architectures for all these models are alike but vary in the dimensions of numerous components within the transformer structure. For instance, the base version is composed of 12 encoder and decoder blocks with 12 attention heads in multi-head attention layers, while the large version features 24 encoder and decoder blocks with 16 attention heads.

This thesis will focus on the base and large variants (referred to as T5-Base and T5-Large) of the North-T5 models, consisting of 580M and 1.2B parameters, respectively. The decision to train only the base and large North-T5 models was influenced by the limited computational resources available. Training larger models such as XL or XXL would have required access to more powerful hardware and a higher budget, which was not feasible in this case. Thus, we focused on base and large models, which still allowed us to explore the potential improvements in summarization performance while keeping the computational requirements manageable. Training two different models with different parameter counts should also give us an idea of how the performance would scale if the larger XL and XXL were to be trained in the future.

3.3.3 T5 Training procedure

The training procedure for fine-tuning the North-T5 models for abstractive summarization incorporated several crucial considerations. Due to limited computational resources, hyperparameters were not explicitly fine-tuned for optimal performance. Nonetheless, the selected configurations aimed to facilitate a reliable and robust training process.

In a similar fashion to the original T5 paper [3], we inserted the prefix ”oppsummer: ” (”summarize: ” in English) at the beginning of the text to be summarized. We later realized that this was most likely unnecessary and is only needed when training a model to perform multiple tasks simultaneously.

Model inputs were processed by limiting the length of the input text to the first 512 tokens, ensuring a consistent input size for training. This truncation allows the model to efficiently process the data while maintaining manageable computational requirements. However, this method has the drawback of discarding data, since any tokens beyond the limit are removed from the input.

To mitigate overfitting to the training set, we employed a dropout strategy, in which 10% of the neurons were dropped out following the layer normalization layers. This approach is consistent with the method used in the training of mT5 models, as described by Xue et al. [108]. Applying dropout aids in reducing overfitting by adding a form of regularization, which encourages the model to learn more robust representations and prevents it from relying too heavily on any single neuron.

An adaptive learning rate was utilized, with the optimizer employing the AdamW algorithm [70] at a learning rate of 5e-5. A linear scheduler was implemented to adjust the learning rate during training. The application of a linear scheduler allows for a gradual and controlled decrease in the learning rate throughout the training process. This can aid in mitigating the risk of being trapped in local minima and improving the chances of converging towards a more optimal solution.

Early stopping was employed as a regularisation technique to prevent overfitting and ensure convergence during the training process. Early stopping involves tracking a performance metric throughout the training and ceasing the training process once this metric ceases to improve or begins to deteriorate. We tracked performance on the validation loss, calculated using cross-entropy on the validation set after each epoch. Early stopping would be activated if the validation loss did not decrease for two epochs meaning that the best model would have the weights from two epochs before.

Gradient accumulation was utilized over four steps in the training process, which enabled the aggregation of gradient values from four mini-batches before performing a single weight update. This approach facilitates training with larger effective batch sizes without increasing memory requirements, as gradients are computed and stored for multiple mini-batches before being averaged and applied in a single update step. This was done to help in mitigating the effects of noisy gradient estimates, which might lead to more stable and efficient optimization during training,

We implemented half-precision training by employing 16-bit floating-point numbers in place of the usual 32-bit values [116], enabling us to speed up the training process and decrease memory consumption with minimal impact on accuracy.

3.3.4 Software and Hardware

Software

This section covers the main software and Python modules used for training the T5 models. A full list of Python modules used is found in Table A.1.

In this study, we employed various libraries developed by Huggingface for training the models. Huggingface is an American company that creates open-source software for working with transformer models. At the core of the Huggingface ecosystem is the 'transformers' Python module [117], which provides access to thousands of pre-trained models and equips users with essential tools for training new models. This module supports two distinct backends, PyTorch [118] and TensorFlow [119], affording flexibility in the selection of underlying frameworks. Drawing from our previous positive experiences with PyTorch, particularly its easy and dependable configurations on Nvidia GPUs, we opted to utilize this backend for this thesis.

PyTorch is a widely-adopted open-source machine learning library primarily developed by Meta (formerly Facebook)'s AI Research lab (FAIR) [118]. PyTorch offers a comprehensive ecosystem for developing deep learning models, advanced tensor computations with GPU acceleration, and an extensive toolkit for constructing neural networks. A distinguishing feature of PyTorch is its dynamic computational graph, also known as "eager execution," which facilitates a more intuitive and flexible development process. Moreover, it enables seamless integration with numerous other libraries and frameworks within the machine learning ecosystem.

Closely associated with the 'transformers' module is the 'datasets' Python module from Huggingface, which includes efficient loaders for loading and creating datasets. Both pre-trained models and datasets can be uploaded to the HuggingFace Hub, enabling easy sharing. The North-T5 models described earlier are accessible on the Hub as of April 5, 2023, and our curated datasets have also been shared.

In addition to the aforementioned tools, we employed DeepSpeed [120] for enhancing the training efficiency of our models. DeepSpeed is an open-source library developed by Microsoft that aims to optimize the performance of large-scale model training. It is integrated into 'transformers' allowing us to take advantage of its optimizations while preserving the original workflow of the HuggingFace ecosystem. By incorporating advanced techniques such as FlashAttention, gradient accumulation, mixed-precision training, and distributed training strategies, DeepSpeed significantly reduces the memory footprint and accelerates training times for deep learning models. Moreover, it supports flexible and customizable training configurations, making it compatible with a wide range of models and applications.

To effectively monitor the training performance of our models and streamline experiment management, we utilized Weights & Biases (WandB) [121]. WandB is a versatile, open-source platform designed to facilitate easy tracking of model training progress, performance metrics, and hyperparameters. With seamless integration into the HuggingFace ecosystem and PyTorch, WandB efficiently records metrics for each training run and provides comprehensive visualizations.

Hardware

After deciding that we wanted to focus on the North-T5 base and large models with 580M and 1.2B parameters, we understood that efficient experimentation necessitated GPU acceleration. Specifically, to accommodate the large-scale nature of these models, we required multiple GPUs equipped with substantial amounts of VRAM, which would enable us to train the models using higher batch sizes. Utilizing GPUs for training would significantly expedite the process, allowing for quicker model convergence and reducing the time spent on iterative experimentation.

To handle the challenges described above, we decided on renting GPU instances from vast.ai. Vast.ai is a cloud computing platform that specializes in providing on-demand access to GPU resources for machine learning and deep learning applications. This platform allows users to rent GPU resources from a diverse marketplace of contributors, enabling cost-effective and scalable access to high-performance computing resources. Due to the varying availability of the instances on vast.ai, the models described in this section were trained on different instances with different GPUs. A full overview of the different hardware setups used for training each T5 model is provided in Table 4.4

3.4 Optimizing Decoding Parameters

As highlighted in Section 2.6, the selection of tokens in the decoding process plays a crucial role in determining the quality of the resulting summaries. While the straightforward greedy approach may yield acceptable results, it often generates repetitive and less inventive sequences.

To determine the optimal token decoding parameters for producing high-quality summaries, we conducted a search for the most suitable configurations. Although the evaluation of summary quality is subjective, we relied on a range of computed metrics for each configuration to guide our selection. We had to perform this parameter search on all four models trained and identify the best parameters for each model as it was no guarantee that the same generation parameters would be the same for all four.

We manually devised 12 different parameter configurations, encompassing the various settings covered in Section 2.6. These configurations can be broadly categorized into two main strategies: generation using beam search and generation through sampling from softmax probabilities. Furthermore, we included the naive, greedy generation technique as a benchmark for comparison. A comprehensive overview of these configurations can be found in Table D.5.

We assessed the performance based on four ROUGE metrics, namely ROUGE-1, ROUGE-2, ROUGE-L, and ROUGE-LSum. ROUGE-1 was the primary metric we used, due to it being the most general of the 4 ROUGE metrics measured as it primarily measures the overlap between single words and not n-grams. If different parameters had similar ROUGE-1 scores, we looked at the ROUGE-2 as a tiebreaker.

3.5 Evaluation

The final evaluation of the models, employing the optimized generation parameters, was carried out using a two-fold approach: automatic evaluation with ROUGE metrics and human evaluation. This combination enabled us to benefit from the advantages of reference-based metrics, such as ROUGE, while also including human assessments to verify the metrics and gain a more comprehensive understanding of the models' behaviour.

3.5.1 Automatic Evaluation

The primary metric we wanted to focus on when evaluating was the different ROUGE metrics. To calculate them, we generated predictions for each document in the test set. For the T5-models, the optimal generation parameters found using the methods in the previous section were employed for each model. We then compared the generated summaries to the reference summaries to compute the corresponding ROUGE scores. Simultaneously, we recorded the average length of the generated summaries as well as the average time required to generate a sample.

3.5.2 Human evaluation

We conducted a manual evaluation on a subset of the generated summaries from each model, driven by two primary motivations. Firstly, we aimed to examine the correlation between our assigned scores and the ROUGE metrics for the summaries. While ROUGE metrics generally correlate with human evaluations, they can be fragile, as they may assign low scores to summaries with the same semantic meaning but different wordings or phrasings. Secondly, we sought to gain a deeper understanding of the model's behaviour, identifying its strengths and weaknesses.

Due to the time-consuming nature of human evaluation, we decided to assess a subset of the data. For each of the two datasets, we selected samples from the test set based on the performance of the large variant models (Large-SNL and Large-CNN/DM): the top 5 best ROUGE-1 scores, the 5 worst ROUGE-1 scores, and 5 randomly chosen documents. This approach ensured that all models would be assessed on a total of 15 documents from either the SNL or CNN/DM datasets, providing a representative sample for evaluation purposes.

Furthermore, to measure performance on out-of-domain data where the models were expected to perform worse, we extracted articles from various sources on different websites. In total, we downloaded 6 articles: 2 from Aftenposten (a Norwegian newspaper), 2 from Norwegian Wikipedia, 1 from Kvinneguiden (a Norwegian forum), and 1 from Komplett.no (a Norwegian online retailer).

We adopted the methodology proposed by Grusky et al. [27] in their Newsroom paper, as discussed in Section 2.2.4. Our evaluation framework encompasses four key dimensions: Informativeness, Relevance, Fluency, and Coherence, as outlined in Table 3.1. Additionally, we introduced a fifth criterion, Factuality, to address the issue of hallucination frequently encountered in large language models, which tend to generate plausible yet factually inaccurate text [122].

We conducted the evaluation ourselves, rating each dimension on a scale from 1 to 5, where 1 represents the lowest score and 5 indicates the highest. The scores were then aggregated for every model and data subset to provide a comprehensive assessment of their performance.

Dimension	Prompt
Informativeness	How well does the summary capture the key points of the article?
Relevance	Are the details provided by the summary consistent with details in the article?
Fluency	Are the individual sentences of the summary well-written and grammatical?
Coherence	Do phrases and sentences of the summary fit together and make sense collectively?
Factuality	To what degree the summary is truthful, is the summary free from hallucinations?

Table 3.1: Human evaluation criteria for summaries used by Grusky Et. Al [27] as well as our own dimension evaluating factuality. The dimension represents the different categories we measured the performance on and the prompt is the instruction for evaluating each dimension

3.5.3 Model Abstractiveness

Similarly to our approach for assessing the abstractiveness of reference summaries in the datasets, we computed coverage, density, and match ratio for the generated summaries as well. The rationale behind this analysis is to gain a deeper understanding of the abstractive summarization models' functioning. As previously noted, these models might rely solely on extracting sentences and phrases rather than generating novel ones. Given that the models are trained on datasets with specific degrees of abstractiveness in their reference summaries, it is reasonable to anticipate that the summaries produced by the models would exhibit similar characteristics.

CHAPTER 4 RESULTS

This chapter will review the results from the different methodologies used, but a more detailed analysis will be covered in Chapter 5. The results of the data curation process and the results of the data exploration process will be shown in Section 4.1. The results of summarization modelling and hyperparameters for token decoding will be covered in Sections 4.2 and 4.3. The final evaluation using ROUGE and human evaluation will be shown in Section 4.4

4.1 Data exploration

This section provides an overview of the results of the dataset curation process as well as the characteristics of the datasets.

Table 4.1 provides an overview of the number of samples in the different subsets for SNL and CNN/DM.

The initial exploration of SNL.no's sitemaps resulted in a collection of 168,685 URLs within the domain. After filtration, the final dataset, spanning all splits, comprised a total of 12,993 samples. After performing the splitting of the samples, this yielded 10 874 train samples, 819 validation samples and 1300 test samples.

56 358 samples from the CNN/DM dataset got translated into Norwegian using the pre-trained OPUS-MT model. After performing the splitting of the samples, this yielded 47 171 train samples, 3551 validation samples and 5636 test samples.

Dataset	# Samples			
	Train	Validation	Test	Total
SNL	10 874	819	1300	12 993
CNN/DM	47 171	3551	5636	56 358

Table 4.1: Number of samples in the different splits of SNL and CNN/DM.

4.1.1 Article and Summary Lengths

Table 4.2 shows the average number of characters for the two datasets as well as the English version of CNN/DM and the compression rate (ratio between article and summary lengths) for each dataset.

	Avg. Num. of Chars.		
Dataset	**Article**	**Summary**	Compression
SNL	1968	239	8.23
CNN/DM (English)	4034	295	13.67
CNN/DM (Norwegian)	3657	281	13.01

Table 4.2: Average number of characters in articles and reference summaries and average compression ratio

The distributions of the number of tokens generated with the North-T5 subword tokenizer are shown in Figure 4.1 for SNL and Figure 4.2 for CNN/DM.

