IT UNIVERSITY OF COPENHAGEN

Automatic Text Summarization For Danish Using BERT

August 31, 2020

Lukas Christian Nielsen luni@itu.dk
Sebastian Lindegaard Veile seve@itu.dk

Master's thesis - Software Design
IT University of Copenhagen

Supervisor: Leon Derczynski
Course code: KISPECI1SE

Abstract

Information overload is a prevailing feature in modern society. Automatic text summarization is a useful tool to address this by enabling the reduction of longer texts down to their most important content. For English and other larger languages, automatic summarization is a relatively well studied field with a plethora of different summarization models. For lower resource languages like Danish, this is not the case. In this thesis, we explore the use of the popular BERT architecture for improving research in automatic summarization for Danish. Furthermore, we introduce TSAuBERT (Textual Similarity Assessment using BERT), a BERT based evaluation metric used for assessing the quality of automatically generated summaries, attempting to address the pitfalls of the current summarization evaluation protocol. We fine-tune 27 different BERT based summarization models using six Danish datasets, two different BERT models, and three summarization strategies. Our results provide new state of the art scores for each Danish datasets and solid scores for future research to improve upon. We further find that the existing Danish BERT model struggle compared to the multilingual BERT model when fine-tuned for a language generation task. Our analysis of TSAuBERT shows that it generally outperforms the most commonly used summarization evaluation metrics in terms of correlation with human assessments. However, the general correlation scores are not significantly large, thus leaving room for further improvements on the summarization evaluation protocol in general.

Contents

Acknowledgements

We want thank our supervisor Leon Derczynski for the guidance, support, and interesting discussions throughout the work on this thesis. We are glad we had the chance to have you as our supervisor, and we are thankful for all the good discussions we have had on both NLP and other topics.

We would also like to thank Daniel Varab and Natalie Schluter for creating a Danish dataset for summarization. Without this contribution, we could not have achieved the results we have in this thesis.

1. Introduction

1.1 Motivation

We live in a time of information overload. The connection of the world through the internet has led to a massive increase in the amount of information available to us from news articles to blog-posts. Much of this information is presented in natural language. With the sheer amount of continuous news-updates, it has become challenging for any individual to keep up. Staying well informed in such a time proves to be a challenging task. A possible solution for this is the development of software that can distill larger pieces of text down to its central elements. **Automatic text summarization** is a sub-field within Natural Language Processing (NLP) centered on developing software systems dedicated to generating short textual summaries of longer texts, that present the key elements of the text in a short and concise format. The literature describes two different summarization strategies; *extractive* and *abstractive*. Extractive summarization systems generate summaries by identifying a subset of sentences from the source text containing the most essential information and extracting them to form the summary. Abstractive summarization systems create summaries by automatically generating each word in the summary, and is capable of generating sentences and phrases not contained in the source text.

Automatic summarization is a well-studied field in NLP, with a rich history of approaches spanning from early feature engineering [Edmundson, 1969, Luhn, 1958, Lin and Hovy, 2000] to the use of modern deep neural networks [Nallapati et al., 2016, Liu and Lapata, 2019, Lewis et al., 2019, Zhang et al., 2019a]. Recent advances in deep neural network architectures, in particular the transformer [Vaswani et al., 2017] and the introduction of pre-trained language models such as BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers) [Devlin et al., 2019], has led a substantial improvement in state of the art results for a plethora of NLP tasks [Devlin et al., 2019], including summarization [Liu and Lapata, 2019].

However, progress in developing and applying these types of models on low resource languages is slow. For Danish, this is no different. NLP research for a language like Danish is challenging, as there are fewer resources, data, and researchers available [Kirkedal et al., 2019]. Research in automatic summarization for Danish remains an unexplored field, with only a recent publication of a dataset and some baseline summarization models [Varab and Schluter, 2020]. This thesis aims to address this gap in the literature, by developing a series of modern automatic summarization models for Danish, using the BERT architecture.

Automatic summarization is generally considered a challenging task in the NLP community. This is partly due to the challenging nature of automatically generating fluent and coherent language for abstractive summarization and part due to the subjective nature of what constitutes a good summary. Evaluating the quality of au-

tomatically generated summaries is a complex task. There are multiple dimensions to language, such as fluency, coherence, and grammaticality. Furthermore, there is no single objective definition as to what constitutes an optimal summary of a source text [Lloret et al., 2017]. The current evaluation metrics, the ROUGE metrics, used for evaluating summarization systems, rely solely on measuring the lexical overlap between automatically generated summaries and reference summaries accompanying the source text. However, this is inherently limited as lexical similarity is no guarantee for actual semantic similarity between two sequences of text. There is a need for a summarization evaluation metric that is able to measure textual similarity on a level deeper than the lexical. In this thesis, we develop such an evaluation metric, utilizing the contextually dependent word embeddings derived from pre-trained BERT models.

1.2 Research problem

The aim of this thesis is two-fold; first we seek to develop and evaluate a series of Danish summarization models using pre-trained BERT models. The aim here is to provide an insight into how we can use available BERT models for summarization of a low resource language and how different pre-trained models perform on this task. Secondly, we aim to develop an automatic summarization evaluation metric that is capable of measuring the semantic similarity of two sequences of text, using the contextually dependent word embeddings produced by BERT models. The aim is to improve the current protocol for evaluating the quality of summarization systems. The research is guided by the following research questions:

- How do automatic summarization systems based on pre-trained BERT models perform on Danish summarization?
- What are the performance differences between using a monolingual BERT model, pre-trained only on Danish data, and a multilingual BERT model pre-trained on 104 different languages?
- How well does a summarization evaluation metric based on measuring semantic textual similarities perform compared to the prevailing standard ROUGE metrics?

2. Background

2.1 Natural Language Processing

Natural language processing (NLP) is a field within artificial intelligence (AI) where software is developed with the goal of achieving human-like language understanding for a wide array of tasks [Deng and Liu, 2018, Liddy, 2001]. The variety of tasks within NLP span a broad range, from Named Entity recognition, Machine Translation, Speech recognition, and Automatic Text Summarization.

In the last decade, research in NLP has experienced a shift, going from statistical and feature-engineering models to the widespread use of machine learning and deep neural architectures [Deng and Liu, 2018]. The emergence of deep neural networks such as the RNN, LSTM, and most recently, the transformer, have allowed for the development of models and algorithms that capture a deeper understanding of natural language for many NLP tasks, pushing the state-of-the-art to new levels [Deng and Liu, 2018]. In recent years, there has been a paradigm shift with the introduction of the transformer (section 2.5.4) and pre-trained models like BERT (section 2.6) achieving state of the art on a broad range of NLP tasks [Vaswani et al., 2017, Devlin et al., 2019]. These model architectures have formed the basis through which research in complex NLP tasks like Automatic Text Summarization have achieved a new state of the art [Liu and Lapata, 2019, Zhang et al., 2019a, Yan et al., 2020, Lewis et al., 2019].

2.2 Automatic Text Summarization

Automatic summarization is a sub-field within NLP focused on the development of software-systems capable of generating short textual summaries containing the central information from a given source text. Summarization systems are generally classified across three dimensions; **input**, **output**, and **purpose**.

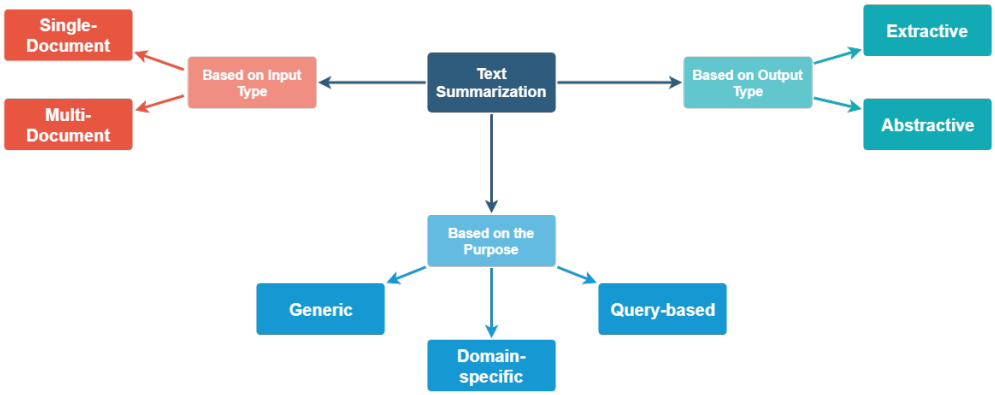


Figure 2.1: Types of text summarization from [Chauhan, 2019]

2.2.1 Input

Input refers to the document type of the source text(s) that summaries are produced from. The two main summarization input types are single- and multi-document. **Single-document** summarization involves identifying the central information from a document and generating a summary of that, while **multi-document** summarization involves identifying the topic in a set of multiple documents and generating summaries thereof [Nenkova and McKeown, 2011]. For efficient multi-document summarization, the set of documents summarized needs to concern the same topic or event.

2.2.2 Output

The output of a summarization system refers to the types of summaries produced. The two main summary types are Extractive and Abstractive. The goal in **extractive** summarization is to identify and extract a subset of the most important sentences in the source text and combine these to a single summary, without any modification to the original wording [Nenkova and McKeown, 2011]. Extractive summarization is often framed as a binary classification task, where each sentence in a source text has to be classified as being part of the summary or not. A central benefit of extractive summarization is that it produces grammatically coherent summaries, as it is simply copying full sentences from the source text. The downside of this is that it cannot in any way alter these, and thus risk extracting sequences of extraneous information.

In **Abstractive** summarization, the summaries are produced by generating new sentences based on the content in the source text. Unlike the extractive approach, abstractive summarization is not restricted to only using words that occur in the original document [Nenkova and McKeown, 2011, Khan et al., 2016]. Abstractive summarization is akin to a natural language generation (NLG) task, where the goal is to identify/predict the next word in a sequence based on input and previously generated words, thereby auto-regressively producing summaries.

Abstractive summaries often have the advantage of being shorter and more concise than extractive summaries since they can remove/ignore redundant or "filler" words from the source text. While having the advantage of producing more compressed summaries, abstractive summarization is generally considered a more complicated task, as generating longer sequences of text that remain grammatically and semantically coherent requires a more complex understanding of natural language and often a more sophisticated summarization model. See figure 2.2 below for an example of an extractive and abstractive summary. (Note these summaries have been generated by summarization models developed in this project). Throughout this thesis, we refer to the output types of summarization systems as the **summarization strategy** of the system.

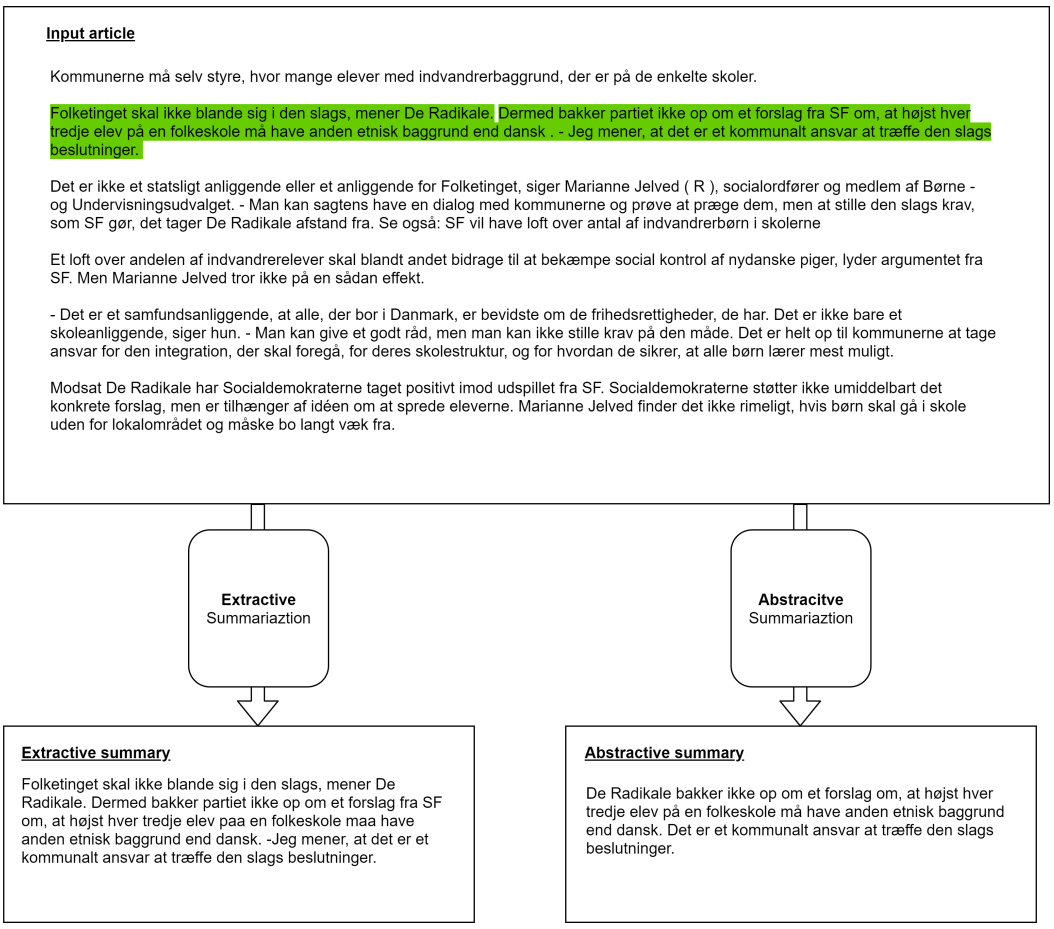


Figure 2.2: Summarization strategies examples - (green indicates sentences extracted)

2.2.3 Purpose

There are three main distinctions in regards to what purpose a summarization system tries to fulfill, domain-specific, generic, or query-based.

A **domain-specific** summarization system is often specialized in producing summaries for a specific domain. There are multiple different domains in summarization like news-articles, scientific articles, books, emails, etc. All these come with their own unique features and challenges. For example, news articles often carry the most important information at the beginning of the document [Kedzie et al., 2018, Liu and Lapata, 2019]. This is referred to as the *layout bias* in news summarization [Kryscinski et al., 2019]. These domain-specific pieces of information can be advantageous to exploit when developing domain-specific summarization systems.

A **generic summarization** system is able to produce summaries while making no assumptions of the domain or genre. The summarization system is not tuned to

any specific decision based on knowledge from the domain [Nenkova and McKeown, 2011].

Query-based summarization systems base their summaries on an initial input, i.e., a query, as well as the original document. This query is the topic of which the summarization system generates its summaries. The intuition behind this is that human-generated summaries are often written with some goal/topic in mind [Nenkova and McKeown, 2011].

The summarization models developed for this project produce single-document summaries domain-specific to Danish news articles, using both extractive, abstractive, and a mix of the two as the summarization strategies.

An important note on naming conventions; we refer to a summary produced by an automatic summarization model as a **candidate summary** and refer to the gold-standard summary accompanying each source text as the **reference summary**.

2.3 Evaluation of summarization systems

Measuring the quality of automatically generated summaries is a critical step in developing summarization systems. However, evaluating the quality of generated language and automatically generated summaries is a challenging task [Lloret et al., 2017]. The challenging nature of the evaluation task stems partly from the multiplicity of dimensions through which the quality of natural language can be assessed and partly from the multiple approaches towards automatic summarization [Lloret et al., 2017]. Furthermore, what constitutes a good summary of a given source text is somewhat subjective. There is no single ground truth for what an optimal summary looks like. Thus evaluation metrics for summarization must rely on other information than objective ground truth when assessing the quality of automatically generated summaries.

The quality of a candidate summary is most commonly assessed against a reference summary that accompanies the source text and is written by a human. The quality and contents of the reference summary have a significant impact on how a candidate summary is scored. Furthermore, what constitutes a 'high quality' summary depends largely on which dimension of language that is being assessed, i.e., whether it is the *linguistic quality* of the summary language, its *fluency*, *content overlap* with the reference summary or some combination of these.

As a result, the development of good summarization evaluation metrics remains an open problem in NLP. That being said, there are several existing frameworks for evaluating the quality of automatic summarization systems currently used in the field. The following sections describe the primary **manual** and **automatic** evaluation metrics.

2.3.1 Manual Evaluation

Manually evaluating candidate summaries with human assessors is generally considered a reliable indicator of the quality of a summarization system [Sellam et al., 2020, Lloret et al., 2017]. The most commonly used human evaluation method is the Pyramid framework.

2.3.1.1 Pyramid

The **Pyramid** method is a framework for evaluating automatically generated summaries using human assessment. The method relies on having multiple human-written reference summaries available during evaluation. This method aims not to analyze the linguistic quality in each candidate summary, but instead their content selection. Human assessors are used to identify summarization content units (SCUs) by grouping information that describe the same content in the reference summaries [Nenkova et al., 2007].