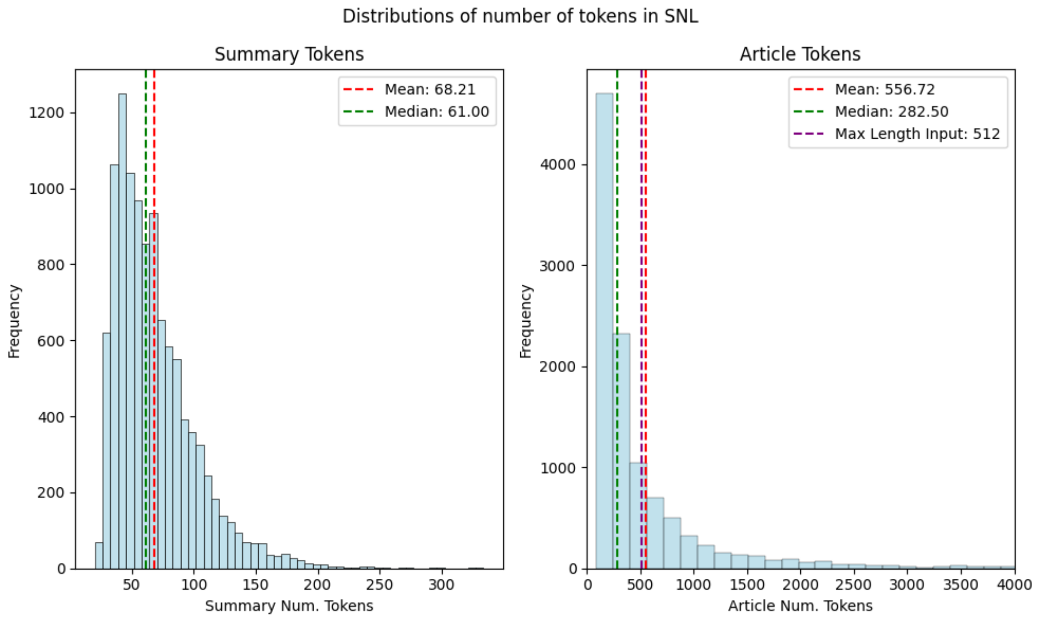


Figure 4.1: Token distributions generated with the North-T5 tokenizer for SNL

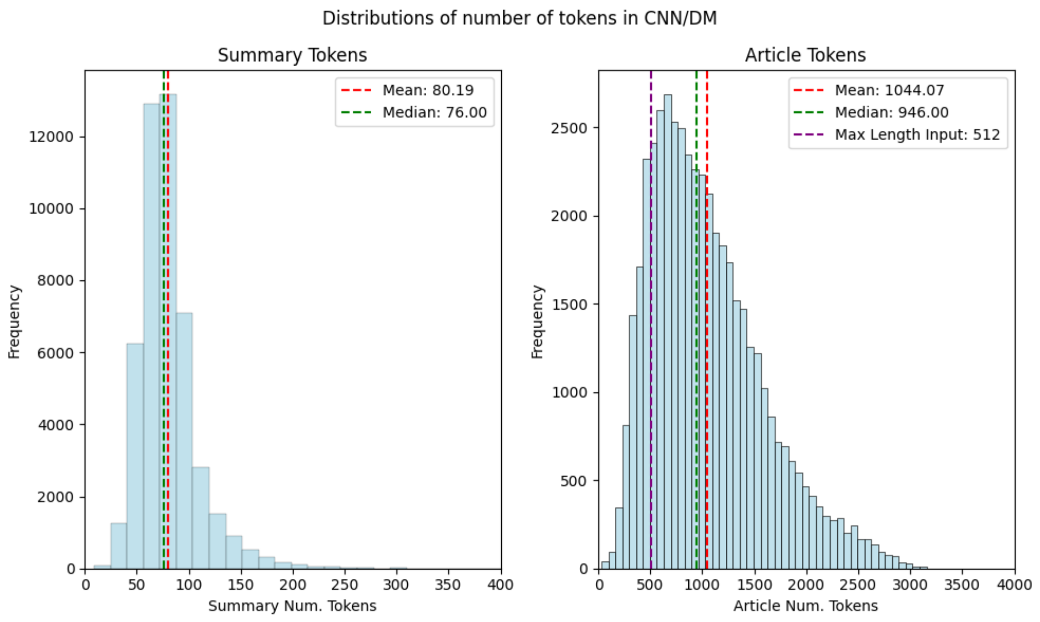


Figure 4.2: Token distributions generated with the North-T5 tokenizer for CNN/DM

4.1.2 Abstractiveness

The results on measuring the abstractiveness of the generated datasets are shown in Table 4.3 and the distributions of coverage and density in Figure 4.3. Higher values indicate a lower level of novel words and sequences in the reference summaries.

Dataset	**Match Ratio**	**Coverage**	**Density**
SNL	0.00	0.57	0.91
CNN/DM	0.01	0.80	2.45

Table 4.3: Comparison of models and datasets using match ratio based on Levenshtein distance, coverage, and density metrics.

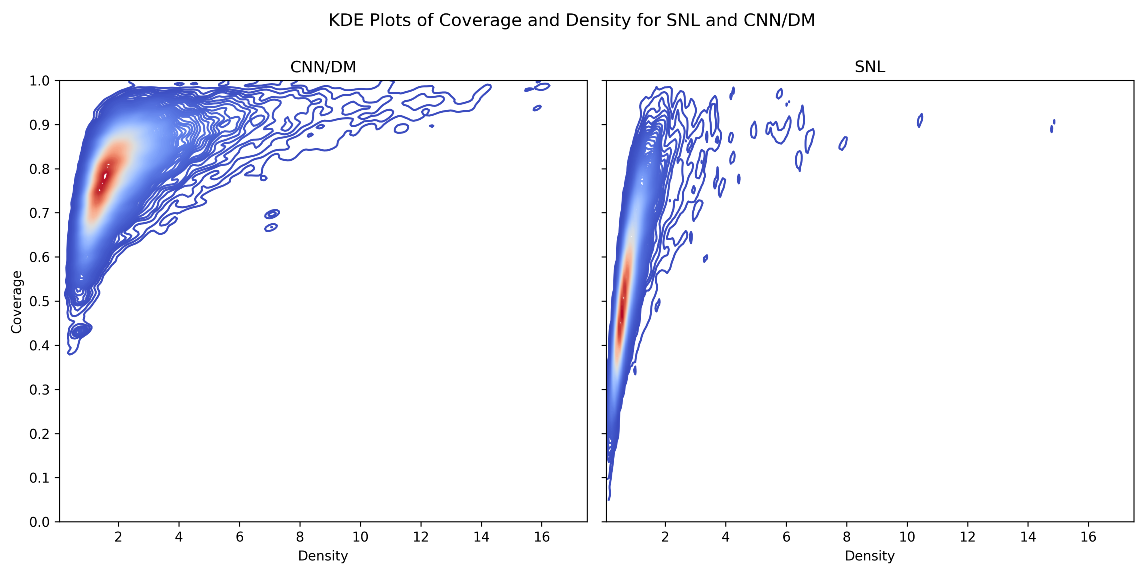


Figure 4.3: KDE plots for CNN/DM Nor and SNL with density on the x-axis and coverage on the y-axis. Red indicates a greater number of samples in that area.

4.2 Training Results

The results of fine-tuning the different North-T5 models on SNL and CNN/DM are covered in this section. Table 4.4 shows the number of epochs and time elapsed until early stopping was activated for each model. The evolution of loss values throughout training for each model is shown in Figure 4.4. Similarly, the evolution of ROUGE scores through training is shown in Figure C.1 in the Appendix.

Model	GPUs	Batch Size (per device)	Total Batch Size	Epochs	Time elapsed
T5-Base-SNL	4 x GTX 3090	4	64	19	2h 33m
T5-Large-SNL	4 x A40	4	64	13	2h 30m
T5-Base-CNN/DM	4 x A40	16	256	15	4h 0m
T5-Large-CNN/DM	4 x A40	4	64	16	16h 34m

Table 4.4: Fine-tuning times and GPUs utilized for different models. The total batch size is calculated with a gradient accumulation of 4 (a technique where the gradient updates are calculated over multiple mini-batches and applied once, which effectively simulates larger batch sizes).

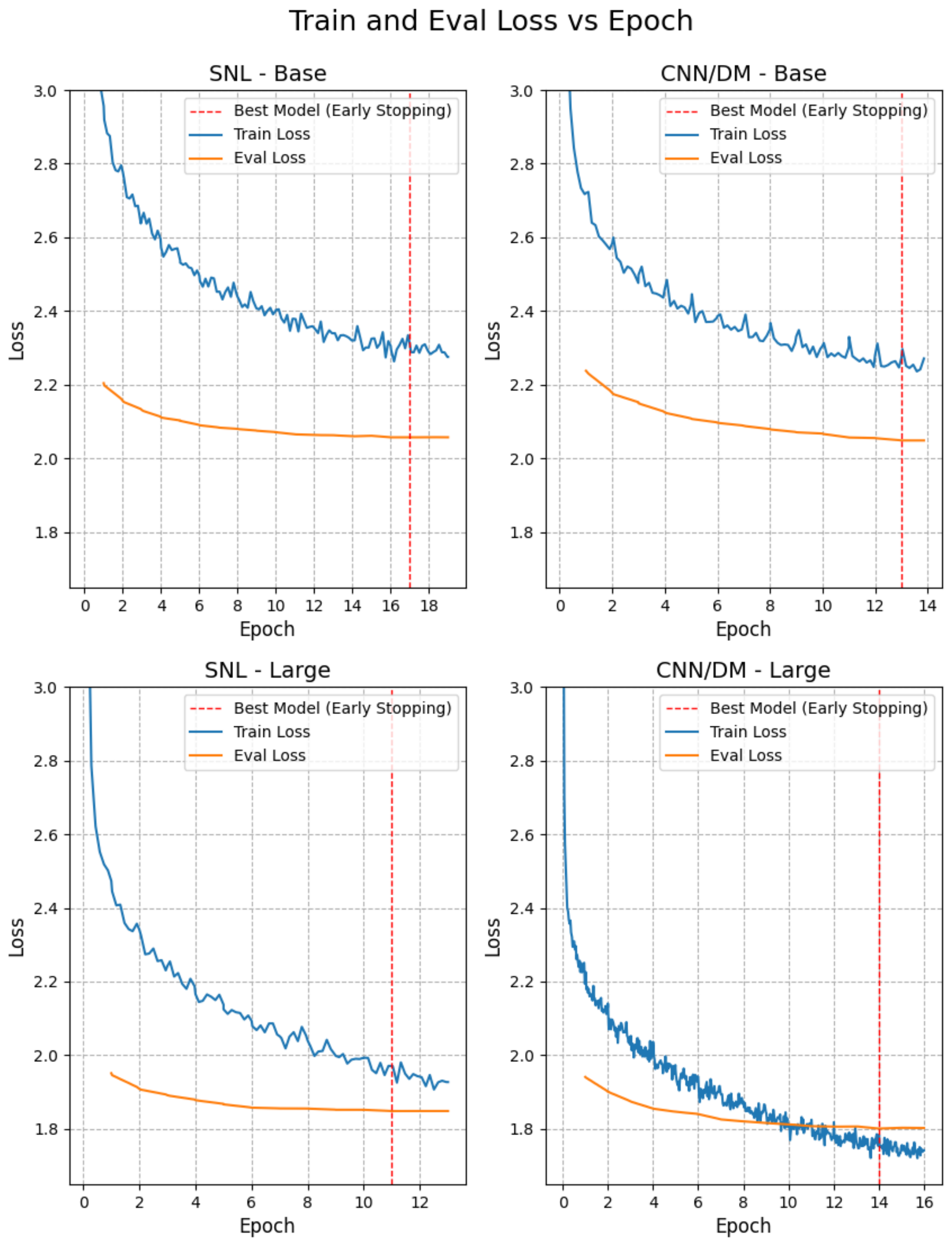


Figure 4.4: Training and validation losses for T5 models during training. The blue lines represent the training loss, while the orange lines represent the validation loss. The vertical dashed red line indicates the epoch where the model achieved the minimum validation loss, which is used for early stopping. For Base-CNN/DM, the loss values were not logged in wandb for the final epoch for unknown reasons.

4.3 Token Decoding Parameters

The best-performing token decoding parameters for each model are shown in Table 4.5 along with the different ROUGE scores on the validation sets. Performance for all of the configurations are shown in Section D.1

Model	Optimal Config	R-1	R-2	R-L	R-Lsum
T5-Base-SNL	Beam No-Repeat N-Gram Size: 3	34.01	15.72	28.84	31.6
T5-Large-SNL	Beam No-Repeat N-Gram Size: 3	36.02	17.39	30.64	33.41
T5-Base-CNN/DM	Beam No-Repeat N-Gram Size: 3	32.28	12.22	22.01	29.84
T5-Large-CNN/DM	Beam No-Repeat N-Gram Size: 5	33.27	13.05	23.03	30.86

Table 4.5: Best performing token decoding parameters for each model

4.4 Evaluation

In this section, we will present our findings during the evaluation of the performance, and abstractiveness, based on metrics, heuristics as well as human evaluation.

4.4.1 ROUGE metrics

Table 4.6 shows the ROUGE scores for each of the models on SNL and CNN/DM as well as the average length of generated summaries and runtime for base and large T5 variants.

Model	Dataset	R-1	R-2	R-L	R-Lsum	Avg. Gen. Length	Runtime per Sample (s)
LEAD-3	SNL	21.25	3.17	13.60	13.61	69.7	-
TextRank-3	SNL	20.16	3.14	12.36	12.35	107.9	-
T5-Base-SNL	SNL	33.91	15.42	28.71	31.33	44.95	0.277
T5-Large-SNL	SNL	**35.07**	**16.74**	**29.86**	**32.50**	41.73	0.486
LEAD-3	CNN/DM	29.82	10.84	18.84	18.83	72.61	-
TextRank-3	CNN/DM	23.79	6.46	14.47	14.48	130.56	-
T5-Base-CNN/DM	CNN/DM	32.53	12.23	22.13	30.05	94.47	0.449
T5-Large-CNN/DM	CNN/DM	**34.02**	**13.55**	**23.67**	**31.52**	95.95	0.765

Table 4.6: ROUGE metrics calculated on the train sets of SNL and CNN/DM. Bald numbers indicate the best-performing model for a metric

4.4.2 Human Evaluation

Tables 4.7 and 4.8 depict the results of the human evaluation performed. The results are divided into the three different subsets (Bottom R-1, Top R-1 and Random) and the two different T5 models trained. The R-1 scores for each subset are also included.

The evaluation results for out-of-domain samples mentioned in Section 3.5.2 can be found in Table G.1 in the Appendix.

SNL Human Evaluation Results

Subset	Model	Semantic		Syntactic			Avg.	R-1
		INF	REL	FLU	COH	FAC		
Bottom R-1	T5-Base-SNL	3.00	4.10	5.00	3.50	4.60	4.04	14.50
(5 samples)	T5-Large-SNL	2.90	4.10	5.00	4.17	4.10	4.05	8.53
Top R-1	T5-Base-SNL	1.40	1.20	5.00	4.40	1.20	2.64	94.44
(5 samples)	T5-Large-SNL	1.40	1.40	5.00	5.00	2.30	3.02	97.94
Random	T5-Base-SNL	3.00	3.40	5.00	5.00	3.00	3.88	36.10
(5 samples)	T5-Large-SNL	3.30	3.70	4.60	5.00	3.20	3.96	34.61

Table 4.7: Results of human evaluation across 5 dimensions on SNL. Results are divided into three different subsets based on the five highest and lowest R-1 scores for T5-Large and a subset of 5 random samples

CNN/DM Human Evaluation Results

Subset	Model	Semantic		Syntactic			Avg.	R-1
		INF	REL	FLU	COH	FAC		
Bottom R-1	T5-Base-CNN/DM	3.60	4.60	4.60	4.10	5.00	4.38	8.16
(5 samples)	T5-Large-CNN/DM	3.40	4.60	5.00	4.10	5.00	4.42	3.34
Top R-1	T5-Base-CNN/DM	1.00	1.00	5.00	5.00	1.00	2.60	61.94
(5 samples)	T5-Large-CNN/DM	1.00	1.00	4.60	4.60	1.00	2.44	99.65
Random	T5-Base-CNN/DM	3.10	4.20	3.80	4.20	4.60	3.98	30.58
(5 samples)	T5-Large-CNN/DM	4.20	5.00	4.40	4.60	5.00	4.64	43.47

Table 4.8: Results of human evaluation across 5 dimensions on CNN/DM. Results are divided into three different subsets based on the five highest and lowest R-1 scores for T5-Large and a subset of 5 random samples

4.4.3 Model abstractiveness

Similarly to in the dataset exploration Section (4.1.2), we calculated the Match Ratio, Coverage and Density to measure the level of abstractiveness on the generated summaries from the different models. The values calculated are shown in Table 4.9 and a KDE plot of the distributions of coverage and density in Figure 4.5.

Model	Dataset	Match Ratio	Coverage	Density
T5-Base-SNL	SNL	0.01	0.77	3.39
T5-Large-SNL	SNL	0.01	0.73	2.44
T5-Base-CNN/DM	CNN/DM	0.24	0.98	7.13
T5-Large-CNN/DM	CNN/DM	0.26	0.98	9.79

Table 4.9: Comparison of models and datasets using match ratio based on Levenshtein distance, coverage, and density metrics

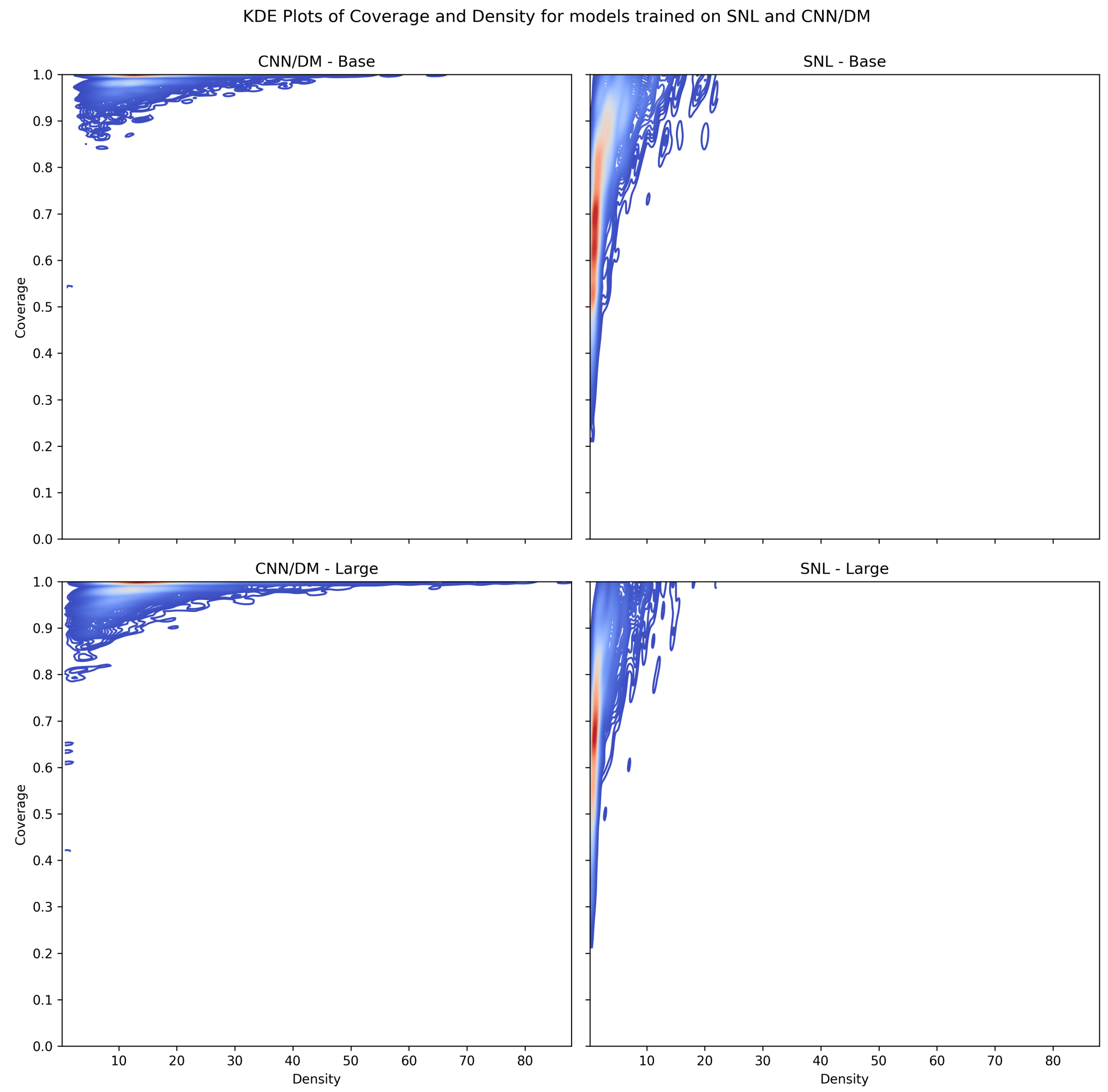


Figure 4.5: KDE plots for models trained on CNN/DM and SNL with density on the x-axis and coverage on the y-axis. Red indicates a greater number of samples in that area. Values were generated on the samples in the test sets.

CHAPTER 5 DISCUSSION

This section offers an in-depth analysis of the results derived from the previous sections. Our discussion is focused on understanding the insights and implications of these findings. Key areas of focus include the nature of the generated datasets, the training and optimization process for the T5 models intended for abstractive summarization, and how performance was evaluated using both ROUGE metrics and human evaluation. We also touch upon potential directions for future work.

5.1 Datasets

The first objective formed was based on exploring and identifying datasets which could be used to train a machine-learning model for automatic summarization. Since we found no suitable existing datasets and could not get access to Aftenposten Oppsummert from Schibsted, we ended up generating two datasets to fulfil the objective. The first dataset was generated by scraping the Store Norske Leksikon (SNL) website, and the second one was a machine-translated version of the CNN/Daily Mail dataset. This section discusses the quality and characteristics of the two datasets.

5.1.1 SNL

The SNL dataset was generated by scraping Store Norske Leksikon, using the first paragraphs as reference summaries and the remaining article content as full documents. This process resulted in a total of 12,993 samples, which were split into 10,874 training samples, 819 validation samples, and 1,300 test samples. The SNL dataset appears to be of high quality, with reference summaries effectively capturing the main ideas and facts from the corresponding articles. Nevertheless, a few low-quality samples were identified during the manual evaluation following the model training.

Two categories of articles, namely those related to Norwegian names and specific dates, did not contain appropriate reference summaries. Articles about names typically include statistics and historical context about the name, while articles about dates cover significant events and notable births or deaths that occurred on that particular date. Instead of containing proper summaries, the reference summaries included the text: "These were published in 2009 and have not been updated since. The editors of Store Norske Leksikon have not decided whether we should keep these articles but are leaving them out for now." The articles "Borgar" and "18. Juni" in Section E.2 serve as examples of these two categories of articles. Interestingly, despite these articles having reference summaries that do not accurately represent the article content, the models trained on SNL manage to produce summaries with near-perfect ROUGE scores (90), as they have learned to mimic the similarities in summaries across these article classes.

5.1.2 CNN/DM

The English version of CNN/DM serves as a widely recognized benchmark for models trained on extractive and abstractive summarization tasks. Our version was translated to Norwegian using the procedure explained in Section 3.1.2. Overall, the quality of the translated sentences in the samples appears satisfactory.

As shown in Table 4.2, the Norwegian translations of both the articles and summaries have fewer characters on average. This aligns with previous research indicating that Norwegian translations often are 5%-10% shorter compared to English [14]. The summaries exhibit a slightly higher compression rate compared to the English version, with the Norwegian having a compression rate of 13.01 and the English 13.67. The increased compression rate in the Norwegian summaries may indicate that they contain more concise and focused information. A more comprehensive analysis and experimentation could provide further insights into the effects of these differences on the performance of models trained on the translated dataset.

Abstractiveness

The density and coverage for the datasets shown in Table 4.3 show that there is a clear distinction between the SNL and CNN/DM datasets; CNN/DM exhibits a significantly higher extractiveness level compared to SNL. Specifically, the coverage for CNN/DM is 0.80, indicating that 80% of the words in its summaries are also found in the corresponding articles. In contrast, this figure drops to 57% for SNL.

The difference in density, which considers the average length of extracted sequences a word in the summary belongs to, is even more pronounced. The average length is 2.45 words for CNN/DM, while it is only 0.91 words for SNL. This substantial difference in density further underscores the higher extractiveness of CNN/DM. Indeed, these findings suggest that our translated version of CNN/DM shares many characteristics with its English counterpart, particularly in terms of the relatively low level of abstractiveness in the reference summaries [27, 102]. These characteristics can also be spotted in the KDE plots of density and coverage in Figure 4.3. Especially the density is distributed around a greater area for CNN/DM than for SNL, with a greater number of samples having densities between six and sixteen.

We also assessed abstractiveness using our novel metric, Match Ratio, which measures overlap at the sentence level, as opposed to the word or sequence level in the case of coverage and density, respectively. Interestingly, the Match Ratio for both datasets is close to zero, as shown in Table 4.3. This implies that there are almost no instances of entire sentences from the articles being included verbatim in the reference summaries.

5.2 Modelling

We successfully accomplished the second objective of this thesis, which entailed designing and implementing an automatic summarization model for Norwegian text using a transformer model. We achieved this by fine-tuning pre-trained North-T5 transformer models on the generated datasets discussed in the previous section. Following fine-tuning, we optimized the token decoding parameters by conducting a search across various configurations. Alongside training and optimizing the T5 models, we established two extractive summarization baselines, namely

LEAD and TextRank, which gave us some good data points to measure the effectiveness of the T5 models. This section discusses the training procedure for the four T5 models trained as well as the process of finding the best-performing token decoding parameters.

5.2.1 Training

Fine-tuning times

The fine-tuning times and hardware requirements for different models, as displayed in Table 4.4, show significant variations in the time and resources needed for optimal performance. T5-Base-SNL was trained using four GTX 3900 GPUs with 24 GB VRAM each, while the remaining models utilized four A40 GPUs with 48 GB VRAM each. This discrepancy in hardware used makes it difficult to directly compare the training times among the various model runs. However, it is evident that the training times for models on the two datasets differ substantially, even though the models share the same number of parameters.For example, the large variants with 1.2B parameters, trained with the same total batch size (64), took more than six times longer to reach early stopping (16h 34m for Large-CNN/DM compared to 2h 30m for Large-SNL), due to the differences in the complexity and size of the datasets. Although Large-CNN/DM needed three additional epochs to converge (16 vs. 13), this disparity is mainly due to the characteristics of the respective datasets.

The primary factor contributing to this difference is the discrepancy in the number of samples in the training sets, as presented in Table 4.1. The SNL dataset contains 10,874 samples, whereas the CNN/DM dataset comprises 47,171 samples. This larger training set size for CNN/DM significantly increases the time required for fine-tuning the model. Furthermore, as depicted by the token length distributions in Figures 4.1 and 4.2, the average number of tokens in an article for CNN/DM is 1,044, while for SNL, it is 556. Despite applying truncation to the first 512 tokens, models trained on the CNN/DM dataset must process a greater number of tokens on average, leading to longer training times.

Loss results

The visualization of the loss values on the train and validation sets throughout training, as depicted in Figure 4.4, demonstrates that the training set loss gradually decreased throughout the training process. Interestingly, the validation set loss, which was computed at the end of each epoch, dropped rapidly for the first few epochs for all four models but saw very little reduction for the remainder of the training. Notably, the plot reveals that the validation loss was lower for all models except Large-CNN/DM, which surpassed the validation loss towards the end of training. A lower validation loss than training loss indicates that the models perform better on unseen samples than on the training data. The primary reason for this is likely the use of dropout during the training process. Models employing dropout deactivate a subset of neurons during training, while all neurons remain active when calculating the validation loss.

Significant hyperparameters, such as dropout rate, optimizer, and others, were kept consistent across all training runs and were not tuned for optimal performance. Although the hyperparameters were selected based on prior research and best practices, it is possible that further optimization could have led to even lower loss values or reduced training times.

5.2.2 Optimal Token decoding parameters

Across all models, we found that the most effective decoding parameters were a combination of beam search and avoiding repeating n-grams as shown in Table 4.5. T5-Large-CNN/DM achieved the best result without repeating n-grams of size 5, while the other models performed optimally without repeating n-grams of size 3. There is a potential for achieving even better results by experimenting with a larger beam width (number of beams) during the beam search process. In our experiments, we limited the beam width to 5, which might constrain the search space and potentially exclude more suitable candidate sequences. A larger beam width would expand the search space, increasing the likelihood of discovering more optimal summaries.

Prior to our experiments, we anticipated that recent sampling-based token decoding techniques like eta-cutoff or nucleus sampling would yield superior results due to their widespread adoption and impressive results reported in their papers. However, the results show that sampling techniques underperformed compared to both greedy and various beam search configurations for almost every model based on the ROUGE scores. This subpar performance may likely be attributed to the nature of summary generation, which is less reliant on creativity than other text-generation tasks such as question-answering or story generation.

The main objective of a summarization model is to produce coherent and relevant summaries that accurately capture the key ideas of the source documents. Consequently, this task does not require a high degree of creativity, which may explain the weaker performance of sampling methods compared to more structured and focused approaches like beam search combined with the avoidance of repeating n-grams. Additionally, the lower ROUGE scores could be due to the tendency of sampling methods to select semantically similar words with different surface forms, such as choosing 'automobile' instead of 'car' when 'car' is present in the reference summary. This discrepancy can result in a lower match between the generated summary and the reference, thus impacting evaluation metrics. It is worth noting that ROUGE has a known weakness in capturing the similarity between words with similar meanings but different surface forms. Ideally, we should have also conducted human evaluations of the different token decoding parameters, but such an approach would have been extremely time-consuming.

As explained in Section 3.4, we manually created 12 different parameter configurations comprising greedy decoding, beam search, and sampling methods. While this approach allowed us to compare various strategies, employing a more sophisticated optimization process, such as using Optuna [123] or other hyperparameter optimization frameworks, might lead to improved hyperparameter settings. With Optuna, we could define a search space for the decoding parameters, including beam width and top-p values, and then apply an efficient optimization algorithm, like the Tree-structured Parzen Estimator (TPE) [124] or Bayesian optimization [125], to explore the hyperparameter space.