A SCU is defined as a collection of text sequences that concern the same topic and/or have the same meaning [Hennig et al., 2010]. A SCU is created by a human annotator, by firstly identifying sentences describing the same or similar information in a set of reference summaries. After the annotator has clustered semantically similar sentences together, the sentences are further broken down into more related sub-units [Nenkova et al., 2007]. Each of these sub-units belongs to one SCU. Thus, a SCU is simply a collection of related sub-sequences of text pooled together by a human annotator.

Each SCU is given a weight based on how many reference summaries describe it. Those same SCUs are then applied to the candidate summaries. The score of a candidate summary is calculated by the sum of its SCU weights, which are held up against an optimal summary with the same amount of SCUs [Nenkova et al., 2007]. A high score means the SCUs of the candidate summaries were of a higher weight, indicating that the candidate summary was able to capture the most important content of the source text. Using multiple reference summaries written by different individuals, this method is able to adjust some of the instability that occurs in human evaluation [Nenkova et al., 2007].

2.3.1.2 Readability

Evaluating the **readability** of a candidate summary is concerned with assessing the quality of the language. This is typically done by asking human assessors to judge each candidate summary on five dimensions [Lloret et al., 2017]; *grammaticality* are there grammatical errors and mistakes, *Non-redundancy* whether there are unnecessary repeated information, *referential clarity* it should be clear what entity the nouns and pronouns belong to, *focus* the summary should make sense as a whole, and *Structure* the coherence of the summary [Lloret et al., 2017]. Each of these dimensions is typically evaluated on a 1-5 scale, 1 indicating low quality, and averaged

to form the summary's readability score.

Although the manual assessment techniques described above are considered good indicators of summary quality, there are significant challenges in using these in the development of automatic summarization systems. Firstly, human evaluation is a rather slow and expensive process that does not scale well. Furthermore, using human assessors to evaluate summaries also brings its own instabilities. Some research shows that there is often a low agreement between human assessors regarding which sentences can be considered good summary sentences. This means that human evaluation results can be difficult to reproduce and dependent largely on the person that does the evaluation [Inderjeet, 2009, Lin and Hovy, 2002].

As a result, the field of automatic summarization has relied on **automatic** evaluation metrics as an indicator of the quality of a system. Automatic evaluation algorithms typically evaluate the degree of similarity between a candidate and reference summary producing a score between [0-1]. The benefit of automatic evaluation is that it is considerably faster than human assessment. Hence it scales better for modern datasets that include thousands of instances in their evaluation dataset. Furthermore, results are easier to reproduce since the automatic evaluation algorithm will always give the same answer to the same candidate and reference summary pair. However, there are some challenges to the current protocol for evaluating the quality of automatically generated summaries [Peyrard, 2019], a concern we explain further in section 2.3.2.2.

2.3.2 Automatic Evaluation

Automatic evaluation metrics remain the most widely used methods for evaluating the quality of automatic summarization systems. Currently, the ROUGE family of metrics remain the predominating evaluation metrics used for summary evaluation. This section explains the central ROUGE metrics and how they are used. The following section provides an overview of some of ROUGEs shortcomings.

2.3.2.1 ROUGE Metrics

Recall-Oriented Understudy for Gisting Evaluation or **ROUGE** refers to a family of metrics developed for assessing the quality of automatic summarization systems [Lin, 2004]. There are several types of ROUGE metrics, but common for all is that they measure the quality of a candidate summary using **n-gram overlap** wrt. the reference summary. Introduced in 2004, the ROUGE metrics remain today the metrics defining state of the art for summarization systems. The ROUGE metrics remain a popular choice for evaluating summaries as they are conceptually simple, relatively fast to compute, and language-ambiguous. Furthermore, their widespread use in the field makes it easy to compare new summarization models to previous ones, establishing a trajectory of improvement. There are several variants of ROUGE. The two most widely used are **ROUGE-N** and **ROUGE-L**. We explain each below.

ROUGE-N measures the quality of a candidate summary, by the *count* of n-gram overlaps between the candidate and reference summary. n denotes the length of the n-gram. For $n = 1$ it is the word overlap, for $n = 2$ it is the bi-gram overlap, and so forth. It uses a recall, precision, and F1-score to calculate the final score for the candidate summary. Using F1-score, the metric is able to account for the variable lengths of the candidate and reference summaries. The final score is between [0-1]. Given a candidate summary C of length n and a reference summary R of length m, with $gram_n$ denoting the number of overlapping n-grams between C and R, the ROUGE-N score is defined as:

$$\textbf{ROUGE-N}_{recall} = \frac{gram_n}{|ngrams \in R|}$$

$$\textbf{ROUGE-N}_{precision} = \frac{gram_n}{|ngrams \in C|}$$

$$\textbf{ROUGE-N}_{F1} = 2 \cdot \left(\frac{recall \cdot precision}{recall + precision} \right)$$

Typically, summarization systems are evaluated using the average ROUGE-N F1-score with $n = 1$ and $n = 2$. These are referred to as ROUGE-1 and ROUGE-2.

ROUGE-L uses the longest common subsequence (LCS) between sentence-pairs of the candidate and reference summary as the basis of generating a similarity score. LCS is defined as the longest, strictly increasing, sub-sequence of words between two sentences [Lin, 2004]. Given a sentence A containing the words $[w_1, w_2, w_3, w_4, w_5]$ in that order and a sentence B containing the words $[w_1, w_3, w_2, w_5, w_4]$ in that order, $LCS(A, B) = \{w_1, w_3, w_5\} = 3$.

ROUGE-L uses the sum of the *union-LCS* between a reference summary sentence r_i and every candidate summary sentence c_j [Lin, 2004]. Similar to ROUGE-N, the final score is expressed as an F1-score. The final score is between [0-1]. Given a candidate summary C consisting of v sentences with a total length of n, and a reference summary R consisting of u sentences with a total length of m, the ROUGE-L score is defined as [Lin, 2004]:

$$\textbf{ROUGE-L}_{recall} = \frac{\sum_{i}^{u} LCS_{\cup}(r_i, C)}{m}$$

$$\textbf{ROUGE-L}_{precision} = \frac{\sum_{i}^{u} LCS_{\cup}(r_i, C)}{n}$$

$$\textbf{ROUGE-L}_{F1} = 2 \cdot \left(\frac{recall \cdot precision}{recall + precision} \right)$$

The advantage of using ROUGE-L is that it does not rely on consecutive matches of n-grams but rather measures the level of similarity of sentence order between the candidate and reference summaries.

See figure 2.3 on the following page for a step-by-step example of a ROUGE calculation. Note that "stemming" refers to a process of reducing each word to its root form, so that words like *'folkeskolerne'* and *'folkeskole'* are reduced to a common root form; *'folkeskol'*. The available tools for stemming in Danish are not perfect. However, they are significantly preferable to using multilingual stemmers, or not stemming at all.

Together, the above ROUGE metrics are reported when defining the new state of the art in automatic summarization. Using both variants of ROUGE, the evaluation of the system's performance is more detailed as it assesses both the total level of lexical similarity (ROUGE-N) and the degree of similar sentence structure (ROUGE-L).

2.3.2.2 Challenges with summarization evaluation

The evaluation metrics used to measure progress within a field of research play a critical role in defining the quality of new approaches and guiding the development of new research. For the field of automatic summarization, it is no different. In fact, one of the challenges in developing better summarization systems relates to the current protocol for evaluating the quality of automatically generated summaries [Peyrard, 2019]. This challenge stems from the fact that the central metrics used for evaluating summarization systems, the ROUGE metrics described above, have some shortcomings in their ability to accurately assess the similarities between two sequences of text.

Previous research has pointed out several issues with the ROUGE metrics. One such point of critique is that they have been found to have weak correlations with human evaluations [Kryscinski et al., 2019, Liu and Liu, 2008], which is usually employed as a measure of how well an automatic evaluation metric encapsulates the domain it is applied in [Peyrard, 2019].

Additionally, because ROUGE metrics use lexical n-gram overlap to measure similarity, they are unable to encapsulate synonymous concepts [Ganesan, 2015]. This

lexical focus of the ROUGE metrics further entails that they are unable to account for situations where two sequences of text are *lexically different* but *semantically similar*, or vice versa. Consider the following sentences:

a) *We went to the cinema*
b) *We have gone to see a movie*
c) *We went to the bank*

Sentences (a) and (b) are lexically different but semantically identical, while (a) and (c) are lexically similar but semantically different. The classical ROUGE metrics are unable to properly assess the similarities between either of the two sentence pairs. For example, $ROUGE - N(a, c) > ROUGE - N(a, b)$. In the context of automatic summary evaluation, if sentence (a) was present in a reference summary and sentence (b) was written to a candidate summary, ROUGE metrics would penalize this situation as a result of the low lexical similarity.

Furthermore, ROUGE metrics are unable to distinguish between different contextual meanings of the same word, e.g., the word "party" can refer to both a festive activity as well as a political organization, depending on the context in which it is used. This point is of particular relevance in the light of the advancements in neural language models, such as those based on the BERT (see section 2.6) architecture, that are able to produce contextually dependent representations of natural language.

The evaluation metric developed in this project (see section 5.2) utilize these contextually dependent embeddings to assess textual similarity on a deeper semantic level.

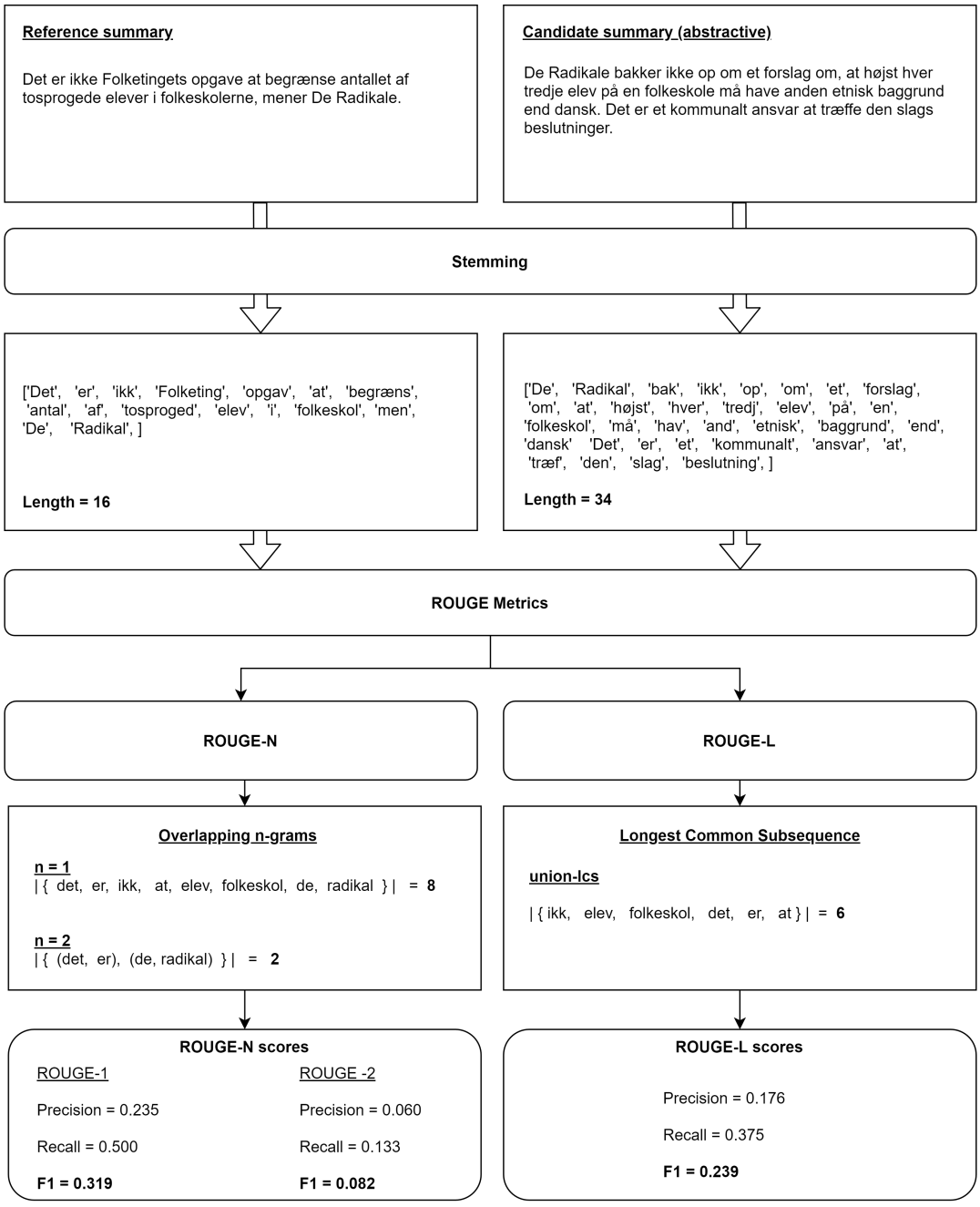


Figure 2.3: ROUGE scores example

2.4 Representing words as vectors

This section describes the concept of **word embeddings** and its importance in NLP. The concept is of particular relevance for this project, as we use a specific type of embedding as the basis of the evaluation metric we have developed.

Computing word embeddings refer to the process of converting a word to a continuous vector representation. These embedding vectors are an integral part of NLP as it provides a representation of words processable by computers. The core concept is to project words to vectors in a learned vector-space, where semantically similar words have embedding vectors in relatively close proximity to each other. Several techniques for generating these embeddings have emerged in the past decade. We present here the most popular embedding/language models.

Word2Vec introduced by [Mikolov et al., 2013b] generates word embeddings using a Feed-Forward Neural network (see section 2.5.1) trained using unsupervised learning on a large corpus of text. The famous King-Queen example gives an indication of the usefulness of these word embeddings. Taking the embedding vectors for the words *King*, *Man* and *Woman* and doing the following computation:

$$['King'] - ['Man'] + ['Woman']$$

The output is another vector, which is closest to the $['Queen']$ vector. This showcases that the word embeddings are able to capture both syntactic and semantic similarities. [Mikolov et al., 2013a]. The conceptualization allows for the calculation of semantic similarity between words using common techniques from linear algebra such as dot product and cosine similarity on their embedding vectors.

Even though Word2vec provides reasonable representations of words, their embeddings are limited in their ability to capture many of the nuances in natural language. This is due to the fact that the embeddings produced by word2vec are **static**, i.e., the same word is mapped to the same embedding vector no matter the context in which the word appears. This is a problem since some words change meaning depending on which context they are used. As in the previous example in 2.3.2.2, the word *'party'* can refer to both a festive activity or a political organization, Numerous words in both Danish and English possess this characteristic.

Thus there is a need for techniques that enable the generation of **contextually dependent** word embeddings, which considers contextual information when generating the vector representation.

Pre-trained language models based on the **transformer** neural network architecture (see section 2.5.4) address this issue. In particular, the language models based on the **BERT** model architecture (see section 2.6) are defined by their ability to generate contextually dependent word embeddings that take into account all other words in the context as well as their relative position to each other [Devlin et al.,

2019]. They are able to do so using the attention mechanism (section 2.5.4.1) inherent in transformer-based models.

These techniques allow BERT models to generate embeddings that are rich with contextual information. We use these embeddings as the basis for our summarization evaluation metric (see section 5.2). We explain further how these embeddings are generated in section 2.6.1.

2.5 Neural Networks

This section describes the central neural network architectures used for NLP tasks and motivates the use of the architecture employed in this project; the **transformer**, and outline how it is used in the BERT models we implement for both the automatic summary generation and the evaluation tasks. The section first lays out the basic elements of neural networks. This is followed by a short overview of the Recurrent Neural Network (RNN) model architecture, describing its core functionality as well as its limitations. The section includes a description of the transformer architecture, explaining its core components, and concludes with an overview of the BERT architecture and how this uses the transformer model.

2.5.1 Artificial Neural Networks

Artificial neural networks form the basis of a diverse class of machine learning algorithms. The general aim is to train a network of connected units (often referred to as **neurons**), to recognize patterns in input data with the purpose of generating predictions based on these patterns. The neurons are organized in layers (input, hidden, and output) and are connected between these layers through connections. Each connection between two units has a weight, attached to it that determines the 'strength' of the connection [Russell and Norvig, 2009]. Neural networks differ in the number of neurons, number, and structure of layers and how information flows through the network [Russell and Norvig, 2009]. A basic version is the Multilayer Feed Forward Network (FFN) illustrated in figure 2.4 below.

Information in this type of network flows only in a forward direction from input to output with no recursion between layers. Each neuron in each layer applies two computations to its input; a weighted sum of the previous layer's connections and an activation function. The activation function is typically a non-linear function such as **sigmoid**, **tanh**, **ReLU**, etc and allows for the network to map non-linear relations in the data [Russell and Norvig, 2009].

Training a neural network refers to the process of deriving the set of weights and biases for the links in the network, which results in the most accurate predictions for a given task. Most networks are trained using supervised learning in which the network is provided with the input and the ground truth value of the desired out-

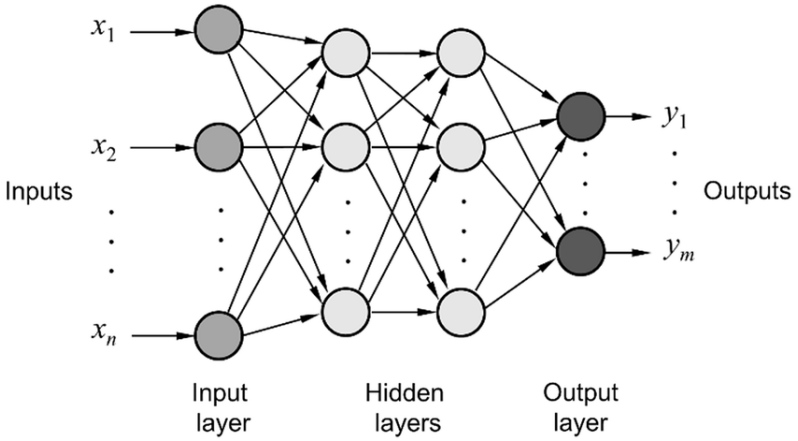


Figure 2.4: Multilayer FFN - Figure from [Pouliakis et al., 2016]

put. The input flows through the network producing the predicted output. A **loss function** is used to compute the loss of the network, i.e., how far off the networks' prediction was to the desired output value for the given input data [Aggarwal, 2018]. This is known as forward propagation. The choice of loss function depends largely on the nature of the learning problem at hand, i.e., whether it is binary classification, multi-class classification, sequence prediction, etc. [Aggarwal, 2018].

Based on the loss, appropriate adjustments can be made to the weights and biases of the network with the purpose of reducing the prediction loss in future iterations. This is usually done with an algorithm called **back-propagation**, which uses gradient descent to calculate the partial derivative with regards to all weights and biases in the network. This process is repeated until a satisfactory result has been reached, or the network cannot further improve with the current set of parameters [Aggarwal, 2018].

2.5.2 RNN

Recurrent Neural Networks (RNN's) are popular neural network architectures for dealing with sequential data like natural language. RNN's are structured as a series of FFN where information of the output of the network at a given time-step t^i is kept in a hidden state h_t and is passed as input in the following time-step t^{i+1}. The main concept is that the information contained in the hidden state allows the RNN to 'memorize' previous information of the sequence when making a prediction at a given time step. See figure 2.5 below for a visualization of an unfolded RNN architecture.

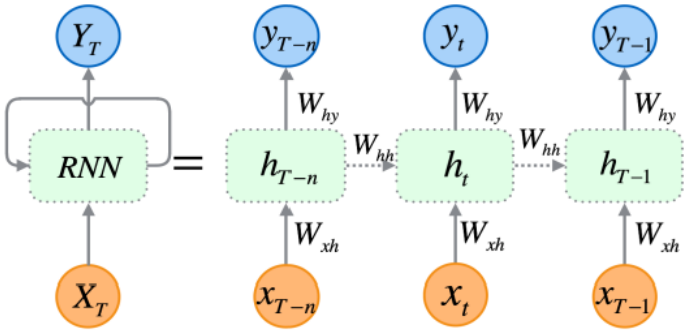


Figure 2.5: Unfolded RNN from [i2tutorials, 2019]

Although this mechanism allows the network to retain information from previous states, the RNN is limited in its ability to learn long-term dependencies in sequential data [Trinh et al., 2018] [Hochreiter and Schmidhuber, 1997]. This is due to the fact that the back-propagation algorithm variant used for calculating gradients in RNN's (backpropagation through time (BPTT)) suffer from problems of vanishing and exploding gradients [Hochreiter and Schmidhuber, 1997] which can lead to inaccurate adjustments of the network weights causing learning problems [Informatik et al., 2003].

This problem stems from a combination of factors, primarily network depth (number of recurrent units), and is amplified by the fact that the same weight parameter is applied for each of the recurrent units, as can be seen in the above figure. For NLP tasks, this can result in the model struggling to maintain the long-term dependencies inherent in textual data.

2.5.3 LSTM

The Long Short-Term Memory model architecture introduced by [Hochreiter and Schmidhuber, 1997] address the gradient-related issues of the original RNN by adding additional computational layers, known as gates, and an internal state or 'cell state' to the core RNN network model. The gates control the flow of information through each memory cell in the network, updating the cell state deciding which components to keep and which to 'forget' [Hochreiter and Schmidhuber, 1997]. See figure 2.6 below for an overview of an LSTM cell.

Through these gates, the network is able to decide on which parts of the previous hidden state and the input should be added to the updated cell state and which to forget, as well as which components of the updated cell state, input, and previous hidden state to carry into the next time-step [Lipton, 2015]. The internal cell state managed by the different gates allows for the stable calculations of gradients during back-propagation, effectively dealing with the vanishing/exploding gradients problem of the original RNN [Lipton, 2015].

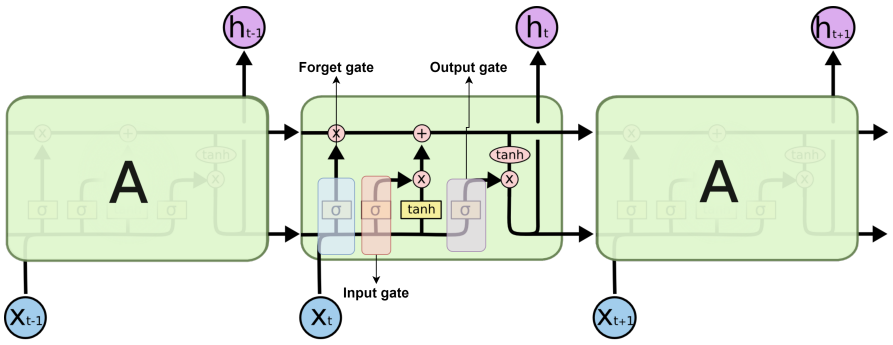


Figure 2.6: LSTM memory cell - modified from [Olah, 2015]

Although the LSTM provides an improved model over the original RNN, the architecture's recurrent nature remains a challenge in two central areas. Firstly LSTM networks still struggle with long-term dependencies in, e.g., text data, as signals from words at the start of the sequence have to flow through the entire network to have an effect on the words towards the end of the sequence. Secondly, RNN based networks are slow to train, as each computation is dependent on the previous cell, and thus cannot utilize the parallelization capabilities of modern GPU's [Vaswani et al., 2017].

The transformer architecture addresses these issues via the use of a self-attention mechanism and positional encodings. We lay out the central elements of the **transformer**, and provide an introduction on how this architecture can be used for language generation tasks such as abstractive summarization in the following section.

2.5.4 Transformers

The transformer model introduced by [Vaswani et al., 2017] follows an encoder-decoder architecture, a widely used model architecture for modelling sequence-to-sequence tasks such as machine translation and abstractive summarization. The encoder is tasked with mapping an input sequence of tokens to a series of embedding vectors. These vectors are then passed to the decoder, which generates a new text sequence based on the vectors. What distinguishes the encoder-decoder implementation of the transformer model compared to those based on RNN architectures, is how it use the concept of **self-attention** and **positional encodings** to generate rich contextually dependent representations of language without relying on a recurrent architecture [Vaswani et al., 2017].

The encoder and decoder consist of two distinct sub-layers of computation; a Multi-Head Attention layer and a Feed-Forward Network [Vaswani et al., 2017]. We explain each sub-layer in the following section, followed by an overview of the encoder-decoder architecture of the transformer and how it is used for language generation.

Before embarking on understanding the Multi-Headed attention layer, we lay out the core principles of the specific attention mechanism used in the transformer.

2.5.4.1 Self Attention

The self-attention mechanism is what enables the transformer models to generate deep contextually rich encodings for each word in an input sequence. It is the essential concept that allows transformers to derive long-term dependencies in textual data [Vaswani et al., 2017]. Figure 2.7 below provides a simple visual representation of a self-attention calculation for one word in an input sentence.

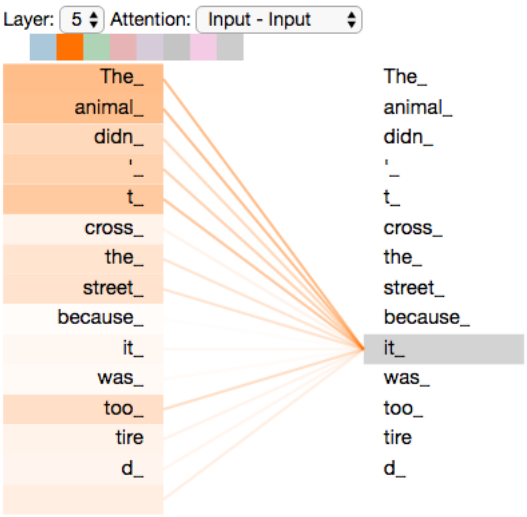


Figure 2.7: Self attention on input sentence, darker color indicates more attention is paid - from [Alammar, 2018b]

When generating the encoding for the word 'it', the attention mechanism places greater emphasis on the words 'the' and 'animal', which makes sense on a semantic level as 'it' refers to 'the animal' part of the input sequence. This shows how the network is able to capture long-term dependencies in the input data using attention.

Formally attention is defined as mapping a query vector q and a set of key-value k, v vectors to an output vector z [Vaswani et al., 2017]. The query, key and value vectors are all derived by multiplying the initial embedding vectors of each word in the input sequence with a set of associated query, key and value weight matrices W_Q, W_k, W_V, that are learned during the training of the network.

For each word w_i in the input, we derive an attention score for every word in the sequence (including a score for w_i itself). These scores indicate how much 'attention' should be paid to the different words when generating the embedding for w_i. Functionally, [Vaswani et al., 2017] generates three matrices one for the query, key, and value vectors for the input words to form three matrices Q, K, V, which allows for faster computations on modern GPU's leading to faster training times. The attention function can then be expressed as:

$$Attention(Q, K, V) = softmax\left(\frac{Q \cdot K}{\sqrt{d_k}}\right) \cdot V$$

The authors' label this attention variant '*scaled dot-product attention*' as we scale the dot product of the matrices with the square root d_k, where d_k is the dimensionality of Q and K. The intent of including this scaling factor is to allow for more stable gradients of the softmax function [Vaswani et al., 2017].

2.5.4.2 Multi-Headed Attention

Rather than employing a single self-attention computation, [Vaswani et al., 2017] uses a multi-headed attention layer where the the query, key and value matrices are split into h number of equal size chunks and fed to h number of different scaled dot-product attention heads ([Vaswani et al., 2017] use $h = 8$). These different attention-heads does not share weights and thus apply attention in different ways. The intent is that this generates a better representation as each head can attend to different embedding sub-spaces leading to an overall better contextual encoding of each word [Vaswani et al., 2017].

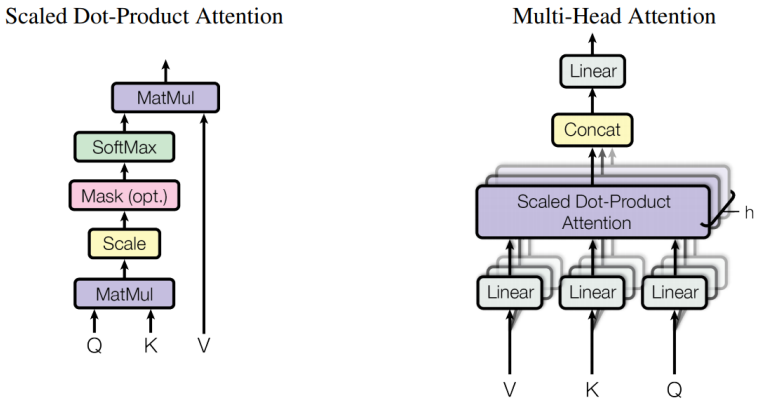


Figure 2.8: Scaled Dot-Product Attention and Multi-Headed Attention, from [Vaswani et al., 2017]

The output of the heads is concatenated and fed to a linear transformation layer before being fed to the next feed-forward network.

2.5.4.3 Feed-Forward Network

The feed-forward network (FFN) consists of two linear transformations with a ReLU activation function between them [Vaswani et al., 2017]. These are applied to every position in the input i.e., the vectors from the Multi-headed Attention layer. The

purpose of the FFN is to further process the output of the attention heads, encoding more information into the embeddings.

2.5.4.4 Encoder - Decoder

The transformer model's final structure is given in figure 2.9 below, with the encoder left and the decoder right.

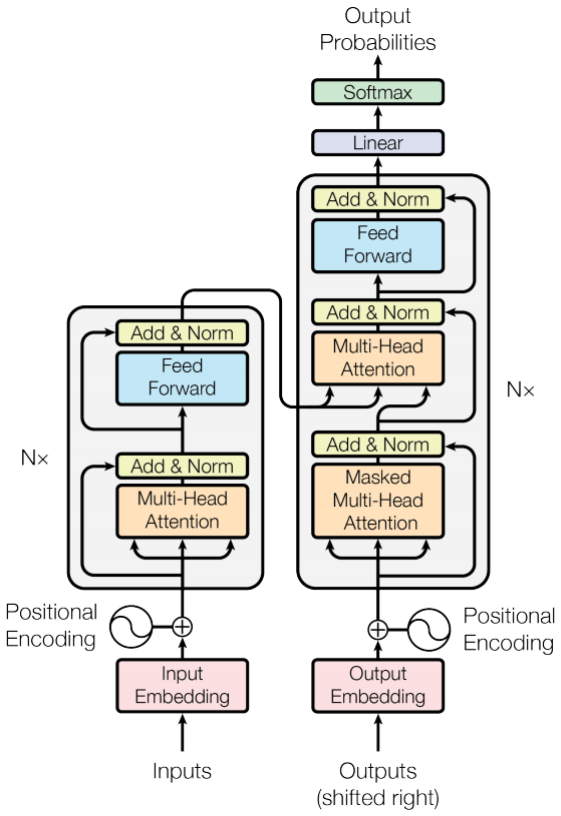


Figure 2.9: Transformer architecture - encoder left and decoder right - from [Vaswani et al., 2017]

Note there are two additional layers to the architecture; positional embeddings and normalization. The purpose of the positional encodings at the beginning of the encoder and decoder is to inject information on the relative position of the tokens in the input. This is important as it allows the network to incorporate the relative distance between tokens when generating the encodings [Vaswani et al., 2017]. The purpose of layer-normalization is to reduce training time.

The **encoder** consists of a stack of 6 identical encoding layers (however, they do not share weights). The output of each layer is fed as input to the next and the final output. With multiple encoding layers, the transformer is able to generate contextually rich word embeddings.

The **decoder** follows the same basic structure as the encoder, it is a stack of 6 identical layers, but each layer contains additional sub-layers of computation. A Multi-Head attention sub-layer that calculates attention on the encoder output is placed between the Multi-Head self-attention layer and the FFN [Vaswani et al., 2017]. This is implemented to allow the decoder to generate attention scores on the output of the encoder. Additionally, the decoder contains a masked Multi-Head Attention layer that is only applied to the output sequence of the decoder stack. The purpose is to ensure that word predictions take into account only the previously generated words [Vaswani et al., 2017].

Each of the 6 decoder layers receives the output of the encoder stack as input. These flow through the decoder network and decoder output are then fed through a linear layer and passed to a softmax function that produces word probabilities over the output vocabulary [Alammar, 2018b]. Selecting the word corresponding to the index of the softmax output with the highest probability the decoder is able to generate an output sequence of words.

The decoder auto-regressively generates each word in the output sequence, feeding the already predicted words into the bottom layer of the decoder stack at each step. Through this mechanism, the transformer architecture functions as a sequence to sequence model, able to generate variable length text sequences from input data.

The BERT architecture we introduce in the following section is an example of a model that uses the encoder component of the transformer to develop an understanding of the structures of natural language.

2.6 Bidirectional Encoder Representations from Transformers

Bidirectional Encoder Representations from Transformers or **BERT** is a pre-trained natural language model architecture that uses the transformer model structure [Devlin et al., 2019]. BERT was released in 2018, where the authors demonstrated the capabilities of the model by achieving state of the art results in 11 downstream NLP tasks with relatively little fine-tuning for each task [Devlin et al., 2019]. Since the release of BERT, a significant amount of research in the NLP community has been dedicated towards investigating the models' capabilities. In several instances, researchers have achieved new state of the art results for the respective task. Examples include Sentiment Analysis [Sun et al., 2019] and automatic summarization [Liu and Lapata, 2019]. These and several other such results have increased the popularity of using pre-trained BERT models immensely.

A significant appeal of the BERT model framework is that it provides a set of generalized pre-trained language models that can be fine-tuned to a variety of downstream tasks with relative ease [Devlin et al., 2019]. This section describes the architecture

of BERT, outlining the central mechanisms of pre-training and fine-tuning and provide an overview of the pre-trained models used in this thesis.

2.6.1 BERT Architecture

At their core, BERT models consist of layers of stacked transformer encoders. Three parameters define the 'size' of the model; the number of encoder layers L, number of attention heads in the Multi-Head attention sub-layer A, and number of connected units in the FFN sub-layer H (the latter is also referred to as the hidden size). Generally we distinguish between two sizes of BERT models; $BERT_{base}$ (L = 12, H = 768, A = 12) and $BERT_{large}$ (L = 24, H = 1024, A = 16). The figure below provides an overview of a $BERT_{base}$ architecture.