Such optimization techniques can systematically explore a vast hyperparameter space, identifying configurations that enhance overall model performance more effectively than manual tuning. By leveraging advanced optimization methods, we may potentially discover decoding strategies that further improve the quality of generated summaries while balancing computational requirements and overall model efficiency. However, it is important to note that sophisticated optimization on the validation set does not guarantee superior performance on the test set due to the possibility of overfitting. Employing a more rigorous evaluation methodology, such as cross-validation or held-out test sets, could provide more reliable estimates of model performance and help mitigate the risk of overfitting.

5.3 Evaluation

The final objective was to evaluate the developed summarization models on both the quantitative ROUGE metric and human evaluation. We did this by calculating ROUGE scores on the test set as well as performing human evaluation of different subsets of the datasets.

5.3.1 ROUGE Scores

Table 4.6 compares ROUGE scores between the different models trained on SNL and CNN/DM.

Across both the SNL and CNN/DM datasets, we found that the fine-tuned T5-Large models, with 1.2 billion parameters, consistently outperformed the T5-Base models with 580 million parameters and the extractive summarization baselines (LEAD-3 and TextRank) for all four ROUGE metrics. Intriguingly, the performance difference between T5-Large and T5-Base models was relatively small, despite the large model having nearly twice as many parameters. This finding is consistent with the scaling laws for neural networks, which propose that although increasing the number of parameters can improve performance, it seldom guarantees a significant enhancement proportional to the increase in parameters [126].

Interestingly, the ROUGE scores, in descending order, were typically highest for ROUGE-1, followed by ROUGE-L, ROUGE-Lsum, and finally ROUGE-2. This pattern can be explained by the nature of these metrics. ROUGE-1 measures the overlap of individual words (unigrams) between the generated summary and the reference summary, whereas ROUGE-2 takes into account the overlap of two consecutive words (bigrams). Therefore, ROUGE-2, requiring not only the correct words but also the correct sequence, is a more stringent metric. Given that our models often rephrase and condense information, preserving the exact wording and sequence from the source text is challenging, resulting in lower ROUGE-2 scores.

ROUGE-L and ROUGE-Lsum, on the other hand, are based on the Longest Common Subsequence (LCS) between the summary and the reference, which can match phrases and sentences of varying lengths and does not require exact sequential matching. As such, they generally yield scores between those of ROUGE-1 and ROUGE-2. However, they still consider the order of the words to some extent, explaining why their scores are lower than ROUGE-1 but higher than ROUGE-2 in our results.

We also observed that the extractive baselines, LEAD-3 and TextRank-3, performed considerably better on CNN/DM compared to SNL. This can be attributed to the fundamental differences in the datasets, with CNN/DM being less abstractive than SNL, as discussed in Section 5.1.2. Given that extractive methods rely on selecting existing sentences from the source text, they excel when the main points can be effectively captured through direct extraction. In the case of CNN/DM, crucial information is often contained within a few specific sentences, enabling extractive methods like LEAD-3 and TextRank-3 to produce accurate summaries. In contrast, the more abstractive nature of the SNL dataset requires greater paraphrasing and sentence rephrasing, posing a challenge for extractive methods and leading to their relatively lower performance on this dataset.

Another noteworthy observation from Table 4.6 is the variation in average generation length among the models and datasets. T5-Base and T5-Large produce shorter summaries compared to the extractive baselines, particularly for the CNN/DM dataset. Briefer summaries may be perceived as more concise and focused, which could be advantageous for specific applications. However, it is vital to determine whether these shorter summaries effectively convey the necessary information from the original text without omitting essential details. Subsequent analysis and human evaluation, as discussed in the next section, will offer a deeper understanding of the quality and informativeness of the generated summaries.

5.3.2 Human evaluation

We conducted a human evaluation on three distinct subsets of both datasets, comprising the 5 samples with the highest and lowest R-1 scores for the large variant, and 5 randomly selected samples. For each sample, we manually rated the quality of the summary on 5 different dimensions: Informativeness, Relevance, Fluency, Coherence, and Factuality. In total, we evaluated 30 samples with two predictions for each (T5-Base and T5-Large). This time-consuming process took several hours to complete. The results and corresponding average R-1 scores for SNL and CNN/DM can be found in Tables 4.7 and 4.8, respectively.

The analysis of the Lowest R-1 and Top R-1 subsets revealed that ROUGE scores do not always correlate with the average human evaluation scores. In fact, for both models, the subsets with the highest human evaluation scores were those with the lowest R-1 scores. This discrepancy can be attributed to well-written summaries that effectively captured the documents' essence but had minimal word overlap with the reference summaries. Examples of this can be found in Appendices E.1 for SNL and F.1 for CNN/DM. These examples underscore the limitations of reference-based metrics such as ROUGE and BLEU in accurately evaluating summary quality. A more effective method might involve collecting multiple summaries for each article or document, and then scoring a summary based on the highest ROUGE scores achieved among them [127].

Given the lower-than-expected correlation between ROUGE scores and human evaluation on the Lowest and Highest R-1 score subsets, the most reliable indicator of the models' true performance is the subset with 5 random articles selected from each dataset. For these subsets, the large variant of North-T5 achieved a marginally higher average score on SNL (+0.12) and a more significant margin on CNN/DM (+0.66).

We assessed summaries along two semantic dimensions: Informativeness, which gauged how effectively a summary captured the key points of the article, and Relevance, which evaluated the consistency of the summary's details with the article. The gap between the base and large models widened for both dimensions, suggesting that the small difference in ROUGE scores on the test sets becomes more pronounced when evaluating meaningful summary generation. In terms of syntactic dimensions (fluency and coherence), both models performed exceptionally well on both datasets, indicative of a high level of understanding of the Norwegian language and syntax.

Lastly, we included Factuality as an additional dimension to address potential hallucinations frequently exhibited by large language models. The base and large models scored equally on SNL, while the large model achieved a +0.66 higher score on CNN/DM. The hallucinations we observed were often challenging to identify, and in many cases, we noticed them only due to our prior knowledge of the topic covered in an article. This illustrates why we wanted to measure factuality. It is easy to be misled by the highly coherent and fluent text and to place undue trust in the facts presented by language models. The dangers are especially clear with chatGPT, a highly popular and publicly available language model, which has been shown to hallucinate and make up facts [128].

5.3.3 Model Abstractiveness

Similarly to how we explored the level of abstractiveness in the reference summaries in the datasets, we also calculated density, coverage and the novel match ratio (Section 2.2.3) for the generated summaries to better understand how the models work.

As demonstrated in Table 4.9, it is evident that the generated summaries lean more towards extraction than their corresponding reference summaries. This is particularly noticeable in the models trained on the CNN/DM dataset. For instance, the T5-Large-CNN/DM variant exhibits a match ratio of 0.24, a substantial increase from 0.01 as found in the reference summaries. Its density metric also increases to 9.78, compared to 2.45 in the reference summaries. These increased figures suggest that the models are resorting to a more extractive approach to the summarization task than intended from our training process, as seen with the same measures for the reference summaries.

Several factors could account for this increase in extractiveness. First, all the models were found to get the best ROUGE scores using beam search when selecting tokens. Previous research has indicated that beam search can cause a strong correlation between the certainty of sequences and a decrease in abstractiveness [129]. We did not investigate the level of abstractiveness that might result from other token decoding techniques, such as those based on sampling. However, it's reasonable to anticipate that these techniques would likely yield less extractive results. This expectation stems from the inherent randomness in the decoding process, which could potentially lead to more diverse and less verbatim outputs.

Secondly, by design, ROUGE rewards n-gram overlap between the generated and reference summaries. Consequently, this might inadvertently encourage the models to extract phrases directly from the source document, as it is a guaranteed method of maximizing n-gram overlap and thereby achieving higher ROUGE scores.

5.4 Out of Domain Behaviour

We generated summaries for a few samples from different sources than our fine-tuned summarization models were trained on. We did this to further understand the model's behaviour and the effect on training on different datasets.

Upon examining the averaged scores of the models on different data (as shown in Appendix G), one of the most striking patterns is the consistent performance of the models fine-tuned on the CNN/DM dataset compared to models trained on SNL. Regardless of the source - be it Aftenposten, Wikipedia, Kvinneguiden, or Komplett - the CNN/DM models displayed impressive robustness and adaptability, consistently scoring high in human evaluation across all five dimensions. This pattern suggests that the training on CNN/DM dataset, which comprises a wide variety of news articles, might have endowed the models with a level of versatility that enables them to effectively handle diverse text domains.

By inspecting the different summaries, we can see, in particular on Komplett.no in Appendix G.4 that the models generate summaries which are similar to the datasets they were trained on. For the models trained on SNL, the generated summaries read similarly to an encyclopedia article with summaries that include the various facts covered in the article. For example, the summary from Large-SNL starts with "Et skjerm-og graffikkort er.. " ("A display and graphics card is..") similar to how an SNL article would start. The models trained on CNN/DM on the other hand read much more like news articles and generate summaries of the same style as when evaluated on the CNN/DM dataset.

We also found that some of the same characteristics around the length of summaries and abstractness also being applicable to these summaries generated on different domains. The summaries from the CNN/DM models are longer and more thorough than for the SNL models and also more extractive as they primarily consist of longer sequences extracted.

5.5 Further Work

One considerable source of uncertainty in this study stemmed from the quality of certain samples within the datasets. To address this issue, future work should involve a more careful examination, selection and pre-processing of the generated datasets before training commences. This would involve identifying and excluding samples containing inadequate or uninformative reference summaries, as well as improving the overall quality and consistency of the dataset. By doing so, the reliability and robustness of the trained models can be significantly enhanced, possibly leading to better performance in generating summaries.

The limitations of our models, which only had access to the first 512 tokens in the articles, introduced a source of error, particularly for the CNN/DM dataset, as significant portions of the articles were not used for summarization. Expanding this limit to, for example, 1024 tokens would substantially increase memory usage. An alternative solution could involve adopting a two-stage approach, akin to the one proposed by Gehrmann et al. [130], which first extracts the most relevant portions of a document before performing abstractive summarization on the reduced text.

LoRA (Low-Rank Adaptation) [131], a recently developed fine-tuning technique, has gained popularity due to its ability to reduce the number of trainable parameters by up to 10,000 times while maintaining comparable performance to full fine-tuning. Similarly, LoRA could be combined with INT8 [132], a new datatype used in machine learning based on 8-bit numbers which further could reduce the memory usage and computational constraints we faced. Leveraging these techniques could enable experimentation with larger North-T5 variants, such as those with 3B and 11B parameters, without the computational constraints encountered in this study.

The abstractive models trained in this work can be considered black boxes, as their decision-making processes remain largely unexplainable. While the abstractiveness metrics and human evaluation offer some insights, many aspects of the model's behaviour, particularly in terms of hallucinations, remain unexplained. Further work could explore explainability methods for transformers, such as AttentionViz which analyzes the attention weights in the network to explain behaviour [133] or model-agnostic SHAP [134], to gain a deeper understanding of the models and potentially improve their performance.

We did not address the potential biases and toxic outputs that may arise from the trained abstractive models. Investigating and mitigating biases in both the training data and the model outputs is an essential aspect of creating responsible AI systems. Further work should focus on detecting and understanding biases present in the generated summaries, as well as devising methods to reduce the occurrence of toxic or harmful content. By incorporating these considerations, we can enhance the ethical and social aspects of our models, ensuring that they produce unbiased summaries.

CHAPTER 6

CONCLUSION

This research marked the first endeavour in developing systems capable of performing abstractive summarization for the Norwegian language.

Two datasets containing articles and reference summaries were generated based on a web scrape of Store Norske Leksikon (SNL) and translation based on a pre-trained OPUS-MT transformer model on CNN/Daily Mail (CNN/DM), a commonly used English summarization benchmarking dataset. The datasets were then used for training different summarization models. The primary model used is a Norwegian T5 language model based on the transformer architecture. We fine-tuned the base and large variants, containing 580M and 1.2B parameters respectively, on the generated datasets. Token decoding parameters were optimized using the validation sets by assessing the performance on a variety of token decoding parameters spanning beam search and other sampling techniques. Additionally, we created two extractive baselines using LEAD-3 and TextRank.

Overall, the T5 models outperform the extractive baselines for both datasets. For the T5 variants, the large variant demonstrated the strongest performance on the different ROUGE metrics with a ROUGE-1 score of 35.07 on SNL and 34.02 on CNN/DM. In our human evaluation, where summaries were ranked on their informativeness, relevance, fluency, coherence and factuality, it averaged 3.96/5 on SNL and 4.64/5 on CNN/DM on randomly selected samples averaged across all criteria measured. Although the models were trained to generate summaries using an abstractive approach, an analysis of coverage, density, and match ratio revealed a higher degree of extractiveness in the model-generated summaries than in the reference summaries used for training. For instance, the T5-large model trained on CNN/DM exhibited a match ratio of 0.26 and a density of 9.79, in contrast to the reference summaries, which had a match ratio of 0.01 and density of 2.45 on CNN/DM.

When the models were tasked with generating summaries for articles from domains different than those of SNL and CNN/DM, they were still able to produce informative content, despite not having been exposed to similar articles during training. Moreover, the stylistic attributes of the generated summaries seemed to reflect the characteristics of their training datasets. Specifically, models trained on SNL tended to generate summaries reminiscent of encyclopedia entries, while those trained on CNN/DM produced summaries bearing more resemblance to news articles.

The central research question that guided this study was *"Is it possible to create a machine learning system capable of performing abstractive summarization on the Norwegian natural lan-*

guage?" This question emerged from a recognition of the scarcity of natural language processing tools specifically designed for Norwegian, coupled with a desire to bridge this gap and contribute to the advancement of Norwegian language technology. Given the results we achieved, despite the limited computational resources and time, we can confidently affirm that it is indeed possible to create a machine learning system capable of performing abstractive summarization in Norwegian.

The evidence lies in the satisfactory ROUGE scores and human evaluation scores our models achieved. These scores are a testament to the models' proficiency in generating summaries that are not just semantically accurate but also contextually informative and linguistically coherent in Norwegian. The models demonstrated an impressive ability to adapt to different text domains and styles, a skill that underscores the versatility of our approach. Moreover, they highlighted the potential of the T5 architecture to be fine-tuned for languages and tasks beyond its original design, thus opening new avenues for future research and applications.

In conclusion, this study has shed light on the potential of machine learning and transformer architectures in enhancing our ability to process and understand Norwegian text. It has broadened the horizon of possibilities for Norwegian language technology and set a promising precedent for further exploration in the field of abstractive summarization for Norwegian.