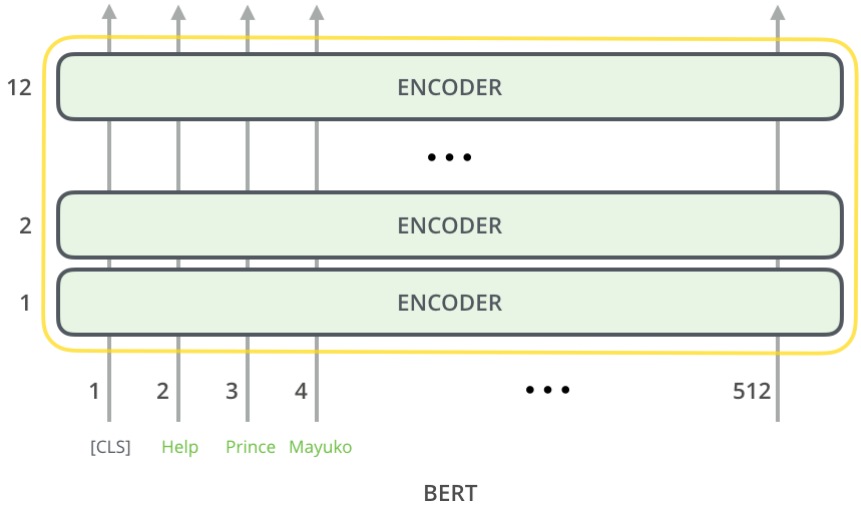


Figure 2.10: $BERT_{base}$ model from [Alammar, 2018a]

Data flows through BERT models as they would through a regular stack of transformer encoders, where the output of the last layer is a series of embedding vectors, one for each word/token in the input sequence.

BERT models process the **input** text by firstly tokenizing the input sequence using a **WordPiece** tokenizer [Wu et al., 2016] using a fixed size vocabulary (30,000 tokens for $BERT_{base}$ and $BERT_{large}$). Using this type of tokenizer, when the BERT models encounter Out of Vocabulary (OOV) words, the tokenizer splits them into several tokens known as word-pieces. For example the sentence: *'I went playing'*, assuming that the word *'playing'* is not in the vocabulary, it results in the following tokens: {'i', 'went', 'play', '##ing'}. The hashtags indicate that this is a continuation of a word.

BERT also adds two special tokens to the input; **[CLS]** and **[SEP]**. [CLS] token is prepended to the input sequence. The output vector for the [CLS] token is used as

a representation of the input sequence for classification tasks [Devlin et al., 2019]. The [SEP] token is used as a separator allowing BERT to distinguish between input sentence pairs, which is used in one of the pre-training tasks [Devlin et al., 2019].

To prepare the data for the encoder layers, BERT generates a vector for each token in the input by adding three different types of embeddings: token embeddings, segment embeddings, and positional embeddings. Token embeddings are the corresponding vector representation to that word in the vocabulary. Segment embeddings are used to identify what words are part of what sentence. Words that belong to sentence A are encoded E_A, and words that belong to sentence B are encoded E_B. Last is positional embeddings, which is the same encoding as for the transformer model, encoding information on each word's relative position in a sequence. The figure below provides an overview of how BERT processes input text before it traverses the encoder layers.

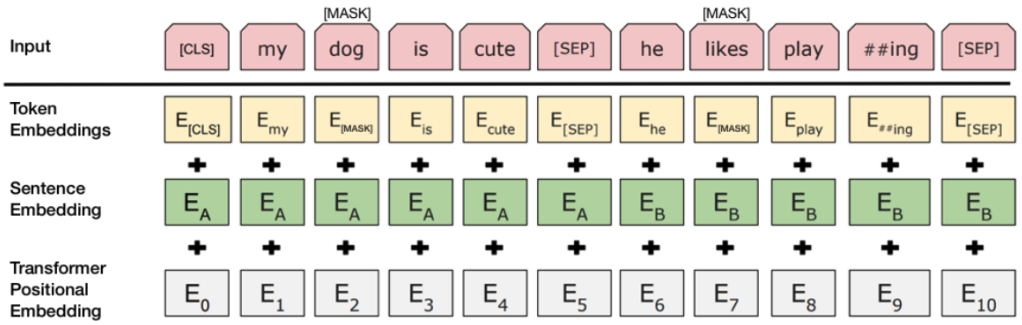


Figure 2.11: BERT input representation from [Devlin et al., 2019]

2.6.2 Pretraining & Fine Tuning

The BERT framework consists of two distinct tasks; **pre-training** and **fine-tuning** [Devlin et al., 2019]. The pre-training is intended for the model to develop a general understanding of language, while fine-tuning consists of tuning the parameters learned during pre-training to a specific downstream task [Devlin et al., 2019]. BERT models are pre-trained on two unsupervised tasks; Masked Language Modelling and Next Sentence Prediction [Devlin et al., 2019].

Masked Language Modelling (MLM) is an approach where random words in a sentence are masked/hidden. The model then feeds the output vector of the masked token through a softmax function over the vocabulary to predict the word [Devlin et al., 2019]. In the process of pre-training BERT, 15% of all tokens were masked. It is through this process that BERT achieves true bidirectionality [Devlin et al., 2019]. This is due to the fact that the model incorporates the token embeddings from both the left and right context of the masked token as the sequence parses through each encoder layer. This illustrates how the self-attention mechanism allows for the development of contextually dependent embedding vectors.

The second pre-training task, **Next sentence prediction** (NSP), happens simultaneously with the MLM approach. This approach differs from MLM by being a binary classification task of sentence pairs. For a given sentence pair (A, B) the model tries to predict for the sentence A whether B is the sentence that would follow A. 50% of the time, B is the next sentence. The rest of the time, a random sentence from the corpus has been inserted instead. Using this training approach BERT is able to learn the relationship between sentences [Devlin et al., 2019].

The pre-training of BERT allows the model to generate a unique understanding of language as compared to other neural language models. The word embeddings produced by a BERT model are unique in their versatility for a variety of downstream tasks. To fine-tune BERT, the model is first initialized with the parameters learned from the pre-training stage. These parameters are then fine-tuned to a specific supervised downstream task. The way fine-tuning takes place is task dependent [Devlin et al., 2019], as there are significant differences between a classification task, like extractive summarization, and a language generation task, like abstractive summarization. Part of the motivation for using BERT for summarization is that it provides a unified framework from which we can fine-tune both extractive and abstractive summarization models. We explain further the fine-tuning of BERT models for the summarization task in section 5.1.1.

2.6.3 Pre-trained models

There are a plethora of pre-trained BERT models currently released for general use [1]. Below is a list of the pre-trained BERT models used in this project. As this project uses BERT models for both the development of automatic summarization models for Danish and the development of an evaluation metric, we specify which of these tasks the BERT model is applied in.

- **Multilingual BERT** (mBERT), [Devlin, 2019] is a $BERT_{base}$ model pre-trained on 104 different languages, including Danish, with a vocabulary size of 119,547 tokens. The model is specifically trained on all Danish Wikipedia articles corresponding to approximately 77.493.876 words. This is relatively low compared to the approximate 3,647,590,434 English words. However, it is also trained on many similar languages to Danish, like Norwegian, Swedish, and German. mBERT is used for both the summarization task and evaluation metric task.
- **Danish BERT** (DaBERT) introduced by BotXO [BotXO, 2019] as part of the Nordic BERT framework, is a $BERT_{base}$ model pre-trained strictly on Danish text. [BotXO, 2019] states it has been trained on 1.6 billion words from various different datasets like CommonCrawl, Heste-nettet.dk, DinDebat.dk, and Danish Wikipedia. DaBERT is used for the summarization task.

[1] see https://huggingface.co/models?search=bert for a list of a subset of these models

- **BERT base** [Devlin et al., 2019] pre-trained on 3.3 billion English words with a vocabulary size of 30,000. Used for the evaluation metric task
- **BERT large** [Devlin et al., 2019] pre-trained on 3.3 billion English words with a vocabulary size of 30,000. Used for the evaluation metric task
- **sentence-BERT** [Reimers and Gurevych, 2019] is an instance of the `bert-base-nli-stsb-mean-tokens` pre-trained model from the paper [Reimers and Gurevych, 2019]. with a vocabulary size of 30,522. What distinguishes this model from other BERT models is that it is pre-trained on Natural Inference data and fine-tuned on Semantic Text Similarity data, with the specific purpose of producing sentence-embeddings useful for semantic similarity and related inference tasks [Reimers and Gurevych, 2019]. We include this model to investigate the differences in performance in using the basic pre-trained BERT models vs. models trained and tuned specifically for a task related to similarity assessment. Used for the evaluation metric task.

3. Related Work

In this section, we present key developments in automatic summarization, emphasizing recent approaches utilizing the transformer architecture described in section 2.5.4. We then introduce some recent attempts at implementing summarization evaluation metrics and outline potential reasons as to why they have been unable to gain traction in the field.

3.1 Developments in summarization

Extractive summarization systems have generally received the most attention in the field, primarily due to the fact that they are simpler to develop, computationally efficient, and produce summaries that are grammatical coherent [Zhong et al., 2019]. The abstractive summarization task introduces additional parameters like fluency and semantics, making the overall task more complex.

3.1.1 Extractive approaches

The task of extractive summarization (see section 2.2.2) has been addressed using a variety of approaches. The earlier efforts relied on developing unsupervised systems that would score and extract sentences based on features like term frequency, number of named entities (people, places, organizations), sentence position, and headline overlap [Edmundson, 1969, Luhn, 1958, Lin and Hovy, 2000]. These feature-based systems relied on assigning a score to each sentence in a source text and simply extract the highest scoring sentences. The core assumption of these systems is that by using these features, they are able to identify the sentences in the source text that contain the most relevant information.

Machine-learning based extractive summarization approaches have seen great popularity. Generally, machine learning approaches frame the extractive task as a series of binary classification problems, where each sentence in the source text has to be assigned a prediction as to whether it is a summary sentence or not. Earlier machine learning approaches used classical probabilistic Naïve-Bayesian classifiers [Kupiec et al., 1995, Wong et al., 2008]. These systems often relied on using sentence-level features such as term frequency, sentence length, and named entities to generate sentence representations.

The simplicity of these systems made them somewhat popular. However, they are inherently limited in their ability to consider more complex relations between sentences, and the entire source text. Using the Naive-Bayes classifier also means that all features are assumed to be independent of each other [Kupiec et al., 1995]. This is not the case for natural language, where the words in sentences are deeply interconnected. Thus other machine learning models like the Hidden-Markov [Conroy and O'leary, 2001], with fewer independence assumptions between features, have been

popular choices as well. These supervised systems often rely on the availability of labeled training data, which is often hard to come by [Allahyari et al., 2017].

Modern extractive systems are based on deep neural networks, using different variants of the RNN, LSTM, and, most recently, transformer architectures. [Nallapati et al., 2016] uses a bi-directional two-layered RNN model to classify sentences in the source text sequentially. For each sentence, their model takes into account the richness of information in the sentence, its importance in the source text, and the degree of new information it contains compared to the previously selected sentences for the summary [Nallapati et al., 2016]. [Zhou et al., 2018] uses a modified RNN architecture to jointly score sentences and extract the most salient ones, taking into account the information in the previously selected sentences. These defined state of the art until recently with the advance of transformer-based extractive models.

The use of the transformer architecture for extractive summarization is a relatively new avenue of research that has shown excellent results, improving ROUGE scores for several datasets. [Egonmwan and Chali, 2019] uses a standard transformer encoder and a softmax classifier to identify the most salient sentences. Improving on this architecture, [Zhang et al., 2019b] pre-trains a BERT model on a sentence-prediction task for better representation of sentences, and fine-tunes them on the extractive task using a softmax classifier. [Liu and Lapata, 2019] provides further improvements by fine-tuning a BERT model on the extractive task using additional transformer encoders and a sigmoid layer for the classification of sentences. The extractive summarization approach developed by [Liu and Lapata, 2019] is vital to this thesis, as we adopt their methodology when fine-tuning our extractive BERT based summarization models for Danish summarization. We outline the specific fine-tuning process in greater detail in section 5.1.2.

3.1.2 Abstractive approaches

Prior to the advance of modern encoder-decoder architectures, abstractive summarization methods relied mainly on simply compressing or merging extracted sentences [Gambhir and Gupta, 2016, Nenkova and McKeown, 2011]. Although these systems can perform relatively well, they are not purely abstractive, as they rely heavily on an extractive component in order to generate summaries. The advance of sequence-to-sequence architectures build on encoder-decoder setups have enabled the development of truly abstractive summarization systems.

Abstractive summarization using neural networks is phrased as a sequence-to-sequence problem. An input sequence (source text) is first encoded into a series of embedding vectors. A decoder is then used to decode the vectors and generate an output sequence; the candidate summary. These neural architectures often rely on the use of additional mechanisms such as attention [Chopra et al., 2016], and pointers [See et al., 2017] that allows the network to focus on some part of the input when generating the summary. Some also introduce copy mechanisms that enable the network

to copy words from the input document.

While most sequence-to-sequence abstractive summarization systems generated fluent summaries, they are sometimes unable to generate the right words and content. Sometimes a date or place would be wrong from the actual source, which meant that the summary would not make sense. This led to the development and implementation of copy mechanisms that were able to copy words from the source and thereby create more meaningful summaries that captured the content better[Gehrmann et al., 2018]. This led the focus away from purely abstractive summarization to an approach that relied on extractive content selectors. [See et al., 2017] uses a hybrid pointer generator network with attention, which can point to words in the input and copy them to be used in the summary. Besides the copying mechanism, they also introduce coverage, which lets the model see what words it has already attended to as a measure to prevent repeating the same words repeatedly in the summary[See et al., 2017]

[Gehrmann et al., 2018] experiments with multiple different extractive content selectors, one being a bidirectional LSTM. They also introduce the concept of bottom-up attention that helps the model decide on specific phrases. This technique beat state of the art on two different datasets [Gehrmann et al., 2018].

Abstractive summarization has, similar to extractive summarization, seen an advance in its state of the art with the release of the transformer architecture (see section 2.5.4). With the transformer and the later release of BERT, the focus and development of abstractive summarization systems shifted to a transfer learning approach by fine-tuning large pre-trained transformer-based models like BERT and BART [Lewis et al., 2019].

[Liu and Lapata, 2019] present an abstractive summarizer using a pre-trained BERT model as the encoder, and a 6 layered transformer for the decoder. [Liu and Lapata, 2019] achieved state-of-the-art on several datasets while presenting a conceptually clear approach. As for extractive summarization, we use the abstractive summarization approach introduced by [Liu and Lapata, 2019] when fine-tuning our Danish abstractive summarization models. We describe the specific fine-tuning setup for this in section 5.1.3.

The BART framework is based on pre-training a transformer-based encoder-decoder architecture for more robust language generation tasks [Lewis et al., 2019]. BART introduces a unique pre-training task where the model is given corrupted/noisy text and is tasked with reconstructing it back into the original. [Lewis et al., 2019] develop this strategy specifically for the improvement of encoder-decoder architectures for language generation. Their abstractive summarization approach uses a 6 layered pre-trained transformer model for the encoder and a 6 layered transformer decoder following the standard transformer setup by [Vaswani et al., 2017]. They fine-tune this pre-trained model on abstractive summarization achieving state of the art re-

sults on a series of datasets. Although the approach performs well on the abstractive task, using their approach for Danish summarization is challenging. Firstly, there is currently no pre-trained Danish BART model available. Secondly, the multilingual BART component has no Danish pre-training either. Thus if we were to use this framework for our Danish summarization models, we would have to pre-train a Danish model from scratch, which is costly in terms of computational resources and the data required.

The **PEGASUS** abstractive summarization model introduced by [Zhang et al., 2019a] relies on a transformer-based sequence-to-sequence approach. PEGASUS is a pre-trained model that has been trained on almost 4TB news-data using a new training objective called "gap sentence generation", in which a sentence is masked from the input. The model then tries to generate the original sentence using the other sentences from the input. PEGASUS was able to beat the previous state of the art on multiple news summarization datasets, by only fine-tuning on 1000 data samples [Zhang et al., 2019a]. While this approach achieves impressive results, it requires a massive amount of data and computational resources for pre-training. This makes it infeasible to use for this project as there is a limited amount of high-quality Danish data available, and we do not have access to the computation infrastructures needed to pre-train and fine-tune this model in a reasonable time.

In general, many of the current, and previous, state-of-the-art summarization models are only trained and fine-tuned on English datasets. This presents a challenge for developing summarization models for lower resource languages as pre-training these transformer-based models is costly and requires large amounts of quality data. Training these models from scratch for Danish would require an enormous corpus of danish text data and powerful/expensive hardware. Hence, why we have chosen to use the approaches developed by [Liu and Lapata, 2019], as it allows us to use available pre-trained BERT models with versions with varying degrees of Danish data used in pre-training. Furthermore, their approach is conceptually more approachable than the other transformer-based implementations, as it is based on fine-tuning already pre-trained BERT models. We explain the approaches developed by [Liu and Lapata, 2019] further in section 5.1.

3.1.3 Automatic Text Summarization for Danish

Currently, there is not much published research on Danish summarization. This can partly be attributed to the fact that prior to the introduction of the **DaNewsroom** dataset by [Varab and Schluter, 2020] earlier this year, there has been no Danish summarization dataset available. The production of these summarization datasets is both expensive and time-consuming. DaNewsroom is a large scale dataset of 1.13 Mill. Danish news articles generated using the approach of [Grusky et al., 2018]. Each article is associated with a reference summary categorized as either extractive, abstractive, or mixed. Together with the dataset, the authors have included ROUGE scores of two extractive summarization systems TextRank and ICSISum.

The DaNewsroom dataset and its accompanying baseline summarization scores currently constitute the published research on Danish summarization. With the release of DaNewsroom, the hope is that more research will be done on Danish summarization, as it constitutes an interesting case for low resource language summarization.

3.2 Recent evaluation metrics

As described in section 2.3.2.2, developing good evaluation metrics that accurately measure the quality of automatically generated language is a challenging task [Lloret et al., 2017]. For automatic summarization, the original ROUGE metrics have remained the standard within the field due to, among other things, their conceptual simplicity, the fact that they work for any language, and that they are well established benchmarks for summarization models' performance. However, given the issues pointed out in section 2.3.2.2, the ROUGE metrics, as standalone metrics are unable to fully encapsulate the complexity of evaluating the similarity between two sequences of text.

Some researchers have made attempts at developing metrics that address some of the core issues of the original ROUGE metrics. **ROUGE 2.0** introduced by [Ganesan, 2015] uses a synonym dictionary to consider possible synonyms of matched words to capture semantically related tokens [Ganesan, 2015]. Their approach is a step in the direction towards addressing the inability of ROUGE to capture semantically related words, developing a framework capable of using synonyms to strengthen the measuring of textual similarity. However, the reliance on a static dictionary limits the capability of ROUGE 2.0 to fully address semantic differences in candidate and reference summaries. Furthermore, it does not address the lexical bias of the original ROUGE metrics described in section 2.3.2.2.

The reliance on static dictionaries also makes it less flexible than the original ROUGE metrics, thus making evaluation for low-resource languages such as Danish challenging. Furthermore, the authors do not provide a study of the correlation between their metric and human evaluations. As such, it remains unclear whether this implementation provides an improvement over the original ROUGE metrics.

Another evaluation metric, **ROUGE-G** employs the use of graph-based random walk algorithms across a semantic network to compute the similarity between two summaries[ShafieiBavani et al., 2018]. Their approach is aimed at addressing the lexical bias of the original ROUGE metrics by using the lexico-semantic relations between words in the WordNet network to include the lexical similarity between sequences of text [ShafieiBavani et al., 2018]. Although they report good correlation scores with human evaluations, there are some caveats to their approach. Firstly, their model is conceptually quite complex and based on complex relations in a lexical database. Using a static database for computing semantic relations means that ROUGE-G suffers the same flexibility issues as ROUGE 2.0, hence it is difficult to apply it on low resource languages. Furthermore, as the authors have not provided

any code for their metric, nor the processing of the data, their results are difficult to reproduce, and their metric difficult to use.

ROUGE-WE uses the static word embeddings derived from the **word2vec** language model (see section 2.4, to assess whether two words in the candidate and reference summary are similar, as measured by the similarity of their embedding vectors [Ng and Abrecht, 2015]. Their key contribution is that of using the proximity of word embeddings in a vector space to assert whether two words are similar. Their conceptual model provides a clear progress towards a more sophisticated evaluation metric that, under the right circumstances, are able to account for semantic similarity rather than purely lexical. However, there are some caveats to their model. Firstly, given that they use the word-embeddings derived from a context-independent language model, they cannot account for the different contextual meanings of the same word. Using the word2vec model, the word *'party'* is assigned the same exact embedding regardless of the context of which it is used. Furthermore, although they provide a study of their metrics' correlation with human assessments, it is unclear how they have processed their data and arrived at the reported results[1].

Although these metrics certainly address some of the core issues with the original ROUGE metrics, neither has been adopted by the summarization community. Thus the original ROUGE metrics, with their flaws, remain the defining metrics used in the field [Kryscinski et al., 2019]. Hence, there is a need for the introduction of additional summarization metrics, that can strengthen the current protocol for summarization evaluation.

Following the experience with the recent metrics mentioned above, a new summarization metric should be conceptually somewhat simplistic, able to work for multiple languages and provide an additional level of analysis of textual similarity than ROUGE. The aim of developing new summarization evaluation metrics should not be to replace ROUGE, but rather add further detail to the evaluation protocol, working in tandem with ROUGE. Furthermore, the source code should be made readily available, so as the development process is transparent.

A recent evaluation system important to the summarization evaluation metric implemented in this thesis is **BERTscore** [Zhang* et al., 2020], which uses the *cosine similarity* between the token-level embedding vectors derived from BERT based transformer models, to assess the level of similarity between two sentences. Although this metric is not developed with summary evaluation in mind, the study presented by the authors demonstrates its usefulness as a measure of textual similarity for other language generation tasks such as machine translation and image captioning [Zhang* et al., 2020]. We use the core scoring concepts laid out by BERTscore in the implementation of our summarization evaluation metric, namely the use of cosine similarity between BERT embedding vectors to express the semantic similarity be-

[1]The source code provided by the authors have shown to have a critical error https://github.com/ng-j-p/rouge-we/issues/1

tween two words/tokens (see section 5.2). This approach has several advantages over other embedding vector-based scoring systems such as ROUGE-WE, most notably that the embedding vectors derived from BERT models are contextually dependent.

4. Datasets

In this section, we present the datasets that form the background for training the Danish summarization models and assessing the strength of `TSAuBERT` as a summarization evaluation metric. In total, we use three datasets, two used in the development and evaluation of the summarization models and one used for evaluating `TSAuBERT`.

4.1 DaNewsroom

The primary dataset used for the development and evaluations of the `PreSumm` summarization models is the **DaNewsroom** dataset, developed by [Varab and Schluter, 2020]. It consists of **1.13 million** article-summary pairs gathered from a URL database of 19 Danish news sites from the years 1997-2019. The reference summaries in DaNewsroom are extracted from meta-data tags *description*, on the website hosting the article [Varab and Schluter, 2020]. The content of these tags is typically short descriptions of the key contents of the news article, written by humans. The dataset further includes an assessment of the 'extractiveness' of each of the reference summaries wrt. to their article body, placing each summary on a scale of extractive-abstractive [Varab and Schluter, 2020]. This measure allows the dataset to be binned into three categories; **extractive**, **abstractive**, and **mixed** based on the level of n-gram overlap between a summary and its article body. See table 4.1 below for an overview and figure 4.1 on the page below for examples of the source text and reference summary for each of the three categories.

We use the ability to split the dataset into these partitions to develop several different summarization models, based on the 'extractiveness' level of the reference summaries the models are fine-tuned on. We remove all articles that consist of 4 sentences or less, but do no further preprocessing as the dataset is already in a processed format ready for training and evaluation

Each partition of the dataset was split into training/validation/evaluation subsets, using a 90/5/5 split.

Abstractive reference example

Reference Summary

En københavner får 1000 kroner i bøde for at overtræde Washingtonkonventionen ved at indføre krokodillehoveder - Krimi

Article body

En dille for krokodillehoveder har kostet en mand fra København en bøde på 1000 kroner. Det er Københavns Byret, som har udmålt bøden. Københavneren overtrådte Washingtonkonventionen samt Miljøministeriets bekendtgørelse om beskyttelse af vilde dyr og planter, da han i september 2010 forsøgte at få to hoveder fra siamesiske krokodiller ind i Danmark via Københavns Lufthavn i Kastrup. Hovederne fra krokodillen med det latinske navn Crocodylus siamensis havde han med sig fra Thailand. Men han havde ingen importtilladelse. Manden fik i første omgang en bøde, men den betalte han ikke, og sagen gik videre til retten. Her dukkede han ikke op, så det tog ikke dommeren mange minutter at idømme ham bøden.

Extractive reference example

Reference Summary

Disney håber på sigt at kunne tjene flere penge på sin egen tjeneste frem for at udleje sine film til Netflix.

Article body

Disney håber på sigt at kunne tjene flere penge på sin egen tjeneste frem for at udleje sine film til Netflix. Disney vil fra 2019 ikke længere levere nye film til Netflix. I stedet vil verdens største underholdningsselskab lancere sin egen streamingtjeneste. Det sker i et forsøg på at indfange seere, der dropper det traditionelle fjernsyn til fordel for det digitale. Disney håber på sigt at kunne generere et større overskud fra sin egen streamingtjeneste frem for at udleje sine film til tjenester som Netflix. Imens bruger Netflix og konkurrenter som Amazon.com og HBO millioner af dollar på at købe og producere eget indhold og streame det direkte til forbrugerne.
extractive

Mixed reference example

Reference Summary

Politiet efterforsker fortsat sagen om de to hunde, som natten til mandag blev stjålet fra en pension i Middelfart.

Article body

Fyns Politi har stadig ikke fundet de to hunde, som natten til mandag blev stjålet fra en hundepension i Middelfart. Politiet er sparsomme med oplysningerne, men ifølge Jørn Fischer fra lokalpolitiet i Middelfart, så efterforsker betjentene fortsat sagen. Den ene hund kommer fra Nyborg og er ifølge politiet af racen amerikansk staffordshire terrier. Den race er ulovlig, og politiet havde derfor bestemt, at dyret skulle aflives. Den anden hund er et gadekryds fra Odense, som tidligere i denne måned bed en anden hund meget voldsomt. Ifølge Fyns Politi er hundene forsvundet "under indbrudslignende forhold".

Figure 4.1: DaNewsroom - article-summary pair examples

4.2 Tv2

The second dataset we use for the automatic summarization tasks consists of 50,077 Danish article-summary pairs scraped from the regional news sites of the Danish news media **Tv2**. The article bodies in the Tv2 dataset are generally short-form news stories, with an average length of **197** words. The reference summaries are simply the subheadings of the news articles. Previous experience with a subset of the Tv2 dataset has led us to discover a set of challenges with the data in regards to automatic summarization. The primary challenge stems from the fact that the dataset uses the articles' subheadings as the reference summaries. In several instances, due to the articles' general short lengths, the sub-heading contains information that is not replicated in the article, or the article refers to events described in the subheading. This makes the production of meaningful summaries a challenging task, as there is information in the reference summary that is not contained in the text body. See examples in figure 4.2 below.

Example 1

Reference Summary

Den gode evert Johanne Dan forlod i dag sin vante plads ved Skibbroen i Ribe. Den blev løftet op af vandet og kørt til værftet i Årøsund.

Article body

En evert er et fladbundet lille sejlskib. I Europa blev det især brugt til sejlads på floder, mens det i Danmark er blevet brugt til sejlads på Vadehavet. Tidens tand er ikke gået sporløst hen over Johanne Dan, så nu venter en omfattende restaurering på værftet i Årøsund. Ikke den eneste De fleste everter er små en-mastede skibe, men den findes også som to-mastet. Den eneste danske evert, der fortsat sejler, er Rebekka fra Fanø. Hun blev bygget i 1921 på Fanø i Nordby.

Example 2

Reference Summary

Poul Erik Jensen blev mandag eftersøgt af politi, hjemmeværn og en helikopter ved Alssund.

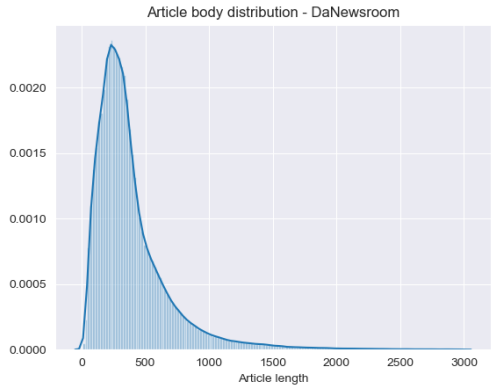
Article body

Han forlod søndag formiddag plejehjemmet Brohaven i Sønderborg. Han har tidligere været forsvundet, men han har holdt sig i nærheden af plejehjemmet. Manden er 175 cm høj, tynd og har kort, gråt hår. Han er iført en grå fleecejakke, forvaskede blå cowboybukser og sorte fodformede sko. Han kan genkendes på en særlig tilbagelænet, staccatoagtig gang, fortæller vagtchefen ved Syd- og Sønderjyllands Politi. Oplysninger til politiet på 1-1-4.

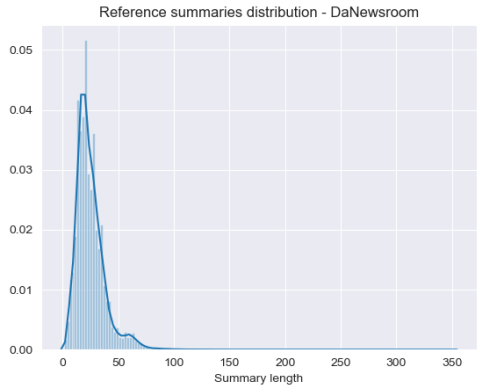
Figure 4.2: TV2 Examples - Low quality article-summary pairs

We see here that the key event(s) mentioned in the reference summaries are not mentioned or described in the article body. Consequently, there is less information in the text body to generate the summary from. Comparing the distribution plots and the data in table 4.1 it is clear that there is a significant difference in the average length of text bodies between the DaNewsroom and TV2 datasets.

We include the TV2 dataset together with DaNewsroom to investigate how different layouts of summarization datasets affect the downstream performance of summarization approaches.

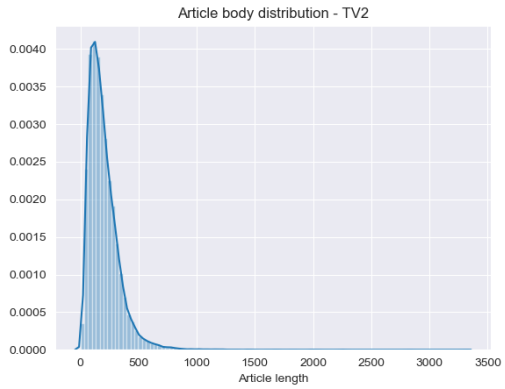


(a) Text body distribution

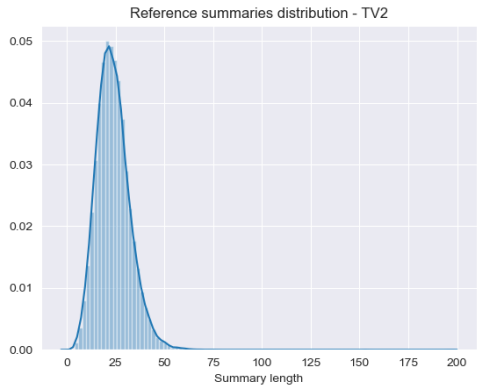


(b) Summary distribution

Figure 4.3: DaNewsroom dataset



(a) Text body distribution



(b) Summary distribution

Figure 4.4: Tv2 dataset

Dataset	Size	Avg. article length	Avg. summary length
DaN-Extractive	615,768	375	28
DaN-Mixed	229,759	446	21
DaN-Abstractive	287,204	395	21
DaN-Full (above three combined)	1,132,734	394	25
Tv2	49,609	197	23

Table 4.1: Summarization Datasets overview

4.3 TAC 2011 - Evaluation metric dataset

For testing summarization evaluation metrics there are generally few datasets available. The most widely used is the 2011 Text Analysis Conference (TAC) 2011 test dataset for the AESOP (Automatically Evaluating Summaries Of Peers) task [1]. The dataset is developed specifically for enhancing the development of summarization evaluation metrics. It contains 4488 multi-document summaries generated by automatic summarizers in the 2011 TAC summarization task. Each summary is written from a document set of 10 documents. There are 44 different topics, each with 2 different document sets. For each document set there are four human-written reference summaries generated manually by NIST (National Institute of Standards and Technology) assessors and 51 machine generated summaries. We are using the machine generated summaries as our candidate summaries, and the human-generated summaries as reference summaries.

The dataset contains human evaluations of the quality of each of the 4488 candidate summaries, measured across 3 dimensions; *Readability* which is an assessment of the linguistic quality of the candidate summary, *Responsiveness* which is a measure of content and linguistic quality, and *Modified Pyramid Score* which is a measure of how well the summary content reflects that of the original document(s). These evaluations forms the basis through which we are able to analyze the ability of our evaluation metric to accurately score summaries.