BIBLIOGRAPHY

[1] Hans Peter Luhn. “The automatic creation of literature abstracts”. In: *IBM Journal of research and development* 2.2 (1958), pp. 159–165.

[2] Alec Radford et al. “Improving language understanding by generative pre-training”. In: (2018).

[3] Colin Raffel et al. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.* 2020. arXiv: `1910.10683 [cs.LG]`.

[4] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* ” O’Reilly Media, Inc.”, 2009.

[5] Hercules Dalianis and Eduard Hovy. “On Lexical Aggregation and Ordering”. In: *Eighth International Natural Language Generation Workshop (Posters and Demonstrations).* 1996. URL: `https://aclanthology.org/W96-0508`.

[6] Eugene Charniak. *Statistical language learning.* MIT press, 1996.

[7] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[8] Abigail See, Peter J. Liu, and Christopher D. Manning. “Get To The Point: Summarization with Pointer-Generator Networks”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 1073–1083. DOI: `10.18653/v1/P17-1099`. URL: `https://aclanthology.org/P17-1099`.

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

[10] Rico Sennrich, Barry Haddow, and Alexandra Birch. “Neural Machine Translation of Rare Words with Subword Units”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725. DOI: `10.18653/v1/P16-1162`. URL: `https://aclanthology.org/P16-1162`.

[11] Taku Kudo. “Subword regularization: Improving neural network translation models with multiple subword candidates”. In: *arXiv preprint arXiv:1804.10959* (2018).

[12] Taku Kudo and John Richardson. “Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing”. In: *arXiv preprint arXiv:1808.06226* (2018).

[13] Linting Xue et al. “mT5: A massively multilingual pre-trained text-to-text transformer”. In: *arXiv preprint arXiv:2010.11934* (2020).

[14] Andiamo. *How much will your text expand or contract?* `https://web.archive.org/web/20221209154703/https://www.andiamo.co.uk/resources/expansion-and-contraction-factors/`. Archived by `https://web.archive.org` on 2022-12-09. 2021.

[15] Hussain Alkharusi. “Categorical variables in regression analysis: A comparison of dummy and effect coding”. In: *International Journal of Education* 4.2 (2012), p. 202.

[16] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space.* 2013. arXiv: `1301.3781 [cs.CL]`.

[17] Singerep. *File:Word vector illustration.jpg — Wikimedia Commons, the free media repository.* [Online; accessed 25-March-2023; modified from the original]. 2022. URL: `https://commons.wikimedia.org/w/index.php?title=File:Word_vector_illustration.jpg&oldid=714073647`.

[18] Alec Radford et al. “Language models are unsupervised multitask learners”. In: *OpenAI blog* 1.8 (2019), p. 9.

[19] Shibhansh Dohare, Harish Karnick, and Vivek Gupta. “Text summarization using abstract meaning representation”. In: *arXiv preprint arXiv:1706.01678* (2017).

[20] Rada Mihalcea and Paul Tarau. “TextRank: Bringing order into texts”. In: *Proceedings of the 2004 conference on empirical methods in natural language processing.* 2004, pp. 404–411.

[21] Daniel Cer et al. *Universal Sentence Encoder.* 2018. arXiv: `1803.11175 [cs.CL]`.

[22] Alexis Conneau et al. *Supervised Learning of Universal Sentence Representations from Natural Language Inference Data.* 2018. arXiv: `1705.02364 [cs.CL]`.

[23] Yang Liu. “Fine-tune BERT for extractive summarization”. In: *arXiv preprint arXiv:1903.10318* (2019).

[24] Parth Mehta. “From Extractive to Abstractive Summarization: A Journey.” In: *ACL (Student Research Workshop).* Springer. 2016, pp. 100–106.

[25] Lu Wang and Claire Cardie. “Domain-independent abstract generation for focused meeting summarization”. In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* 2013, pp. 1395–1405.

[26] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. “Sequence to sequence learning with neural networks”. In: *Advances in neural information processing systems* 27 (2014).

[27] Max Grusky, Mor Naaman, and Yoav Artzi. “Newsroom: A Dataset of 1.3 Million Summaries with Diverse Extractive Strategies”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers).* New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 708–719. DOI: `10.18653/v1/N18-1065`. URL: `https://aclanthology.org/N18-1065`.

[28] Tulu Tilahun Hailu, Junqing Yu, Tessfu Geteye Fantaye, et al. “Intrinsic and Extrinsic Automatic Evaluation Strategies for Paraphrase Generation Systems”. In: *Journal of Computer and Communications* 8.02 (2020), p. 1.

[29] Chin-Yew Lin. “Rouge: A package for automatic evaluation of summaries”. In: *Text summarization branches out.* 2004, pp. 74–81.

[30] Kishore Papineni et al. "Bleu: a Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, July 2002, pp. 311–318. DOI: `10.3115/1073083.1073135`. URL: `https://aclanthology.org/P02-1040`.

[31] B. Dorr et al. "A Methodology for Extrinsic Evaluation of Text Summarization: Does ROUGE Correlate?" In: *IEEvaluation@ACL*. 2005.

[32] Bonnie Dorr et al. "A methodology for extrinsic evaluation of text summarization: does ROUGE correlate?" In: *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*. 2005, pp. 1–8.

[33] W John Hutchins. "Machine translation: A brief history". In: *Concise history of the language sciences*. Elsevier, 1995, pp. 431–445.

[34] Peter F Brown et al. "The mathematics of statistical machine translation: Parameter estimation". In: (1993).

[35] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016. arXiv: `1409.0473 [cs.CL]`.

[36] Chris Callison-Burch, Miles Osborne, and Philipp Koehn. "Re-evaluating the Role of Bleu in Machine Translation Research". In: *Conference of the European Chapter of the Association for Computational Linguistics*. 2006.

[37] Tom Michael Mitchell et al. *Machine learning*. Vol. 1. McGraw-hill New York, 2007.

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[39] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. "Thumbs up? Sentiment classification using machine learning techniques". In: *arXiv preprint cs/0205070* (2002).

[40] John A Hartigan and Manchek A Wong. "Algorithm AS 136: A k-means clustering algorithm". In: *Journal of the royal statistical society. series c (applied statistics)* 28.1 (1979), pp. 100–108.

[41] Ian T Jolliffe. "Principal component analysis: a beginner's guide—I. Introduction and application". In: *Weather* 45.10 (1990), pp. 375–382.

[42] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[43] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *nature* 518.7540 (2015), pp. 529–533.

[44] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *nature* 529.7587 (2016), pp. 484–489.

[45] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[46] OpenAI. *Introducing ChatGPT: Our Latest Advancement in Natural Language Processing*. Archived by `https://web.archive.org` on 2023-04-18. 2022. URL: `https://web.archive.org/web/20230418171039/https://openai.com/blog/chatgpt`.

[47] Jason Yosinski et al. "How transferable are features in deep neural networks?" In: *Advances in neural information processing systems* 27 (2014).

[48] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: `10.1109/CVPR.2009.5206848`.

[49] Sinno Jialin Pan and Qiang Yang. “A survey on transfer learning”. In: *IEEE Transactions on knowledge and data engineering* 22.10 (2010), pp. 1345–1359.

[50] Alex Krizhevsky, Geoffrey Hinton, et al. “Learning multiple layers of features from tiny images”. In: (2009).

[51] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444.

[52] Sebastian Ruder. “Neural transfer learning for natural language processing”. PhD thesis. NUI Galway, 2019.

[53] Gringer. *File:Overfitting.svg — Wikimedia Commons, the free media repository.* [Online; accessed 22-April-2023; modified from the original]. 2023. URL: `https://commons.wikimedia.org/wiki/File:Overfitting_svg.svg`.

[54] Trevor Hastie et al. *The elements of statistical learning: data mining, inference, and prediction.* Vol. 2. Springer, 2009.

[55] Hubert L Dreyfus. *What computers still can’t do: A critique of artificial reason.* MIT press, 1992.

[56] TseKiChun. *File:Neural network explain.png — Wikimedia Commons, the free media repository.* [Online; accessed 27-April-2023]. 2021. URL: `https://upload.wikimedia.org/wikipedia/commons/d/d2/Neural_network_explain.png`.

[57] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors”. In: *nature* 323.6088 (1986), pp. 533–536.

[58] Sebastian Raschka and Vahid Mirjalili. *Python Machine Learning, 3rd Ed.* Birmingham, UK: Packt Publishing, 2019, p. 748. ISBN: 978-1789955750.

[59] Nitish Shirish Keskar et al. *On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima.* 2017. arXiv: `1609.04836 [cs.LG]`.

[60] Yann LeCun et al. “Efficient backprop”. In: *Neural networks: Tricks of the trade.* Springer, 2002, pp. 9–50.

[61] Varun Godbole et al. *Deep Learning Tuning Playbook.* Version 1. 2023. URL: `https://github.com/google-research/tuning_playbook`.

[62] Jun Han and Claudio Moraga. “The influence of the sigmoid function parameters on the speed of backpropagation learning”. In: *From Natural to Artificial Neural Computation: International Workshop on Artificial Neural Networks Malaga-Torremolinos, Spain, June 7–9, 1995 Proceedings 3.* Springer. 1995, pp. 195–201.

[63] Sepp Hochreiter. “The vanishing gradient problem during learning recurrent neural nets and problem solutions”. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6.02 (1998), pp. 107–116.

[64] Kunihiko Fukushima. “Cognitron: A self-organizing multilayered neural network”. In: *Biological cybernetics* 20.3-4 (1975), pp. 121–136.

[65] Dan Hendrycks and Kevin Gimpel. *Gaussian Error Linear Units (GELUs).* 2020. arXiv: `1606.08415 [cs.LG]`.

[66] John Bridle. “Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters”. In: *Advances in neural information processing systems* 2 (1989).

[67] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[68] Mads Dyrmann. *Neural Network Dropout.svg.* [Online; accessed 22-April-2023; modified from the original]. 2021. URL: `https://commons.wikimedia.org/wiki/File:Neural_Network_Dropout.svg`.

[69] Alexander Pauls and J Yoder. "Determining optimum drop-out rate for neural networks". In: *Midwest Instructional Computing Symposium (MICS)*. 2018.

[70] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization.* 2019. arXiv: `1711.05101 [cs.LG]`.

[71] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning.* MIT press, 2016.

[72] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization.* 2016. arXiv: `1607.06450 [stat.ML]`.

[73] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* 2015. arXiv: `1502.03167 [cs.LG]`.

[74] Jeffrey L Elman. "Finding structure in time". In: *Cognitive science* 14.2 (1990), pp. 179–211.

[75] fdeloche. *File:Recurrent neural network unfold.svg — Wikimedia Commons, the free media repository.* [Online; accessed 27-April-2023]. 2022. URL: `https://upload.wikimedia.org/wikipedia/commons/b/b5/Recurrent_neural_network_unfold.svg`.

[76] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[77] Kyunghyun Cho et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *arXiv preprint arXiv:1406.1078* (2014).

[78] Medhabarve. *File:Basic Encoder-Decoder Architecture.png — Wikimedia Commons, the free media repository.* [Online; accessed 27-April-2023]. 2022. URL: `https://upload.wikimedia.org/wikipedia/commons/f/f8/Basic_Encoder-Decoder_Architecture.png`.

[79] Savas Yildirim and Meysam Asgari-Chenaghlu. *Mastering Transformers: Build state-of-the-art models from scratch with advanced natural language processing techniques.* Packt Publishing Ltd, 2021.

[80] Yun Chen et al. *A Stable and Effective Learning Strategy for Trainable Greedy Decoding.* 2018. arXiv: `1804.07915 [cs.CL]`.

[81] Alex Graves. *Sequence Transduction with Recurrent Neural Networks.* 2012. arXiv: `1211.3711 [cs.NE]`.

[82] Patrick von Platen. *How to generate text: using different decoding methods for language generation with Transformers.* Archived by `https://web.archive.org` on 2023-03-28. 2023. URL: `https://web.archive.org/web/20230328003458/https://huggingface.co/blog/how-to-generate`.

[83] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. "A neural probabilistic language model". In: *Advances in neural information processing systems* 13 (2000).

[84] Ari Holtzman et al. *The Curious Case of Neural Text Degeneration.* 2020. arXiv: `1904.09751 [cs.CL]`.

[85] Nitish Shirish Keskar et al. *CTRL: A Conditional Transformer Language Model for Controllable Generation.* 2019. arXiv: `1909.05858 [cs.CL]`.

[86] John Hewitt, Christopher D Manning, and Percy Liang. "Truncation Sampling as Language Model Desmoothing". In: *arXiv preprint arXiv:2210.15191* (2022).

[87] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.* 2021. arXiv: `2010.11929 [cs.CV]`.

[88] Iz Beltagy, Matthew E Peters, and Arman Cohan. “Longformer: The long-document transformer”. In: *arXiv preprint arXiv:2004.05150* (2020).

[89] Sinong Wang et al. *Linformer: Self-Attention with Linear Complexity*. 2020. arXiv: 2006.04768 [cs.LG].

[90] Tri Dao et al. *FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness*. 2022. arXiv: 2205.14135 [cs.LG].

[91] Alexander Wettig et al. *Should You Mask 15% in Masked Language Modeling?* 2023. arXiv: 2202.08005 [cs.CL].

[92] Tom B Brown et al. “Language Models are Few-Shot Learners”. In: *arXiv preprint arXiv:2005.14165* (2020).

[93] Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. “Neural network acceptability judgments”. In: *Transactions of the Association for Computational Linguistics* 7 (2019), pp. 625–641.

[94] Daniel Cer et al. “SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation”. In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, 2017. DOI: 10.18653/v1/s17-2001. URL: https://doi.org/10.18653%2Fv1%2Fs17-2001.

[95] Thomas Indrias Biniam and Adam Morén. *Extractive Text Summarization of Norwegian News Articles Using BERT*. 2021.

[96] Ramesh Nallapati et al. “Abstractive text summarization using sequence-to-sequence rnns and beyond”. In: *arXiv preprint arXiv:1602.06023* (2016).

[97] Jörg Tiedemann and Santhosh Thottingal. “OPUS-MT – Building open translation services for the World”. In: *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*. Lisboa, Portugal: European Association for Machine Translation, Nov. 2020, pp. 479–480. URL: https://aclanthology.org/2020.eamt-1.61.

[98] Jörg Tiedemann. “The Tatoeba Translation Challenge–Realistic Data Sets for Low Resource and Multilingual MT”. In: *arXiv preprint arXiv:2010.06354* (2020).

[99] Marcin Junczys-Dowmunt et al. “Marian: Fast neural machine translation in C++”. In: *arXiv preprint arXiv:1804.00344* (2018).