Feature	Content	Type
Summary	Candidate summary	String
refsum1-4	Reference summaries 1-4	String
Pyramid score	Modified pyramid score	[1-14]
Readability	Linguistic quality assessment	Likert scale [1-5]
Responsiveness	Overall responsivenes assessment	Likert scale [1-5]

Table 4.2: Dataset overview

[1] https://tac.nist.gov/2011/Summarization/AESOP.2011.guidelines.html

4.4 Preprocessing

For **DaNewsroom** we remove all instances of articles that consists of less than four sentences, as these articles are already short. For the **Tv2** data we remove all instances where the summary or text body are empty. Our preprocessing of the TAC data contains three steps; first, we assemble the data from the distributed files into one easy to manage CSV file. Secondly, we remove all instances where **Summary** feature is not a string. This leaves us with 4311 instances. The code for both steps is found in the `TSAuBERT/preprocessing` directory. The third and final step is to calculate ROUGE score baselines for each of the candidate summaries. We calculate ROUGE-1, ROUGE-2, and ROUGE-L scores for each candidate summary, taking the average score wrt. the four reference summaries as the final score.

5. Methodology

In this section, we outline the methodological approaches that form the basis for our experiments. Specifically, we describe the adapted BERT architecture developed by[Liu and Lapata, 2019] we use for summarization and how we use this architecture to train different summarization models. We further describe the conceptualization and design of the summarization evaluation metric we have developed.

5.1 Adapting BERT for summarization

For the task of summarization, the basic BERT architecture requires alterations. This is largely due to the way BERT produces vectors to tokens, and not sentences[Liu and Lapata, 2019]. This is a problem since most summarization models rely on token- as well as sentence-level manipulations for producing summaries. *Text Summarization for Pre-trained Encoders*, or PreSumm, introduced by [Liu and Lapata, 2019] presents an architecture addressing this, enabling the use of pre-trained BERT models for automatic summarization.

We use the PreSumm architecture and the accompanying fine-tuning strategies for extractive, abstractive, and mixed summarization to train each of our summarization models. We chose to use the PreSumm framework as it provides several advantages in relation to developing summarization models for a low resource language. Firstly, it provides a unified framework for fine-tuning both extractive and abstractive summarization models (although each strategy has its own unique fine-tuning setup) build on using pre-trained BERT models. The advantage of incorporating the use of pre-trained BERT models is that the amount of training/fine-tuning required to achieve good results on a downstream is considerably lower given that the BERT models already have a strong understanding of the structures of natural language [Devlin et al., 2019]. Furthermore, in the paper introducing PreSumm, the authors [Liu and Lapata, 2019] produced (at the time) state of the art results on several datasets, demonstrating the strength of the framework.

We explain the architectural changes that[Liu and Lapata, 2019] made to the original BERT layout to enable summarization, as well as the setup used to fine-tune summarization models for the three different summarization strategies in the following sections.

5.1.1 Text summarization for pretrained encoders

The alterations by [Liu and Lapata, 2019] to allow for summarization with BERT centers on developing a BERT-based encoder that can be employed for both extractive and abstractive summarization tasks. The authors adapt the original BERT encoder model in three ways; by inserting **additional `[CLS]` tokens**, introducing **alterating segmentation embeddings** and expanding the number of **positional**

embeddings. See figure 5.1 below for a visualization.

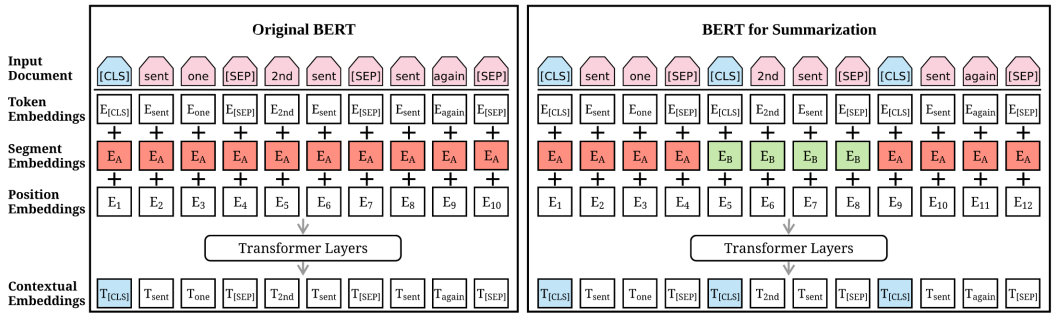


Figure 5.1: Figure from[Liu and Lapata, 2019] showcasing architectural changes for summarization - BERT (left) and BERTSUM (right)

Where the original BERT models prepend a single `[CLS]` token only at the start of the input sequence, for summarization, and extractive summarization especially, we need to be able to make decisions about the relevance of each sentence in a document, hence requiring a representation for each sentence. [Liu and Lapata, 2019] address this by inserting extra `[CLS]` tokens at the start of all sentences. This allows for a single vector level representation for each individual sentence [Liu and Lapata, 2019]. The authors of PreSumm further assign alternating values for each sentence in **the segment embedding** layer to distinguish when a sentence ends, and a new sentence begins in a document as illustrated in figure 5.1. Lastly, they increase the number of positional embeddings possible beyond the original 512 allowed by BERT models to allow for the larger input sequences needed for the summarization task. [Liu and Lapata, 2019] label their altered version of the BERT encoding architecture **BERTSUM**, and use it as the basis for all three summarization fine-tuning strategies implemented, each of which is described in the following sections.

5.1.2 PreSumm - Extractive Summarization

The task of Extractive summarization comes down to a classification task in which one needs to predict for a document D consisting of sentences $D = \{sent_1, sent_2 ... sent_n\}$ whether $sent_i$ should be extracted as part of the summary. Thus, for each sentence in the input document we need to assign a prediction value $\hat{y}$ indicating whether the sentence should be included in the summary.

Using the approach defined by [Liu and Lapata, 2019], we first derive the encodings for the input document by feeding it to the BERTSUM encoder. A given sentence in the input is represented by the embedding vector for its corresponding `[CLS]` token in the BERTSUM output. The vectors for the sentence `[CLS]` tokens are then fed to a two-layered transformer, each layer containing the two standard transformer sub-layers; a Multi-Headed Self-Attention and an FFN (see section 2.5.4.

These additional transformer layers add further positional information to each vector. These subsequent transformer-layers are fine-tuned together with the BERTSUM encoder[Liu and Lapata, 2019]. The final output vector for each sentence is then fed to a **sigmoid classifer** to calculate the prediction value $\hat{y}$ whether the sentence is to be included in the final summary or not. See figure 5.2 below for a visualization of the extractive summarization strategy.

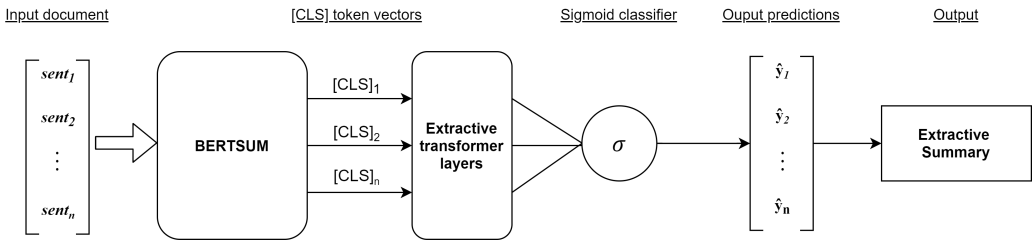


Figure 5.2: PreSumm - extractive summarization approach

The model extracts the 3 sentences with the highest probabilities from the predicted output.

The extractive model is fine-tuned on minimizing the binary classification entropy between the prediction value for a sentence $\hat{y}$ against the ground truth label y for that sentence. In order to facilitate this, we need to assign a value to each sentence in a document indicating whether that sentence is part of the gold standard extractive reference summary. PreSumm automatically does this by applying a greedy algorithm that produces an extractive reference summary of 3 sentences from the source text, which maximizes the ROUGE-2 score of the extractive reference wrt. to the original reference summary [Liu and Lapata, 2019]. This is a common technique to enabling extractive training using neural networks [Nallapati et al., 2016, Egonmwan and Chali, 2019].

5.1.3 PreSumm - Abstractive Summarization

Abstractive summarization is a language generation task that consists of iteratively predicting and generating each word in the output summary. PreSumm uses a standard encoder-decoder architecture using a `BERTSUM` model for the encoder, and a 6-layered transformer for the decoder, following the standard decoder architecture laid out by [Vaswani et al., 2017] (see section 2.5.4.4).

During encoding, the source document is parsed as input to a BERTSUM encoder, which generates a set of embedding vectors as output, which is then parsed to each layer of the decoder. These embeddings flow through the decoder layers. The output of the last layer of the decoder is parsed through a linear layer and a **softmax** layer to generate a probability distribution over the model's vocabulary. Based on the words with highest probability, the decoder auto-regressively generate the output

summary, using each predicted word as input for the next iteration. The process of decoding continues until an `end-of-sequence` (EOS) token or a `max_length` parameter is reached.

A **beam search** algorithm is applied when generating the output sequence for the abstractive summaries. Beam search uses conditional probabilities to consider multiple candidates at each time step when generating sequential output [Freitag and Al-Onaizan, 2017]. In language generation tasks, the beam search algorithm is applied on top of the probability distribution over the vocabulary to consider multiple candidate words to generate at each time-step. The benefit of beam search is that it is able to find potential high probability sequences unreachable by simple greedy algorithms.

A beam size parameter K designates the number of candidates considered at each time step. Figure 5.3 below provides a visual intuition on how a beam search progresses. Each edge has a probability attached to it.

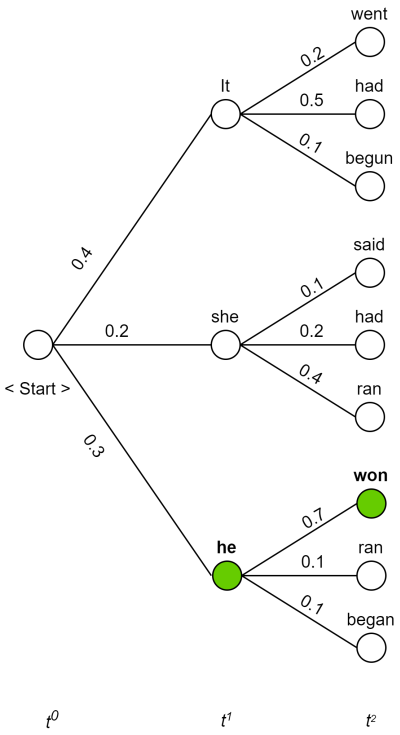


Figure 5.3: Beam Search example - beam size $K = 3$

At each time-step t^i we derive a new probability distribution over the vocabulary based on the input sequence and the output word from time step t^{i-1}, and take the K words from the vocabulary with the highest associated probabilities and expand from those. This process continues until an end-of-sequence or max length is reached. From the sequences, we have generated, we select the one with the highest

overall probability. In the above example we see that using beam search we generate the sequence *'he won'*, where a greedy algorithm would have generated the sequence: *'it had'*.

Using the setup from PreSumm, the beam size parameter $K = 5$ in the abstractive summarization models we have implemented. Increasing the beam size may lead to better generated sequences, however, this potential gain comes at a significant cost to decoding speed [Freitag and Al-Onaizan, 2017]. Figure 5.4 below provides an overview of the abstractive summarization strategy.

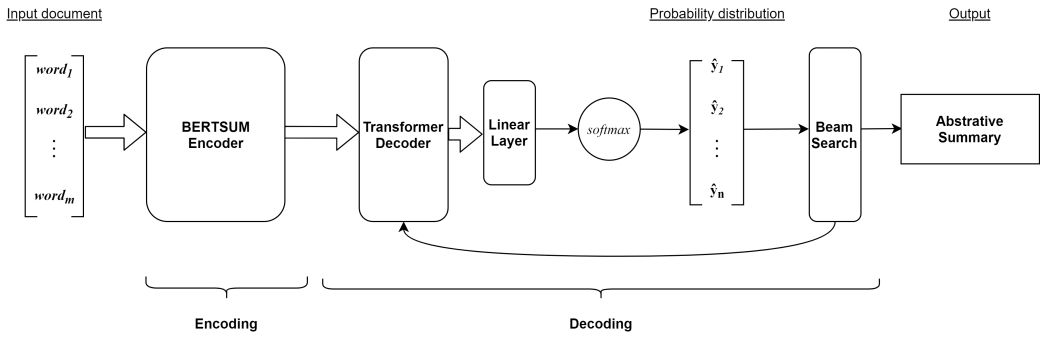


Figure 5.4: PreSumm - **abstractive** summarization approach

The abstractive summarization models are fine tuned on minimizing the Label Smoothing Loss. One of the interesting approaches PreSumm has used in fine-tuning an encoder-decoder architecture for abstractive is the separation of optimizers for the encoder and decoder. The assumption is that due to the pretrained nature of the encoder and the randomly initialized state of the decoder, the fine-tuning can be unstable [Liu and Lapata, 2019] if the same optimizer is used for both. [Liu and Lapata, 2019] performed a handful of experiments with different sets of learning rates and found this setup to give the best results.

5.1.4 PreSumm - Mixed Summarization

Besides the extractive and abstractive summarization strategies, PreSumm also introduces a two-stage training strategy mixing the two. The mixed summarization strategy employs components from both strategies in an attempt to combine the benefits of each.

The mixed summarization strategy of PreSumm uses an encoder-decoder architecture similar to the one used for abstractive summarization, but with the key distinction that the BERTSUM encoder is first fine-tuned on the extractive summarization task [Liu and Lapata, 2019]. The results reported by [Liu and Lapata, 2019] indicate that this summarization strategy produces better results than the purely extractive and abstractive strategies.

Following the mixed summarization setup from PreSumm, we first tuned the BERTSUM encoder using the extractive fine-tuning strategy outlined above. The fine-tuned BERTSUM is then instantiated as the encoder component of an abstractive summarization fine-tuning strategy. Thus, the mixed summarization strategy is simply to first fine-tune an encoder on extractive objectives and then use this encoder in an abstractive summarization setup outlined in the above section. The assumption is that the fine-tuning on extractive objectives will improve the abstractive summarizers performance [Liu and Lapata, 2019].

We will be comparing results from summarization models based on all three strategies using different pre-trained BERT models. The altered PreSumm code used for this project can be found on github [1]

5.2 TSAuBERT

We introduce Textual Similarity Assessment using BERT, or **TSAuBERT** (pronounced 'zaubert'), an evaluation system based on utilizing the contextual word-embeddings derived from pre-trained BERT-based language models to assess the similarities between two sequences of text. The aim is to provide an insight into how pre-trained transformer models based on the BERT architecture can be used to evaluate textual similarities in an unsupervised manner, and how well they can be used to evaluate automatically generated summaries.

Given a candidate summary C of n tokens and reference summary R of m tokens, we represent the **semantic similarity** between a given candidate and reference token as the **cosine similarity** between their BERT embedding-vectors. The score of a candidate summary is defined by the overall level of similarity between the embedding vectors for the tokens in the candidate summary wrt. the reference summary.

The evaluation process for the TSAuBERT algorithm contains two steps; **1)** The encoding of the summaries using pre-trained models to generate embedding vectors for each token, and **2)** The scoring approaches implemented that define how the final score for the candidate summary is calculated.

Each functionality of TSAuBERT was tested using one or multiple **unit-tests**. The full test harness can be found under the `TSAuBERT` directory in the code accompanying the thesis.

5.2.1 Encoding process

We use the contextually dependent embedding vectors derived from BERT-based models to represent each token in each of the input summaries. The encoding process contains three distinct steps.

[1] Altered PreSumm - https://github.com/SebastianVeile/PreSumm

Step 1, for each summary, we tokenize (using the wordpiece tokenizer used by BERT models) and encode each sentence sequentially, deriving a set of embedding vectors for each sentence. We then remove the vectors for the `[CLS]` and `[SEP]` tokens as well as the vector for the "." at the end of each sentence, leaving us with the vectors for the original tokens in the text. [2]. See model 5.5 below for an example.

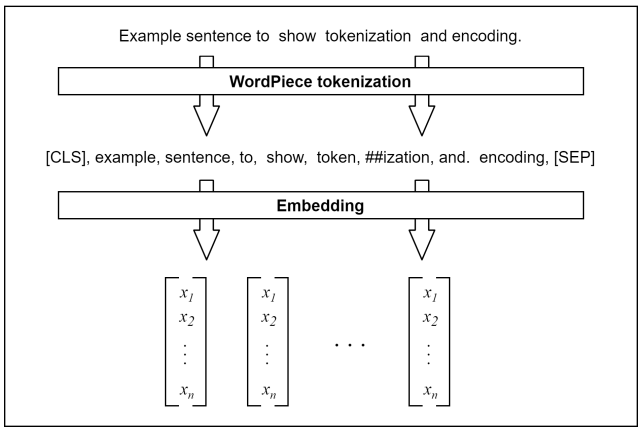


Figure 5.5: Basic embedding example

For a detailed description of the wordpiece tokenizer and how the final embeddings for a token in a sentence is derived via self-attention mechanisms, please see section 2.6. The following two steps of the encoding process for TSAuBERT are optional steps, designed to enable different levels of vector representations of the summaries.

Step 2 is an optional pooling of vectors, for those words that have been split into several tokens by the wordpiece tokenizer. As described in section 2.6, when the wordpiece tokenizer encounters OOV words, it splits them into a number of know tokens. In example, the word `'tokenization'` is split into the tokens `{'token', '##ization'}`. We implement a functionality that recognizes when a word has been split, and pools together its vectors to form **one** vector representing the original word. To pool together the embedding vectors, we simply take their mean, as this is a common technique for combining embedding-vectors. See model 5.6 below for an example.

The intuition behind this follows that it provides us with a more accurate representation of the original word. The aim is that this leads to more accurate similarity scores downstream in the evaluation metric.

Step 3 is an optional *n-gram* or *sentence-level* pooling for each sentence in a summary. This functionality allows us to pool the token-vectors in a sentence according

[2]The code for the encoding process can be found in the `encode.py` script in the `TSAuBERT` directory

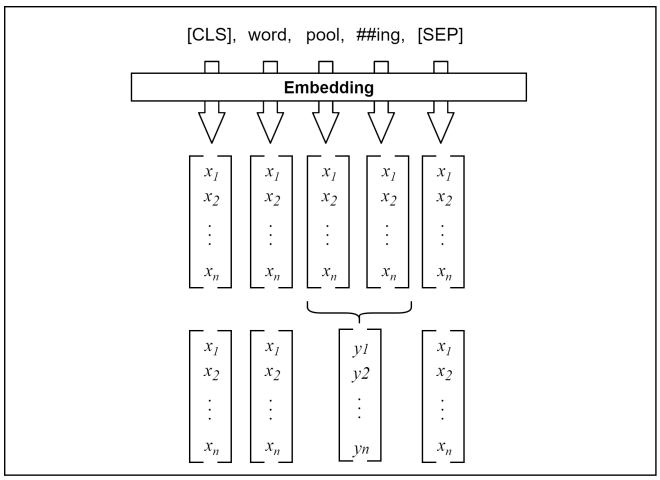


Figure 5.6: WordPiece pooling example

to the size of n-gram n. In example, with $n = 2$, the sentence `{we went to the bank.}` contains the following 2-grams (called bi-grams):
`{(we, went), (went, to), (to, the), (the, bank)}`, for n-gram pooling in this example we simply combine the vectors for the words in each n-gram, giving us a new representation of the original sentence. For sentence-level pooling we simply pool together all the token-vectors in a sentence to form one vector for the entire sentence.

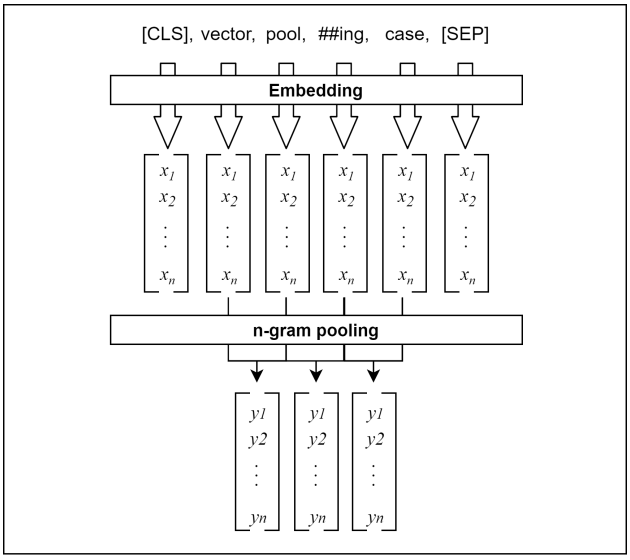


Figure 5.7: N-gram pooling, n = 2, no pooling of wordpieces

The intuition follows that by combining token vectors, we are able to generate different levels of representation for the summary. We investigate the effects of different

n-gram levels of encoding on the accuracy of the scoring metric. Lastly, we take each of the final vectors and add them to one matrix, forming the final representation for the summary. Note that in figure 5.7 above, we do not include the vectors for the `[CLS]` and `[SEP]` in the final set of embeddings. These vectors are never included in the final set of embeddings for a sentence, regardless of n-gram encoding level.

5.2.2 Scoring approaches

We implement and analyze two distinct scoring approaches for deriving the final similarity score between a candidate and reference summary [3]. Both approaches use the *cosine similarity* between embedding vectors to express the semantic similarity of the token(s).

Cosine similarity (CS) is used to measure the similarity of two vectors in a given vector space. It ranges from $[-1, 1]$ 1 indicating that the vectors are equal to each other. It is defined as the **dot product** of the two vectors adjusted for the **magnitude** of each vector. Formally, the CS of two non-zero vectors $\vec{v}$ and $\vec{w}$ is defined by:

$$CS(\vec{v}, \vec{w}) = \frac{\sum_{i=0}^{n} v_i \cdot w_i}{\sqrt{\sum_{i=0}^{n} v_i^2} \cdot \sqrt{\sum_{i=0}^{n} w_i^2}}$$

In the context of word embeddings, we can think of the CS between embedding vectors from a common vector space as a measure of semantic relation between the original words. Given that the $CS(\vec{v}, \vec{w}) < 1$ for all cases where $\vec{v} \neq \vec{w}$, we can use it as a *soft measure* of textual similarity [Zhang* et al., 2020] rather than relying on a hard exact-match lexical measure. This conceptualization has proven useful for a variety of NLP tasks concerned with approximating textual similarity, such as sentiment classification [Thongtan and Phienthrakul, 2019], text classification [Li, 2013] and paraphrase classification [Zhang* et al., 2020].

Mean based scoring
The first scoring approach we define, represents the similarity score of the candidate summary wrt. the reference summary as the **average CS** between their embedding vectors.

Given a candidate summary C represented as a set of embedding vectors $V = \{\vec{v_1}, \vec{v_2}, ..., \vec{v_n}\}$, and a reference summary R represented as a set of embeddings vectors $W = \{\vec{w_1}, \vec{w_2}, ..., \vec{w_m}\}$, the **mean-based** TSAuBERT score for C wrt. R is defined as:

[3] The code for the scoring process can be found in the `score.py` script in the `TSAuBERT` directory

$$TSAuBERT_{mean}(C, R) = \frac{1}{n} \cdot \sum_{i=0}^{n} (\frac{1}{m} \cdot \sum_{j=0}^{m} CS(\vec{v_i}, \vec{w_j}))$$

For each vector in the candidate summary, we calculate the *mean* CS for that vector wrt. to all the vectors in the reference summary. The final score for the candidate summary is the average all the candidate vectors' similarity scores.

Argmax based scoring
The second scoring approach uses a maximization strategy to represent the similarity between the candidate and reference summaries. For each vector in the candidate summary, the similarity score is the **max CS** score wrt. the reference vectors. For the **argmax** scoring approach we use a *precision* and *recall* measure to calculate an *f1* score, which represents the final argmax score of the summary. This scoring approach follows the one develop for the BERTscore metric [Zhang* et al., 2020].

Given a candidate summary C represented as a set of embedding vectors $\{\vec{v_1}, \vec{v_2}, ..., \vec{v_n}\}$, and a reference summary R represented as a set of embeddings vectors $\{\vec{w_1}, \vec{w_2}, ..., \vec{w_m}\}$, the **argmax-based** TSAuBERT score for C wrt. R is defined in terms of the following functions:

$$Precision(C, R) = \frac{1}{n} \cdot \sum_{i=0}^{n} argmax(CS(\vec{v_i}, \vec{w} \in R))$$

$$Recall(C, R) = \frac{1}{m} \cdot \sum_{i=0}^{m} argmax(CS(\vec{w_i}, \vec{v} \in C))$$

$$TSAuBERT_{argmax} = F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Using precision, recall and f1 for this scoring approach has the advantage of adjusting for the variable lengths of the candidate and reference summaries. See figure 5.8 below for a model over the workflow of `TSAuBERT`.

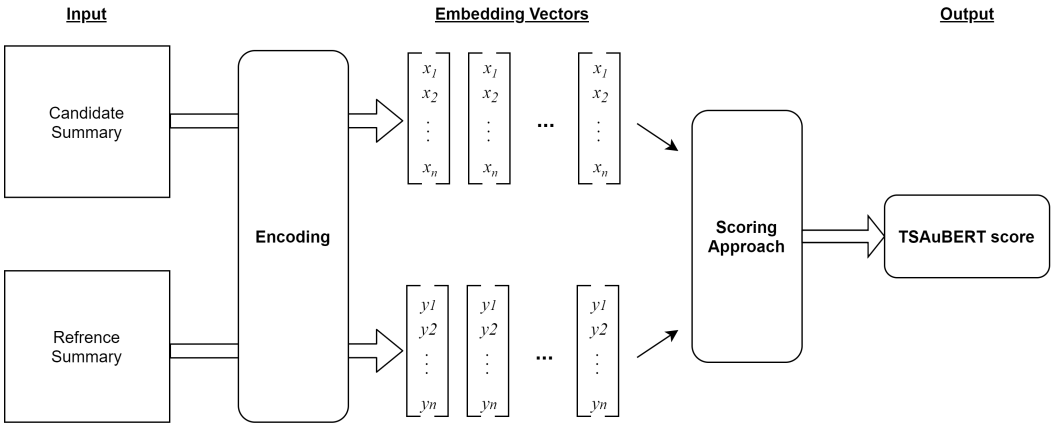


Figure 5.8: TSAuBERT workflow

The conceptual framework we present for `TSAuBERT` is developed to address several of the problems of ROUGE presented in section 2.3.2.2. Firstly, we address the lexical bias by relying on embedding-vector similarities rather than lexical overlap. Using the BERT embedding vectors means that `TSAuBERT` is able to account for different contextual meanings of the same word given that the BERT embedding vector for a given word differs dependent on the context in which it is mentioned.

Following the recommendations made in section 3.2 in regards to new summarization evaluation metrics, we argue that `TSAuBERT` fulfills several of these criteria as well. It is conceptually clear how the candidate summary is scored wrt. the reference summary. It scores summaries on a different level than ROUGE, looking at vector similarity in context-dependent embedding vectors, providing additional detail to the existing evaluation protocol. Given that we have based `TSAuBERT` on the popular `huggingface/transformers` library [Wolf et al., 2019], it is easy to simply use a BERT model pre-trained on a different language to produce the embedding vectors. Hence `TSAuBERT` can be used for any language with an available pre-trained BERT model (or one can use the multilingual BERT model). Finally, we have made all code, related to both the `TSAuBERT` scoring algorithm, test harness, analysis scripts, etc. available at `https://github.com/Lukecn1/TSAuBERT`.

6. Experimental setup

This section describes the fine-tuning strategy for each of the summarization models we develop and the experimental setup employed to test `TSAuBERT`'s capability as a summarization evaluation metric.

6.1 Automatic summarization experiments

Given the multiple datasets (see table 4.1), pre-trained BERT models (DaBERT and mBERT) and summarization strategies (see section 5.1) we end up with a total of **27** different summarization models to be fine-tuned and evaluated. In order to provide a clear distinction between each model, we present our naming convention for the different summarization models in figure 6.1 below.

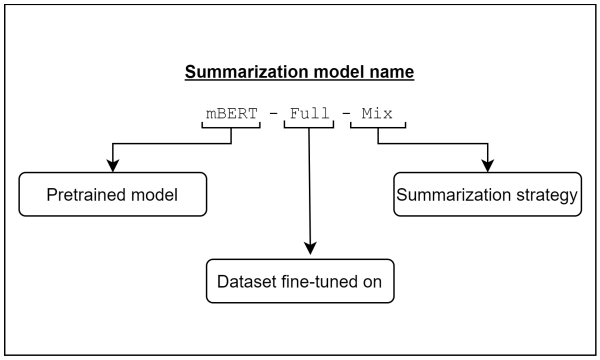


Figure 6.1: Naming convention for summarization models

We follow the above convention when referring to a summarization model. We provide an overview of a subset of the summarization models in table 6.1 below. For an overview of each summarization model, see table A.1.1 in the appendix.

Summarization model name	**Pre-trained BERT model**	**Dataset fine-tuned on**	**Summarization strategy**
DaBERT-Ext-Abs	DaBERT	DaN-Extractive	Abstractive summarization
DaBERT-Mix-Mix	DaBERT	DaN-Mixed	Mixed summarization
DaBERT-Full-Ext	DaBERT	DaN-Full	Extractive summarization
mBERT-Ext-Ext	mBERT	DaN-Extractive	Extractive summarization
mBERT-Full-Mix	mBERT	DaN-Full	Mixed summarization
mBERT-TV2-Abs	mBERT	TV2	Abstractive summarization

Table 6.1: Summarization Models subset - overview

Following the above naming convention, when referring to a given summarization model as an abstractive, extractive, or mixed summarization model, we refer to the summarization strategy that the model uses.

6.1.1 Extractive fine-tuning

All models fine-tuned using the extractive summarization strategy were fine-tuned on an RTX-2070 GPU for ~25 hours for 50.000 steps. We use a batch size of 2000 and accumulate the gradients at every 8th step. We save a checkpoint for every 1000 steps and select the top 3 checkpoints based on the validation loss from the validation set. We generate summaries on the top 3 checkpoints and select the one that produces the highest ROUGE scores[1] as our final model. We use the adam optimizer with a of $warmup = 10000$, a $learning - rate = 0.002$ as well as $\beta 1 = 0.9$, and $\beta 2 = 0.999$.

6.1.2 Abstractive & Mixed fine-tuning

Initially, all summarization models fine-tuned using the abstractive summarization strategy were fine-tuned on an RTX-2070 GPU for 3-6 days, depending on the pre-trained BERT model (DaBERT or mBERT) between 100.000-200.000 steps. We use a batch size of 10 and accumulate the gradients every 30 steps. We save a checkpoint for every 2000 steps and select the top 3 checkpoints based on the validation loss from the validation set. We generate summaries on the top 3 checkpoints and choose the one that produces the highest ROUGE scores as our final model.

We use the same separate optimizer scheduling as PreSumm (see section 5.1.3) with a higher learning rate and lower warm-up for the decoder. Label smoothing has also been utilized with an $\alpha = 0.1$. We applied a dropout rate with a probability of 0.2 for all linear layers.

We use the same settings for the mixed summarization strategy. The difference being that for fine-tuning mixed summarization, for each individual mixed summarization model, the encoder is the BERTSUM component from the best validated extractive summarization model (for that particular dataset). Otherwise, the fine-tuning for mixed summarization follows the same schedule as that for abstractive summarization.

6.1.3 Hyper-parameter optimization for DaBERT

The majority of summarization models were fine-tuned using the hyper-parameter values, as outlined in the above sections. However, during the initial fine-tuning of the different summarization models, it became apparent that the abstractive summarization models using DaBERT struggled in reducing the model loss compared to the abstractive models based on mBERT. Therefore, we decided to run a hyper-parameter optimization for the `DaBERT-Abs-Abs` summarization model with the aim of testing whether it could reach similar levels of validation loss as the `mBERT-Abs-Abs` model. The final set of hyper-parameters we for the optimization with are:

- Learning rate encoder

[1] We used ROUGE-1, ROUGE-2, and ROUGE-L F1 scores.

- Learning rate decoder
- Accumulate count (number of steps before we accumulate the gradients)
- $\beta 1$
- $\beta 2$

For optimizing we have used the HyperOpt python library[2] and the Tree-structured Parzen Estimator Approach(TPE). TPE minimizes the objective function by looking at the previous result and utilize Bayesian reasoning to select the next set of parameters. [Bergstra et al., 2011]

The challenge with optimizing for this task is that fine-tuning abstractive summarization models on any dataset are computationally expensive and takes a significant amount of time. Combined with the limitations of the hardware available to us, it means that we are unable to do full iterations of the models when optimizing. Instead, we run each optimization iteration on a smaller number of steps (2000 in this case) and run it for 30 iterations with a total running time of approximately 95 hours. We select the 3 best models from our results, which are then fine-tuned and compared on a larger number of steps (12000) for a combined time of 72 hours. We present the results from this experiment in section 7.1.

6.1.4 Evaluating summarization models

We evaluate all summarization models developed in this project using a selection of ROUGE metrics, see section 2.3.2.1. We use pyrouge[3], a wrapper for the original ROUGE implementation in python, to evaluate all our models. We calculate the following ROUGE scores per model: ROUGE 1, 2 and L and use the F1 of each to evaluate the performance of the summarization models.

The original ROUGE implementation only has an English stemmer, which does not perform well on Danish words. Therefore we have implemented a Danish stemmer using the NLTK Snowball library[4]. A stemmer reduces words to their base form, such that words like *'jumping'* and *'jumped'* would be reduced to their common base form; *'jump'*. If a stemmer is not used, abstractive summarization systems tend to be punished wrt. their ROUGE scores, as abstractive systems, tend to paraphrase sentences. We stem both the candidate and reference summary prior to calculating the rouge scores. See figure 2.3 for an example of the evaluation process using ROUGE.