[100] Helsinki NLP. *English-Norwegian Translation Model*. Archived by https://web.archive.org on 2023-04-07. 2023. URL: https://web.archive.org/web/20230407151908/https://github.com/Helsinki-NLP/Tatoeba-Challenge/tree/d20a5002da37b291fe54a6a5b0a81b6 models/eng-nor.

[101] Jon-Mikkel Korsvik and Jørgen Navjord. *jkorsvik/opus-mt-eng-nor*. https://huggingface.co/jkorsvik/opus-mt-eng-nor. (Accessed on 23/01/2023).

[102] Danqi Chen, Jason Bolton, and Christopher D Manning. “A thorough examination of the cnn/daily mail reading comprehension task”. In: *arXiv preprint arXiv:1606.02858* (2016).

[103] Vincent Chen, Eduardo Torres Montaño, and Liezl Puzon. “An examination of the CNN/DailyMail neural summarization task”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. 2017, pp. 2358–2367.

[104] Romain Paulus, Caiming Xiong, and Richard Socher. “A deep reinforced model for abstractive summarization”. In: *arXiv preprint arXiv:1705.04304* (2017).

[105] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. *Don’t Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization*. 2018. arXiv: 1808.08745 [cs.CL].

[106] Bogdan Gliwa et al. "SAMSum Corpus: A Human-annotated Dialogue Dataset for Abstractive Summarization". In: *Proceedings of the 2nd Workshop on New Frontiers in Summarization*. Association for Computational Linguistics, 2019. DOI: 10.18653/v1/d19-5409. URL: https://doi.org/10.18653%2Fv1%2Fd19-5409.

[107] National Library of Norway. *Norsk aviskorpus*. https://web.archive.org/web/20221126220115/https://www.nb.no/sprakbanken/ressurskatalog/oai-nb-no-sbr-4/. Archived by https://web.archive.org on 2022-11-26. 2020.

[108] Linting Xue et al. *mT5: A massively multilingual pre-trained text-to-text transformer*. 2021. arXiv: 2010.11934 [cs.CL].

[109] Yennie Jun. *All languages are not created (tokenized) equal*. May 2023. URL: https://blog.yenniejun.com/p/all-languages-are-not-created-tokenized.

[110] Edward Loper and Steven Bird. *NLTK: The Natural Language Toolkit*. 2002. arXiv: cs/0205028 [cs.CL].

[111] Vladimir I Levenshtein et al. "Binary codes capable of correcting deletions, insertions, and reversals". In: *Soviet physics doklady*. Vol. 10. 8. Soviet Union. 1966, pp. 707–710.

[112] Mišo Belica. *Automatic text summarizer*. Archived by https://web.archive.org on 2023-14-02. 2023. URL: https://web.archive.org/web/20230214055316/https://github.com/miso-belica/sumy.

[113] Per E Kummervold. *t5_base_NCC_lm*. https://huggingface.co/north/t5_base_NCC_lm. Accessed: 2023-04-05. 2023.

[114] Per E Kummervold et al. *Operationalizing a National Digital Library: The Case for a Norwegian Transformer Model*. 2021. arXiv: 2104.09617 [cs.CL].

[115] Per Kummervold, Freddy Wetjen, and Javier de la Rosa. "The Norwegian Colossal Corpus: A Text Corpus for Training Large Norwegian Language Models". In: *Proceedings of the Thirteenth Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, June 2022, pp. 3852–3860. URL: https://aclanthology.org/2022.lrec-1.410.

[116] Paulius Micikevicius et al. *Mixed Precision Training*. 2018. arXiv: 1710.03740 [cs.AI].

[117] Thomas Wolf et al. *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. 2020. arXiv: 1910.03771 [cs.CL].

[118] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703 [cs.LG].

[119] Martín Abadi et al. *TensorFlow: A system for large-scale machine learning*. 2016. arXiv: 1605.08695 [cs.DC].

[120] Jeff Rasley et al. "Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 3505–3506.

[121] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: https://www.wandb.com/.

[122] Razvan Azamfirei, Sapna R Kudchadkar, and James Fackler. "Large language models and the perils of their hallucinations". In: *Critical Care* 27.1 (2023), pp. 1–2.

[123] Takuya Akiba et al. *Optuna: A Next-generation Hyperparameter Optimization Framework*. 2019. arXiv: 1907.10902 [cs.LG].

[124] James Bergstra et al. "Algorithms for hyper-parameter optimization". In: *Advances in neural information processing systems* 24 (2011).

[125] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. *Practical Bayesian Optimization of Machine Learning Algorithms.* 2012. arXiv: `1206.2944 [stat.ML]`.

[126] Jared Kaplan et al. “Scaling laws for neural language models”. In: *arXiv preprint arXiv:2001.08361* (2020).

[127] Mousumi Akter, Naman Bansal, and Shubhra Kanti Karmaker. “Revisiting Automatic Evaluation of Extractive Summarization Task: Can We Do Better than ROUGE?” In: *Findings of the Association for Computational Linguistics: ACL 2022.* 2022, pp. 1547–1560.

[128] Yejin Bang et al. *A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity.* 2023. arXiv: `2302.04023 [cs.CL]`.

[129] Jiacheng Xu, Shrey Desai, and Greg Durrett. “Understanding neural abstractive summarization models via uncertainty”. In: *arXiv preprint arXiv:2010.07882* (2020).

[130] Sebastian Gehrmann, Yuntian Deng, and Alexander M Rush. “Bottom-up abstractive summarization”. In: *arXiv preprint arXiv:1808.10792* (2018).

[131] Edward J. Hu et al. *LoRA: Low-Rank Adaptation of Large Language Models.* 2021. arXiv: `2106.09685 [cs.CL]`.

[132] Tim Dettmers et al. *LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale.* 2022. arXiv: `2208.07339 [cs.LG]`.

[133] Catherine Yeh et al. *AttentionViz: A Global View of Transformer Attention.* 2023. arXiv: `2305.03210 [cs.HC]`.

[134] Scott Lundberg and Su-In Lee. *A Unified Approach to Interpreting Model Predictions.* 2017. arXiv: `1705.07874 [cs.AI]`.

APPENDIX A PYTHON MODULES

Module	Purpose
datasets	Handles loading and processing of datasets for machine learning tasks
deepspeed	Optimizes deep learning computations for PyTorch, particularly in distributed systems
evaluate	Provides a set of evaluation metrics for assessing machine learning models
huggingface/accelerate	Offers utility functions for accelerating Hugging Face models
huggingface/peft	Tracks performance of Hugging Face models
huggingface/transformers	Includes a collection of pre-trained transformer models for various NLP tasks
huggingface/evaluate	Assists in evaluating machine learning models on various benchmarks and datasets
huggingface_hub	Manages model sharing and versioning for Hugging Face models
nltk	Supports a wide range of natural language processing tasks
numpy	Enables numerical computations in Python
onnx	Facilitates sharing of machine learning models across different frameworks via the Open Neural Network Exchange format
onnxruntime-gpu	Provides GPU support for ONNX runtime
pandas	Used for data manipulation and analysis in Python
rouge	Computes ROUGE scores for the evaluation of text summaries
seaborn	Aids in statistical data visualization in Python
spacy	An industrial-strength library for natural language processing in Python
sumy	An automatic text summarization library that includes TextRank among other algorithms
thefuzz	Handles string matching, manipulation, and fuzzy matching using Levenshtein distances
wandb	Allows for experiment tracking, dataset versioning, and model management

Table A.1: List of Python modules used in this thesis and their purposes

APPENDIX B DATASET AND MODEL URLS

Links to the different datasets generated and models trained in this thesis.

Datasets

Dataset	Description
SNL	Lead paragraphs and remaining articles from Store Norske Leksikon (SNL)
CNN/DM	Translated version of CNN/Daily Mail
XSUM	Extreme summarization (one sentence summaries) of articles from BBC
SamSum	Summaries of short dialogues
VG	Ingress and articles from VG (Norwegian newspaper)
Wikipedia	Lead paragraphs and remaining articles from Norwegian Wikipedia

Table B.1: Different datasets generated in this work. The dataset names are clickable and will direct to their respective Huggingface Hub pages.

Fine-tuned Models

Model	Dataset
north/t5_base_NCC_lm	-
north/t5_large_NCC_lm	-
T5-Base-SNL	SNL
T5-Large-SNL	SNL
T5-Base-CNNDM	CNN/DM
T5-Large-CNNDM	CNN/DM

Table B.2: Overview of the different North-T5 used and the fine-tuned summarization models. Model names are hyperlinked to their respective Huggingface Hub pages.

APPENDIX C

ROUGE THROUGHOUT TRAINING

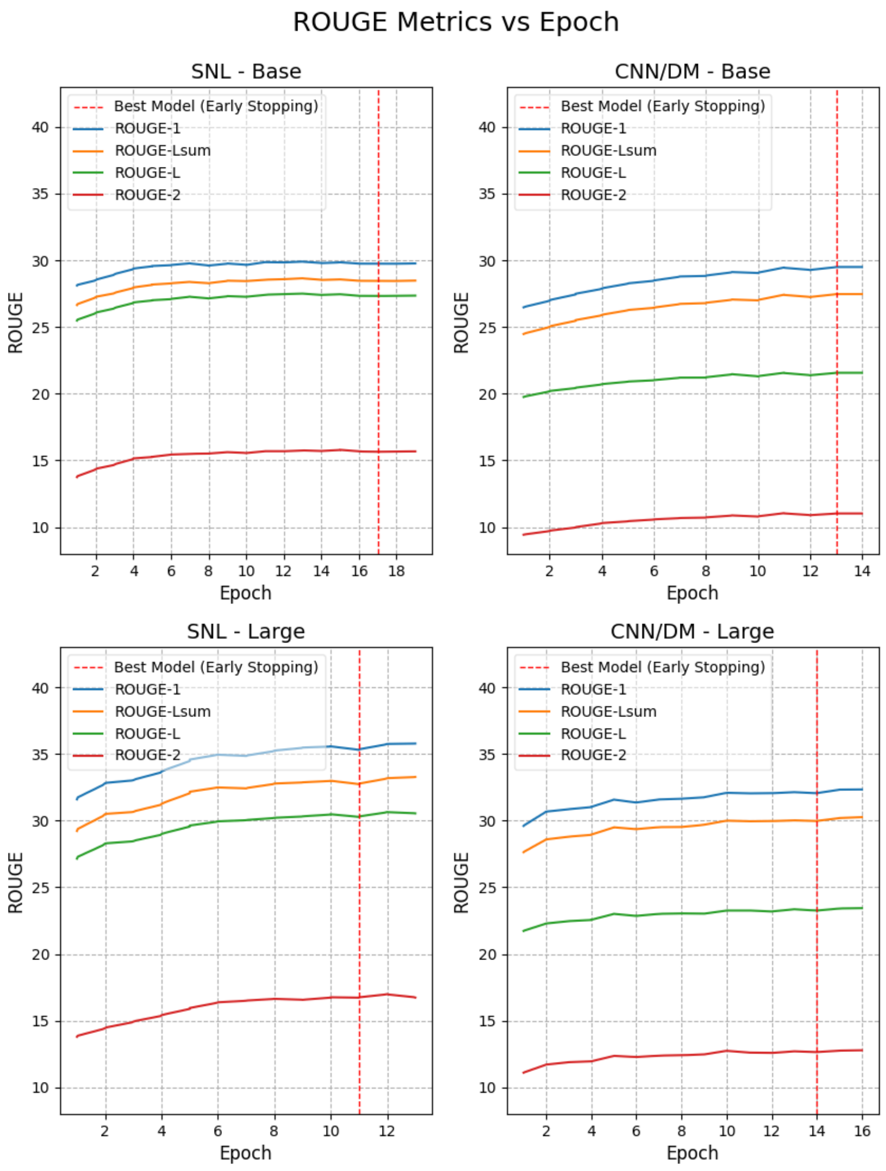


Figure C.1: ROUGE scores for T5 models during training. The vertical dashed red line indicates the epoch where the model achieved the minimum validation loss. For Base-CNN/DM, the ROUGE values were not logged in wandb for the final epoch for unknown reasons.

APPENDIX D

TOKEN DECODING HYPERPARAMETERS

D.1 Parameter search results

D.1.1 SNL Models

T5-Base-SNL

Config name	R-1	R-2	R-L	R-Lsum	Avg. Gen. Len.
Greedy	32.94	14.90	28.55	30.77	46.34
Beam default	30.78	14.32	26.80	28.65	69.45
Beam No-Repeat N-Gram Size: 3	**34.01**	**15.72**	**28.84**	**31.60**	45.32
Beam No-Repeat N-Gram Size: 5	33.63	15.69	28.68	31.33	47.75
Beam Encoder No-Repeat N-Gram Size: 5	30.52	13.98	26.73	28.50	69.31
Beam Repetition Penalty: 1.2	32.06	14.79	27.45	29.79	58.89
Sample Top-K: 50	29.87	10.73	24.02	27.03	59.42
Sample Temperature: 1.2 Top-K: 50	27.93	9.00	21.73	25.05	68.73
Sample Top-P: 0.7	32.73	13.81	27.41	30.16	49.16
Sample Top-P: 0.95	30.67	11.57	24.92	28.03	55.91
Sample Eta: 0.0006	29.81	10.94	23.96	26.99	59.56
Sample Eta: 0.0003	30.12	11.16	24.05	27.26	59.82

Table D.1: Hyperparameter search results for T5-Base-SNL on the validation set. The best-performing model for each ROUGE metric is highlighted in bold.

T5-Large-SNL

Config name	R-1	R-2	R-L	R-Lsum	Avg. Gen. Len.
Greedy	35.11	16.66	30.17	32.61	42.66
Beam default	33.84	16.57	29.38	31.41	57.25
Beam No-Repeat N-Gram Size: 3	**36.02**	17.39	**30.64**	**33.41**	41.86
Beam No-Repeat N-Gram Size: 5	35.64	**17.42**	30.54	33.14	43.32
Beam Encoder No-Repeat N-Gram Size: 5	33.50	15.88	29.17	31.10	55.64
Beam Repetition Penalty: 1.2	34.82	17.00	30.03	32.36	49.77
Sample Top-K: 50	31.54	12.35	25.73	28.80	55.49
Sample Temperature: 1.2 Top-K: 50	29.40	10.64	23.56	26.59	63.29
Sample Top-P: 0.7	34.70	15.68	29.24	32.06	46.33
Sample Top-P: 0.95	32.59	13.28	26.83	29.84	53.18
Sample Eta: 0.0006	31.90	12.36	26.03	29.10	55.68
Sample Eta: 0.0003	31.47	12.50	25.63	28.55	55.49

Table D.2: Hyperparameter search results for T5-Large-SNL on the validation set. The best-performing model for each ROUGE metric is highlighted in bold.

D.1.2 CNN/DM Models

T5-Base-CNN/DM

Config name	R-1	R-2	R-L	R-Lsum	Avg. Gen. Len.
Greedy	29.58	11.05	21.52	27.52	77.16
Beam default	29.52	11.11	21.02	27.15	105.56
Beam No-Repeat N-Gram Size: 3	**32.28**	12.15	22.00	**29.84**	93.96
Beam No-Repeat N-Gram Size: 5	32.14	**12.22**	**22.01**	29.73	98.02
Beam Encoder No-Repeat N-Gram Size: 5	27.48	8.34	19.27	25.38	82.86
Beam Repetition Penalty: 1.2	30.51	11.51	21.46	28.11	101.76
Sample Top-K: 50	27.66	8.27	18.14	25.61	84.37
Sample Temperature: 1.2 Top-K: 50	25.68	6.46	16.29	23.68	87.86
Sample Top-P: 0.7	29.89	10.51	20.63	27.75	81.50
Sample Top-P: 0.95	28.47	8.89	18.92	26.41	84.05
Sample Eta: 0.0006	27.74	8.31	18.22	25.75	85.57
Sample Eta: 0.0003	27.90	8.41	18.35	25.89	85.33

Table D.3: Hyperparameter search results for T5-Base-CNN/DM on the validation set. The best-performing model for each ROUGE metric is highlighted in bold.

T5-Large-CNN/DM

Config name	**R-1**	**R-2**	**R-L**	**R-Lsum**	**Avg. Gen. Len.**
Greedy	32.21	12.72	**23.25**	30.09	73.12
Beam default	32.43	12.83	22.92	30.06	98.45
Beam No-Repeat N-Gram Size: 3	**33.27**	13.05	23.03	**30.86**	91.88
Beam No-Repeat N-Gram Size: 5	33.26	**13.20**	23.17	**30.86**	95.59
Beam Encoder No-Repeat N-Gram Size: 5	30.64	9.65	21.05	28.40	74.60
Beam Repetition Penalty: 1.2	32.85	13.05	23.06	30.44	97.18
Sample Top-K: 50	28.76	8.91	19.09	26.63	77.90
Sample Temperature: 1.2 Top-K: 50	26.33	6.98	16.99	24.30	80.94
Sample Top-P: 0.7	31.74	11.72	22.16	29.57	75.59
Sample Top-P: 0.95	29.90	9.82	20.21	27.67	77.09
Sample Eta: 0.0006	29.00	9.08	19.27	26.82	77.60
Sample Eta: 0.0003	28.97	9.08	19.32	26.92	77.82

Table D.4: Hyperparameter search results for T5-Large-CNN/DM on the validation set. The best-performing model for each ROUGE metric is highlighted in bold.

D.2 Parameter configurations

Name	Rep. Penalty	No-Rep. N-Gram	Enc. No-Rep. N-Gram	Temp.	Top-P	Eta Cutoff
Default	1.0	0	0	1	1.0	0.0
Greedy configurations						
Greedy	–	–	–	–	–	–
Beam search configurations						
Beam default	–	–	–	–	–	–
No-Repeat N-Gram Size: 3	–	3	–	–	–	–
No-Repeat N-Gram Size: 5	–	5	–	–	–	–
Encoder No-Repeat N-Gram Size: 5	–	–	5	–	–	–
Repetition Penalty: 1.2	1.2	–	–	–	–	–
Sampling configurations						
Top-K: 50	–	–	–	–	–	–
Top-K: 50, Temperature: 1.2	–	–	–	1.2	–	–
Top-P: 0.7	–	–	–	–	0.7	–
Top-P: 0.95	–	–	–	–	0.95	–
Eta: 0.0006	–	–	–	–	–	0.0006
Eta: 0.0003	–	–	–	–	–	0.0003

Table D.5: Different generation parameters used for optimizing summarizations. Default parameters are the default generation parameters used in Huggingface Transformers. The '-' symbol represents a configuration that uses the default parameter value. Configurations are divided into three groups: Greedy, Beam Search, and Sampling configurations.

APPENDIX E SNL EXAMPLES

E.1 Lowest ROUGE-1

Examples of two of the summaries which got the lowest ROUGE-1 scores on T5-Large-SNL with their corresponding references

Alliance Internationale de Tourisme

Document: Organisasjonen ble stiftet i 1898 som en ideell, ikke-statlig organisasjon som representrerte interessene til nasjonale foreninger og forbund. Den har hatt sitt nåværende navn siden 1919. AIT hadde tidlig tilslutning fra blant annet sykkelorganisasjoner, men senere har motororganisasjonene fått dominerende plass. [...]

Reference: Alliance Internationale de Tourisme, verdensomfattende reiselivsorganisasjon med hovedkontor i Genève.

T5-Base-SNL (R-1: 4.25) AIT er en internasjonal organisasjon for reiselivs- og motororganisasjoner. Organisasjonen ble stiftet i 1898 som en ideell, ikke-statlig organisasjon som representerer interessene til nasjonale foreninger og forbund. AIT har konsultativ status ved FN siden 1947.

T5-Large-SNL (R-1: 8.0) ACI er en internasjonal organisasjon for reiselivs- og motororganisasjoner. Organisasjonen ble grunnlagt i 1898.

Brakteater

Document: Brakteater i Norge Norske brakteater preget kort etter 1100 ser ut til å være de første brakteatmynter overhodet. Med unntak av perioden rundt 1280-1319 var brakteaten den dominerende myntformen i Norge fra begynnelsen av 1100-tallet til slutten av 1300-tallet. Det ble preget brakteater i Norge frem til begynnelsen av 1500-tallet, de siste under erkebiskop Erik Valkendorf. [...]

Reference: Brakteater var mynter med preg bare på den ene siden. Blanketten på brakteaterne var så tynn at adversens opphøyde preg vises negativt på reversen.

T5-Base-SNL (R-1: 11.54) Brakteat er en myntform som ble preget i Norge fra 1100-tallet til slutten av 1500-tallet. I Norge var brakteat den dominerende myntformen i nordisk middelalder.

T5-Large-SNL (R-1: 8.33) Brakteater er en middelaldermynt som er preget med en liten billedflate til rådighet. Myntene er ofte enkle og enkle i utformingen, men har ofte et enkelt motiv på mynten. De norske brakteatmyntene ble preget i Norge fra begynnelsen av 1100-tallet til slutten av 1300-tallet.

E.2 Highest ROUGE-1

Examples of two of the summaries which got the highest ROUGE-1 scores on T5-Large-SNL with their corresponding references.

Borgar

Document: Namnet har fleire variantar: Borgar, Borger.Namnedag er 11. juni. Bjørg og Bjørge har også namnedag denne dagen. Namnestatistikk I perioden 1900–2017 var det omtrent 360 menn (fødde i Noreg) som hadde Borgar som det einaste førenamnet eller som det fyrste av fleire. [...]

Reference: Borgar er eit mannsnamn som går tilbake til det norrøne Borg-arr. I dette toledda namnet tyder forleddet 'berging; vern' og etterleddet 'krigar'.

T5-Base-SNL (R-1: 88.89) Borgar er eit mannsnamn som går tilbake til det norrøne Borgarr. I dette toledda namnet tyder forleddet 'strid' og etterleddet' 'gud'. Disse ble publisert i 2009 og er ikke blitt oppdatert etterpå. Redaksjonen i Store norske leksikon har ikke bestemt oss for om vi skal beholde disse artiklene, men lar dem ligge ute enn så lenge.

T5-Large-SNL (R-1: 93.83) Borgar er eit mannsnamn som går tilbake til det norrøne Borg-r. I dette toledda namnet tyder forleddet 'berging' og etterleddet' krigar'.

18. Juni

Document: Les om måneden juni.18. juni er nasjonaldag i Seychellene.Bjarne og Bjørn har navnedag.Noen kjente personer med fødselsdag 18. juni er Feofan Prokopovitsj (1681), Miklós Horthy (1868), Anastasia (1901), Otto Grieg Tidemand (1921), Kjell (Arve) Lund (1927), Reidar Thomassen (1936), Ailo Gaup (1944), Torstein Bugge Høverstad (1944) og Bent Stiansen (1963).18. juni 1928 døde Roald Amundsen.Noen andre historiske begivenheter som har foregått 18. juni: [...]

Reference: Dette er en datoartikkel. Disse ble publisert i 2009 og er ikke blitt oppdatert etterpå. Redaksjonen i Store norske leksikon har ikke bestemt oss for om vi skal beholde disse artiklene, men lar dem ligge ute enn så lenge.

T5-Base-SNL (R-1: 100.0) Dette er en datoartikkel. Disse ble publisert i 2009 og er ikke blitt oppdatert etterpå. Redaksjonen i Store norske leksikon har ikke bestemt oss for om vi skal beholde disse artiklene, men lar dem ligge ute enn så lenge.

T5-Large-SNL (R-1: 100.0) Dette er en datoartikkel. Disse ble publisert i 2009 og er ikke blitt oppdatert etterpå. Redaksjonen i Store norske leksikon har ikke bestemt oss for om vi skal beholde disse artiklene, men lar dem ligge ute enn så lenge.

E.3 Random samples

Examples of some random samples from the test split of the SNL dataset

Årevinger

Document: Andre grupper av årevinger som for eksempel darwinvepser og plantevepser er mindre kjente, til tross for at de omfatter svært mange arter både på verdensbasis og her i landet.Årevingene utgjør en egen orden blant insektene, som også kalles veps. [...]

Reference: Årevinger er en stor gruppe insekter med mer enn 4000 arter bare i Norge. Den omfatter velkjente insekter som maur, stikkeveps og humler, og husdyret honningbie.

T5-Base-SNL (R-1: 38.09) Årevinger er en gruppe insekter i ordenen veps. De omfatter rovdyr, pollenspisere, honningsugere, gallvepser og snyltevepser.

T5-Large-SNL (R-1: 37.21) Årevinger er en orden av insekter i ordenen veps. De omfatter bier, humler, stilkveps, broddvepser og snyltevepser.

Borgerlig begravelse

Document: Arbeidet med å utvikle en ikke-religiøs gravferdsseremoni har i Norge særlig skjedd i regi av Human-Etisk Forbund, som brukte betegnelsene «borgerlig bisettelse» og «borgerlig begravelse» på sine gravferdsseremonier frem til 2006, da navnet ble endret til «humanistisk gravferd».Også borgerlig begravelse må foregå på en kirkegård eller i et krematorium. Ved kremasjon kan urnen settes i urnehall der man har dette, ellers må urne eller kiste settes ned på kirkegård. [...]

Reference: Borgerlig begravelse er en tradisjonell betegnelse på gravferd som foregår uten medvirkning av et religiøst trossamfunn. Før 1. januar 1970 var borgerlig begravelse forbeholdt personer som ikke var medlem av Den norske kirke, men i dag har alle adgang til å nytte en slik seremoniform.

T5-Base-SNL (R-1: 36.96) Borgerlig begravelse er en seremoni for personer som ikke er medlem av en kirke eller en folkekirke. I Norge brukes betegnelsen «borgerlig bisettelse» og «humanistisk begravelse» som en uformell samlebetegnelse for ikke-religiøse gravferdsseremonier i regi av Human-Etisk Forbund eller Humanistene (tidligere Humanistforbundet).

T5-Large-SNL (R-1: 37.21) Borgerlig begravelse er en ikke-religiøs gravferdsseremoni for personer som ikke er medlem av kirken eller den danske folkekirken.

APPENDIX F CNN/DAILYMAIL

F.1 Lowest ROUGE-1

Examples of two of the summaries which got the lowest ROUGE-1 scores on T5-Large-CNN/DM with their corresponding references

Document: (CNN) – Ca 160 mennesker ble arrestert av saudiarabiske sikkerhetsstyrker tidlig fredag morgen i Buraida for ulovlig protester utenfor den byens Bureau of Investigation and Aktoration, ifølge Saudi Press Agency. Talsmannen for Qassim-politiet sa at 15 kvinner og 6 barn var blant de 161 [...]

Reference: Amnesty International kaller arrestasjoner forfølgelse. Dusinvis protesterer i byen. Demonstranter er blitt opprørt over det rettslige system. Saudiske offisielle gjør referanse til avvikende grupper

T5-Base-SNL (R-1: 20.0) Ca 160 mennesker ble arrestert i Buraida for ulovlig protester utenfor byens Bureau of Investigation and Aktoration. Demonstranter har gjentatte ganger samlet seg for å kreve at de fengslede slektningene de sier har vært holdt fengslet i årevis uten å ha blitt tiltalt, prøvd eller gitt adgang til advokater. Amnesty International kritiserer arrestasjonene.

T5-Large-SNL (R-1: 2.56) En talsmann for Qassim-politiet sier at 15 kvinner og 6 barn var blant dem som ble tatt inn. Demonstranter ble arrestert etter at offiserer på stedet ikke hadde klart å overbevise dem om å avslutte demonstrasjonen. Demonstranter har gjentatte ganger samlet seg for å kreve at de fengslede slektningene deres blir løslatt.

Document: (Tribune Media Services) – Du kan dekke mye territorium i Skandinavia uten å noensinne sjekke inn på hotell. Overnattende luksuscruiseskip stablet med badstuer, smorgasboder og pliktfrie shoppingseilas nattlig mellom Stockholm og Helsingfors. [...]

Reference: Rick Steves skriver europeiske reiseguidebøker og vertskap for reiseshow. Hans TV-serie, Rick Steves' Europa, - om PBS-stasjoner. Steves selskap, Europe Through the Back Door, gjennomfører Europa-turneer.

T5-Base-SNL (R-1: 1.98) Båter mellom Stockholm og Helsingfors tilbyr gledene av et luksuscruiseskip til en rimelig pris. Viking og Silja forbinder hovedstedene i Sverige og Finland. Begge linjene tilbyr topp moderne skip med luksuriøse måltider, rimelige lugarer, rikelig med underholdning og nok pliktfrie godsaker til å senke et skip. Hver linje har en eldre, mindre luksuriøs flåte, men cateringer bedre til lavbudsjettreisende, tilbyr rabatter til studenter, seniorer og skinnepass-holdere.

T5-Large-SNL (R-1: 2.13) Båter mellom Stockholm og Helsingfors tilbyr gledene av et luksuscruiseskip til en rimelig pris. Båtene er store – på 56 000 tonn, nesten 200 meter lange – og med 2700 senger er de de største (og billigste) luksushotellene i Skandinavia. Viking har ry som festbåten. Silja regnes som mer elegant, men har likevel sin andel av tidvis irriterende og støyende passasjerer.

F.2 Highest ROUGE-1

Examples of two of the summaries which got the lowest ROUGE-1 scores on T5-Large-CNN/DM with their corresponding references

Document: (KNN Studentnytt) – 9. september 2011. Last ned PDF-kart relatert til dagens visning:. New York; Shanksville, Pennsylvania; Arlington, Virginia. Trykk her for å få tilgang til utskriften av dagens CNN Student News-program. Merk at det kan være en forsinkelse mellom tidspunktet da videoen er tilgjengelig og når transkripsjonen publiseres.

Reference: Daglig transkripsjon er en skriftlig versjon av hver dags CNN Student News-program. Bruk denne utskriften til å hjelpe elever med å forstå og forstå ordforrådet. Bruk ukeavisen til å teste din kunnskap om historier du så på CNN Student News.

T5-Base-SNL (R-1: 63.49) Denne utskriften er en skriftlig versjon av hver dags CNN Student News-program. Bruk Transkripsjonen til å hjelpe elever med leseforståelse og ordforråd.

T5-Large-SNL (R-1: 100.0) Daglig transkripsjon er en skriftlig versjon av hver dags CNN Student News-program. Bruk denne utskriften til å hjelpe elever med å forstå og forstå ordforrådet. Bruk ukeavisen til å teste din kunnskap om historier du så på CNN Student News.

Document: 9. mars 2015. Denne uken markerer årsdagen for flere historiske hendelser: en borgerrettsmarsj til Selma, Alabamas Edmund Pettus-bro, forsvinningen av en Malaysia Airlines-flyvning, og minnehøytiden for den internasjonale kvinnedagen. På denne siden finner du dagens program Transcript og et sted du kan be om å få være med på CNN Student News Roll Call. TRANSCRIPT. Trykk her for å få tilgang til utskriften av dagens CNN Student News-program.

Reference: Denne siden inkluderer serien Transcript. Bruk Transkripsjonen til å hjelpe elever med leseforståelse og ordforråd. Nederst på siden kommenterer man for en sjanse til å bli nevnt på CNN Student News. Du må være lærer eller elev i alderen 13 år eller eldre for å be om en omtale på CNN Student News Roll Call.

T5-Base-SNL (R-1: 73.58) Denne siden inkluderer serien Transcript og et sted du kan be om å bli nevnt på CNN Student News Roll Call. Bruk Transkripsjonen til å hjelpe elever med leseforståelse og ordforråd. Du må være lærer eller elev 13 år eller eldre for å be om at du blir nevnt.

T5-Large-SNL (R-1: 99.115) Denne siden inkluderer serien Transcript. Bruk Transkripsjonen til å hjelpe elever med leseforståelse og ordforråd. Nederst på siden kommenterer man for en sjanse til å bli nevnt på CNN Student News. Du må være lærer eller elev i alderen 13 år eller eldre for å be om en omtale på en CNN Student News Roll Call.

F.3 Random samples

Examples of some random samples from the test split of the CNN/DM dataset

Document: Kabul, Afghanistan (CNN) – Et NATO-ledet International Security Assistance Force helikopter krasjet fredag i Afghanistans hovedstad Kabul, og drepte 12 mennesker, sa ISAF i en uttalelse. ISAF bekreftet at helikopteretgikk ned men sa at årsaken til ulykken var ukjent. Det var ingen Anmeldelse om Opprørervirksomhet i Området,sa Kaptein. [...]

Reference: Tolv ISAF-medlemmer døde i krasjet, sier ISAF. Helikopteret styrtet i det østlige Kabul, sier en politisjef. De drepte er identifisert som tyrkiske borgere. Det er ingen rapporter om sivile tap, sier politisjefen.

T5-Base-SNL (R-1: 27.59) ISAF bekrefter at helikopteret gikk ned, men sier at årsaken til ulykken var ukjent. Det var ingen Anmeldelse om Opprørervirksomhet i Området.

T5-Large-SNL (R-1: 46.57) ISAF bekrefter at helikopteret gikk ned, men sier at årsaken til ulykken er ukjent. Kabul-politiet sier at fem lik ble trukket fra vraket og de ble identifisert som tyrkiske borgere. Det var ingen umiddelbare rapporter om sivile tap.

Document: En 18-årig mann er arrestert hjemme i Birmingham på mistanke om å forberede seg på å reise til Syria for å slutte seg til Den islamske staten. West Midland Police har sagt at tenåringen, fra Hodge Hill-området i Birmingham, ble arrestert rundt kl. 7 i dag av offiserer fra West Midlands Counter Terrorism Unit. En talsmann for styrken sa: _ "Operasjonen var planlagt og etterretningen ledet. Det var ingen umiddelbar trussel mot offentlig sikkerhet..."

Reference: West Midland Police sa at mannen er fra Hodge Hill i Birmingham. Han ble arrestert kl. 7 på grunn av mistanke om terrorlovbrudd, sier politiet. Det kommer etter at tre britiske tenåringsjihadister ble arrestert i Tyrkia. Sikkerhetsoperasjonen stoppet dem i å slutte seg til islamistiske statsforkjempere i Syria.

T5-Base-SNL (R-1: 42.59) En 18-årig mann er arrestert hjemme i Birmingham på mistanke om å forberede seg på å reise til Syria for å bli med i Den islamske staten. Han ble arrestert rundt kl. 7 i dag av offiserer fra West Midlands Counter Terrorism Unit. En talsmann for styrken sa: "Operasjonen var planlagt og etterretningen ledet. Det var ingen umiddelbar trussel mot offentlig sikkerhet."

T5-Large-SNL (R-1: 60.55) West Midland Police har sagt at tenåringen, fra Hodge Hill-området i Birmingham, ble arrestert rundt kl. 7 i dag av offiserer fra West Midlands Counter Terrorism Unit. Mannen ble arrestert på hjemmeadressen sin på mistanke om forberedelse til terrorhandlinger, i strid med terrorlovens § 5 2006. Den kommer dagen etter at tre britiske tenåringsjihadister ble arrestert i Tyrkia.

APPENDIX G OUT-OF-DOMAIN SAMPLES

Samples that were extracted from different sources than the different models were trained on. These documents don't have corresponding reference summaries.

G.1 Aftenposten

Mann siktet for drapsforsøk på Sunnaas sykehus

Document: Hendelsen skal ha skjedd ved 00.30-tiden natt til onsdag. En kvinne i 40-årene ble alvorlig skadet, men er utenfor livsfare, skriver VG. Hun ble operert onsdag. – Saken er kodet som drapsforsøk. Det er en alvorlig voldshendelse, bekrefter politiadvokat Torgeir Lutro til VG. Det er foreløpig uklart hva som foranlediget hendelsen.
Kvinnen var på besøk hos siktede, som var innlagt på sykehuset. Politiet har også beslaglagt et våpen de mener ble brukt. Det er ikke klart hva slags våpen dette er. VG skriver at brannvesenet kom til stedet før politiet, og at de passet på gjerningspersonen frem til politiet ankom. – Vi rykket fra brann, vi er de lokale. I dette tilfellet var politiet et stykke unna, forteller utrykningsleder Søren Rodriguez til avisen. [...]

T5-Base-SNL: Sunnaas sykehus er et åpent sykehus i Oslo. Sykehuset ligger på Nesodden utenfor Oslo. Det er Norges største spesialsykehus innen fysikalsk medisin.

T5-Large-SNL: Sunnaas sykehus er et sykehus i Oslo som er siktet for drapsforsøk etter en voldshendelse natt til onsdag.

T5-Base-CNN/DM: En kvinne i 40-årene ble alvorlig skadet, men er utenfor livsfare, skriver politiet. Kvinnen var på besøk hos siktede, som var innlagt på sykehuset. Politiet har også beslaglagt et våpen de mener ble brukt. Det er foreløpig uklart hva som foranlediget hendelsen.

T5-Large-CNN/DM: En kvinne i 40-årene ble alvorlig skadet, men er utenfor livsfare. Hun var på besøk hos siktede, som var innlagt på sykehuset. Politiet har beslaglagt et våpen de mener ble brukt.

G.2 Wikipedia

Levenshtein-distanse

Document: I informasjonsteori refererer Levenshtein-distansen mellom to strenger (f.eks. to ordformer) til det minste antallet operasjoner som trengs for å endre den ene strengen til en annen, hvor operasjonene er innsetting, sletting eller erstatning. Levenshtein-distansen har fått navnet sitt etter den russiske forskeren Vladimir Levenshtein, som satte opp mål for redigeringsdistanse i 1965. [...]

T5-Base-SNL: Vladimir Levenshtein-distanse er et mål for redigeringsdistanse mellom to strenger (f.eks. to ordformer).

T5-Large-SNL: Levenshtein-distanse er et mål for antall operasjoner som trengs for å endre en streng til en annen

T5-Base-CNN/DM: Levenshtein-distansen mellom to strenger refererer til det minste antallet operasjoner som trengs for å endre den ene strengen til en annen. Den brukes i retteprogram, men også i bioinformatikk.

T5-Large-CNN/DM: Levenshtein-distansen refererer til det minste antallet operasjoner som trengs for å endre den ene strengen til en annen. Den er nyttig for å finne ut hvor like to strenger er, og blir brukt mye i søkemotorer og søkefunksjoner.

G.3 Kvinneguiden

Jeg vil ikke delta i konfirmasjonen

Document: ”Jeg er stemor til to, og de har vært i livet mitt siden de var ganske små. Moren deres har vært vanskelig å ha med å gjøre hele tiden, selv om jeg ikke hadde noe med bruddet å gjøre. Som eksempler kan nevnes at hun bare har vandret inn i mitt hus når det passet henne, og hun har løyet om meg til barna. [...]

T5-Base-SNL: Dette er et spørsmål jeg ønsker å stille til foreldrene i forbindelse med bruddet i familien. Det er svært vanskelig å svare på spørsmålet, men det er vanvittig av meg å sette foten ned og ikke finner meg i mer psykisk mishandling. Jeg har bestemt meg for å dra til yngstes konfirmasjon.

T5-Large-SNL: Det er vanskelig å si noe om hvordan jeg har det i livet mitt. Jeg har vært i et forhold siden jeg var ganske liten, og har opplevd at jeg har blitt usynlig i mitt eget liv. Nå føler jeg meg svelget den siste kamelen, og føler meg i mer psykisk mishandling enn jeg noensinne har gjort.

T5-Base-CNN/DM: Mor til to har vært vanskelig å ha med å gjøre hele tiden, selv om jeg ikke hadde noe med bruddet å si. Hun har løyet om meg til barna, og hun har sagt at jeg ikke var usynlig. Jeg har bestemt meg, så trenger ingen råd, men er det så vanvittig av meg å sette foten ned og ikke finner meg i mer psykisk mishandling? Far er rasende på meg, og jeg føler at jeg har svelget den siste kamelen

T5-Large-CNN/DM: Mor har vært vanskelig å ha med å gjøre hele tiden, selv om jeg ikke hadde noe med bruddet å gjøre. Til tross for slike og lignende ting, stilte jeg opp for fullt i eldstes konfirmasjon. Jeg vil ikke dra i yngstes konfirmasjon. Far er rasende på meg, og jeg føler at jeg har svelget den siste kamelen.

G.4 Komplett.no

Skjermkort og grafikkort - GPU - Sørg for god grafikk

Document: Et godt grafikkort er viktig for å få det beste ut av spillopplevelsene! Hos Komplett finner du skjermkort for deg som skal game, jobbe med avansert design eller utføre enklere oppgaver.
En datamaskin er som mennesker flest – helt og holdent avhengig av flere deler i kroppen som spiller på lag. Prosessoren er datamaskinens hjerne og har ansvar for de fleste arbeidsoppgavene. Grafikkjobben derimot, det å vise levende bilder og filmer, er skjerm- og grafikkortets domene.
Enkelt fortalt avgjør grafikkort (også kalt skjermkort, GPU eller graphic card) hvor fort maskinen klarer å oppdatere skjermen når det gjelder oppløsning og bildehastighet. [...]

T5-Base-SNL: En datamaskin har et skjermkort som gjør det mulig å vise levende bilder og filmer på skjermen. Skjermkortet er en viktig komponent for å få det beste ut av spillopplevelsene på en datamaskin.

T5-Large-SNL: Et skjerm- og grafikkort er en komponent i en datamaskin som brukes til å vise levende bilder og filmer. Dette er den delen av maskinen som gjør de tyngste løftene. Det er skjermkortet som avgjør hvor fort maskinen klarer å oppdatere skjermen når det gjelder oppløsning og bildehastighet.

T5-Base-CNN/DM: En datamaskin er helt og holdent avhengig av flere deler i kroppen som spiller på lag. Grafikkjobben derimot, det å vise levende bilder og filmer, er skjerm- og grafikkortets domene. Produsentkongene Nvidia og AMD har sluppet noen av sine beste produkter på mange år. Det er en god idé å vurdere hvilket grafikkort du har behov for, for dette er den delen av maskinen som gjør de tyngste løftene.

T5-Large-CNN/DM: Det er fullt mulig å investere i et bra grafikkort til gaming uten å svi av en formue. Produsentkongene Nvidia og AMD har sluppet noen av sine beste produkter på mange år. For å få en jevn og god spillopplevelse i nyere spill er det nærmest et krav å ha et topp moderne grafikkort.

G.5 Human Evaluation Results

Results from the human evaluation performed on the out-of-domain samples.

Source	Model	INF	REL	FLU	COH	FAC	Overall
Aftenposten	T5-Base-CNN/DM	3.50	5.00	5.00	4.50	5.00	4.60
	T5-Large-CNN/DM	4.25	5.00	4.75	5.00	5.00	4.80
	T5-Base-SNL	2.75	3.75	4.50	5.00	3.75	3.95
	T5-Large-SNL	2.25	3.00	5.00	4.00	3.00	3.45
Wikipedia	T5-Base-CNN/DM	4.25	5.00	5.00	4.50	5.00	4.75
	T5-Large-CNN/DM	5.00	5.00	5.00	5.00	5.00	5.00
	T5-Base-SNL	3.50	4.75	5.00	–	4.25	4.38
	T5-Large-SNL	4.00	5.00	5.00	5.00	5.00	4.80
Kvinneguiden	T5-Base-CNN/DM	3.00	4.00	5.00	5.00	4.00	4.20
	T5-Large-CNN/DM	3.50	5.00	5.00	4.50	5.00	4.60
	T5-Base-SNL	1.00	1.00	4.50	3.50	2.00	2.40
	T5-Large-SNL	1.50	1.00	2.00	4.00	1.00	1.90
Komplett	T5-Base-CNN/DM	5.00	5.00	5.00	5.00	5.00	5.00
	T5-Large-CNN/DM	3.50	5.00	5.00	3.50	5.00	4.40
	T5-Base-SNL	4.00	5.00	5.00	5.00	4.50	4.70
	T5-Large-SNL	4.50	5.00	5.00	5.00	5.00	4.90

Table G.1: Results of human evaluation across 5 dimensions on the out-of-domain samples. Missing value was due to no coherence scored, due to only having one sentence in the generated summary.