6.1.5 Baseline summarization models

Firstly, we compare the performance of the various summarization models we have fine-tuned with a series of baseline summarization models. All fine-tuned sum-

[2]HyperOpt - https://github.com/hyperopt/hyperopt
[3]Pyrouge - https://pypi.org/project/pyrouge/
[4]Danish Stemmer - https://www.nltk.org/api/nltk.stem.htmlnltk.stem.snowball.DanishStemmer

marization models were held up against the Lead3 baseline. The Lead3 baseline consists of comparing the first three sentences of the source document to the reference summary. [Nallapati et al., 2016]. Besides Lead3, the models fine-tuned on DaNewsroom dataset were compared against two additional baselines; TextRank and ICSISum, both of which are provided by the authors of the dataset [Varab and Schluter, 2020]. TextRank is an unsupervised graph-based extractive summarization algorithm, while ICSISum revolves around selecting and extracting sentences that contain the most important bi-grams [Varab and Schluter, 2020].

It is important to note that the authors of DaNewsroom have not used stemming in the calculation of their rouge scores. Having done so might have impacted their rouge scores by a small margin.

6.1.6 Mono vs. Multi-lingual BERT models

To assert which of the pre-trained BERT (DaBERT and mBERT) models that generally leads to the generation better summaries for Danish, we compare their performance across different summarization strategies and datasets. The performance of a summarizer is measured by ROUGE-1, ROUGE-2, and ROUGE-L scores. The intend of comparing multiple different summarization models trained for both DaBERT and mBERT with different summarization strategies on different summarization datasets is to ensure increased robustness of the results, allowing for a more solid analysis of the performance differences between the different pre-trained models.

6.1.6.1 Investigating Layout bias in news data

As previously mentioned in section 2.2.3, when producing summaries of news articles, the first sentences in the source text often contain the central pieces of information relevant to the article, referred to as the **layout bias** in news summarization. In order to investigate our summarization models' reliance on sentence-position in the source text when producing summaries, we compare a subset of the summarization models' performance on a dataset before and after shuffling the sentences of the source text.

We take the `DaN-Full` evaluation dataset and shuffle the article sentences before evaluating the selected summarization models. Comparing the ROUGE-scores of the candidate summaries before and after shuffling gives us an indication of the degree to which the models rely on sentence-position when generating summaries. For evaluation, we have selected each of the summarization models; `mBERT-Full-Ext`, `mBERT-Full-Abs`, `mBERT-Full-Mix`, giving us a total of three summarization models with different summarization strategies. The intend is to strengthen the validity of our results by including summarizers based on different summarization strategies.

6.2 TSAuBERT experiments

To assess the effectiveness of `TSAuBERT` as a summarization evaluation metric, we analyze `TSAuBERT`s ability to accurately score summaries by testing its correlation with the human assessments available in the TAC 2011 AESOP dataset (see section 4.3). Using correlation with manual evaluations done by human annotators is a common approach to assess the quality of a summarization evaluation metrics [Peyrard, 2019]. The idea is that the greater the correlation with human assessments, the more accurate the metric is in truly indicating the quality of the candidate summary.

To do so, we first calculate the average `TSAuBERT` score of the candidate summary wrt. the four reference summaries. Then we calculate the level of correlation of the `TSAuBERT` score for each of the three human metrics. We calculate three distinct correlation statistics; **Pearson's r**, **Kendall's Tau** and **Spearman's rank coefficient**. These measure correlations between two variables in slightly different manners, thus allowing for a more detailed analysis of the correlation patterns of `TSAuBERT`. We compare the human evaluation correlation scores of the best performing TSAuBERT iterations with correlation scores for ROUGE. The aim is to get an understanding of whether these metrics differ in their ability to evaluate the quality of automatically generated summaries.

An important note to be made in relation to the TAC 2011 AESOP Dataset is that although the dataset provides a set of baseline ROUGE scores and correlation scores between these and the human evaluation metrics, we have not been able to reproduce these using the guidelines for different scoring approaches provided by [TAC, 2017]. In order to alleviate potential problems stemming from this when comparing `TSAuBERT` and ROUGE scores, we generate all ROUGE scores using a stable software package, and calculate Pearsons, Spearmans and Kendalls correlations for the ROUGE scores against each of the human evaluation metrics following the same exact procedure as for `TSAuBERT`.

We test `TSAuBERT` using 4 different pre-trained models all based on the BERT architecture; `bert-base-uncased`, `bert-large-uncased`, `mbert-base-uncased` and `sentence-bert`. See section 2.6.3 for a description of each.

6.2.1 Parameter optimization for TSAuBERT

In order to find the combination of parameters that yield the highest correlation with human assessments, we employ a **random search** strategy [Bergstra and Bengio, 2012] over the search space of possible parameter-combinations for `TSAuBERT`. Random search is a well established hyper-parameter tuning strategy for neural networks, and have shown empirically that it is capable of finding as good parameter sets as grid-search strategies [Bergstra and Bengio, 2012]. The key benefit of random search is that it greatly reduces the amount of time taken to find a good set of parameters for a given model.

Although `TSAuBERT` is not a trained neural network model we can still utilize an optimized search strategy find good combinations of parameters that yield higher correlations with human evaluations. As we are dealing with a relatively computationally intensive task (given multiple sentences and multiple references), random search allows us to find an optimal (or close to optimal) combination of model parameters while spending fewer resources.

Parameter	Values
BERT model	bert-base-uncased bert-large-uncased bert-base-multilingual-uncased sentence-bert
Encoding layers	base : [7-12] large : [18-24]
Wordpiece pooling	[True, False]
Encoding level	n-gram : [1-8] sentence-level
Scoring approach	[mean, argmax]

Table 6.2: Parameter overview

6.2.1.1 Random search setup

The random search was done across three stages, with 4 iterations (1 for each pretrained BERT model used) per stage. Each iteration ran 8-10 combinations of `TSAuBERT` parameters. After each stage, the results were analyzed, and the search space reduced for the following stage. Our **objective function** to be maximized was the *average Pearson correlation* of the `TSAuBERT` score wrt. the human assessment metrics. The random search was conducted on 20% of the TAC 2011 data. The reduction of the amount of data effectively reduces the search space, potentially limiting the ability of the search strategy to find optimal/close to optimal parameter combinations. The reduction is done because the significant computational time of running `TSAuBERT` on all instances with all reference summaries. Thus the amount of data used is reduced in order to make the random search feasible, both in terms of time and the resources spent.

The data was shuffled using a random seed allowing for replication of our reported results[5]. We analyze correlation scores for all instances in each search, so as to gain a better indication of how to reduce the search space for the next stage.

We utilize the best parameter-sets from the random search as the basis of the analysis of the performance of `TSAuBERT` as a summarization evaluation metric. We report the results from the random search in section 7.3.1.

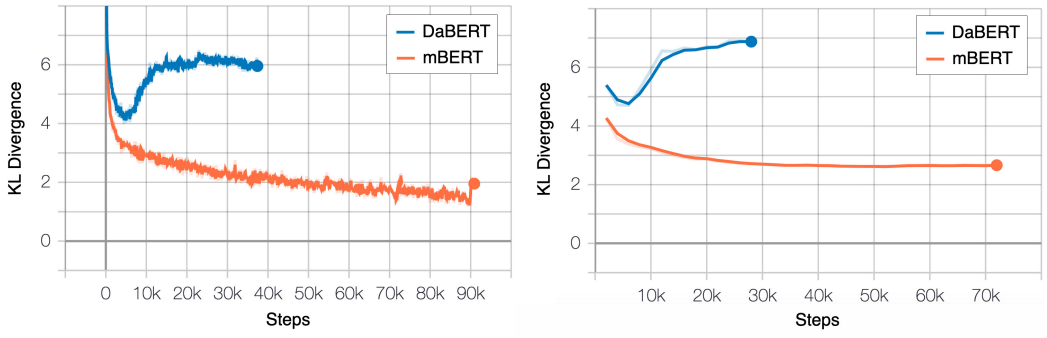
[5]The code for the random search is found in `TSAuBERT/parameter_optimization` directory

7. Results and analysis

In this section, we present the results of the experiments related to our Danish summarization models and the performance of `TSAuBERT` as a summarization evaluation metric. The section concludes with an overview of the key findings, providing answers to the research questions.

7.1 Results of Hyper-parameter optimization

Here we present the results from hyper-parameter optimization experiments described in section 6.1.3. The aim is to investigate whether the `DaBERT-Abs-Abs` summarization model could reach similar levels of validation loss as the `mBERT-Abs-Abs` model. This was based on observations made during the initial fine-tuning of these models. Looking at the training and validation plots 7.1a and 7.1b, we see that the DaBERT based summarization model is a lot more unstable than the mBERT based one.



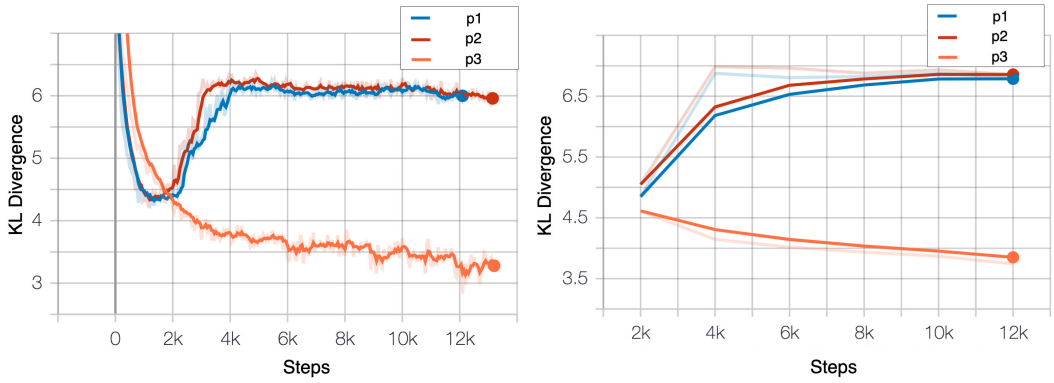
(a) Training progress

(b) Validation progress

Figure 7.1: Training/Validation of mBERT and DaBERT

Although we see a decreasing loss for the first 5000 steps, the DaBERT models' loss jumps and does not seem to be able to reach a better point. We hypothesize that the model is getting stuck on a local minimum and cannot get out with the original parameters. Using the `HyperOpt` python package, we look for better parameters that minimize the loss. The top 3 sets of parameters of the 30 iterations can be seen in figure 7.2. We compare the top 3 sets of parameters and select the best. We then compare the training/validation loss to that of mBERT for both the new and old set of parameters for DaBERT.

In figure 7.2, we clearly see that the parameter set **p3** performs the best in terms of reducing the loss function. We select this parameter set and fine-tune the `DaBERT-Abs-Abs` summarization model with them. The figure below shows the training and valida-

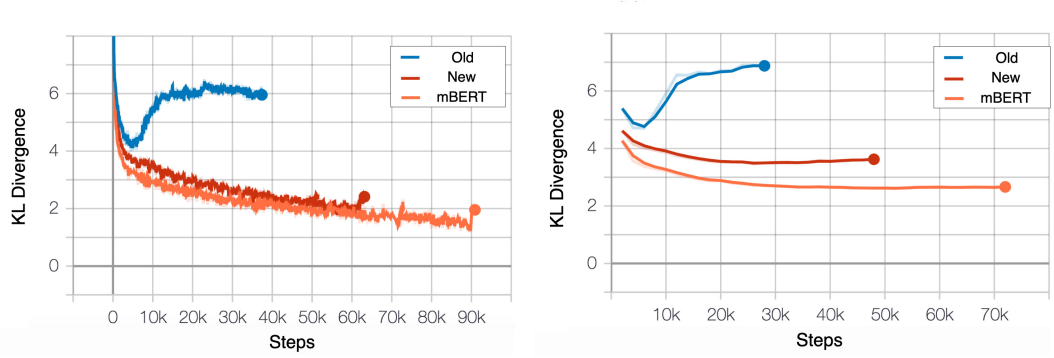


(a) Top 3 parameters training

(b) Top 3 parameters validation

Figure 7.2: Top 3 selected sets of parameters

tion loss before and after the parameter optimization.



(a) Training progress after tuning hyperparameters

(b) Validation progress after tuning hyperparameters

Figure 7.3: Training/Validation of mBERT and old vs new parameters of DaBERT

It is clear that using the new parameters, the `DaBERT-Abs-Abs` model performs significantly better in terms of reducing training and validation loss. It now performs only slightly worse than the `mBERT-Abs-Abs` model during validation as shown in 7.3b. See table 7.1 below for a comparison of old vs new parameter values for the new parameter values.

Parameter	Initial Value	New Value
Learning rate encoder	0.002	0.004
Learning rate decoder	0.2	0.1982
Accumulate count	30	89
$\beta 1$	0.999	0.913
$\beta 2$	0.9	0.981

Table 7.1: old vs new parameter values

Even though the best set of parameters landed very close to the original set, two things stand out. The encoder learning rate has doubled, and the accumulate count has nearly tripled. We hypothesize that because the gradients are accumulated at a much later step, than with the original set of parameters, we are able to make a much more calculated decision of how to adjust the weights. Thereby not making the mistake of getting stuck in a sub-optimal local minimum.

However, this comes with a cost since the models now can only train ~12000 steps in 24 hours. As a result, we have demonstrated that the performance in terms of validation loss could be improved via hyperparameter optimization. However, the improved hyperparameter values combined with the limited hardware at our disposal means that it is significantly more time-consuming for us to fine-tune using these. The original trend we observed during the fine-tuning and validation for the `DaBERT-Abs-Abs` model, we also observed for all abs and mix DaBERT based models. Based on the findings from the hyper-parameter tuning experiments, we use the new set of parameters to fine-tune all of the abs and mix strategy DaBERT based summarization models for the remaining experiments.

7.2 Summarization for a low resource language

In this section, we present the evaluation results of all our 27 Danish summarization models that we have fine-tuned on the various datasets, (See table 4.1 for an overview of the datasets) as well as the baseline models. By analyzing results across a wide range of different datasets, summarization strategies, and pre-trained models, we aim to increase the robustness and validity of our results. We are thus gaining a better understanding of the strengths and challenges of using pre-trained BERT models for automatic summarization of a low-resource language.

7.2.1 Evaluation results

We report the evaluation results using ROUGE F1 scores in the 7.2 table below, with the best performing models for each dataset marked as bold. (For the naming convection of our fine-tuned summarization models we refer to figure 6.1 in section 6.1).

Evaluation dataset **DaN-Full**

Models	R1	R2	RL
Baselines			
Lead 3	43.02	35.41	40.82
TextRank	26.92	14.95	22.23
ICSISum	26.83	14.99	22.22
Our models			
mBERT-Full-Abs	48.33	39.98	44.98
mBERT-Full-Ext	43.93	36.20	41.72
mBERT-Full-Mix	**49.99**	**41.62**	**46.67**
DaBERT-Full-Abs	33.52	25.77	29.97
DaBERT-Full-Ext	43.93	36.14	41.69
DaBERT-Full-Mix	36.12	29.08	33.11

Evaluation dataset **DaN-Mix**

Models	R1	R2	RL
Baselines			
Lead 3	28.50	14.07	24.14
TextRank	22.10	8.15	16.66
ICSISum	20.71	7.33	14.90
Our models			
mBERT-Mix-Abs	33.50	16.93	26.72
mBERT-Mix-Ext	29.23	14.53	24.75
mBERT-Mix-Mix	**33.59**	**16.98**	**26.74**
DaBERT-Mix-Abs	21.27	9.61	16.06
DaBERT-Mix-Ext	29.95	15.01	25.25
DaBERT-Mix-Mix	21.89	10.11	16.57

Evaluation dataset **DaN-Abs**

Models	R1	R2	RL
Baselines			
Lead 3	19.47	4.89	15.38
TextRank	15.11	2.73	10.73
ICSISum	14.72	2.54	9.97
Our models			
mBERT-Abs-Abs	**22.16**	**6.59**	**17.03**
mBERT-Abs-Ext	19.81	5.00	15.65
mBERT-Abs-Mix	22.07	6.50	16.91
DaBERT-Abs-Abs	13.64	3.52	10.00
DaBERT-Abs-Ext	20.84	5.34	16.34
DaBERT-Abs-Mix	13.96	3.66	10.24

Evaluation dataset **DaN-Ext**

Models	R1	R2	RL
Baselines			
Lead 3	59.72	57.81	59.20
TextRank	34.07	23.00	29.47
ICSISum	34.20	23.54	29.40
Our models			
mBERT-Ext-Abs	69.54	66.65	68.77
mBERT-Ext-Ext	61.33	59.47	60.87
mBERT-Ext-Mix	**71.07**	**68.20**	**70.40**
DaBERT-Ext-Abs	37.28	30.07	34.08
DaBERT-Ext-Ext	61.18	59.30	60.70
DaBERT-Ext-Mix	37.83	31.54	34.50

Evaluation dataset **Tv2**

Model	R1	R2	RL
Baselines			
Lead3	24.85	8.54	20.00
Our models			
mBERT-Tv2-Abs	29.26	10.64	**21.84**
mBERT-Tv2-Ext	26.86	9.08	21.27
mBERT-Tv2-Mix	**29.54**	**10.67**	21.79

Table 7.2: ROUGE F1 scores for our summarization models - including baselines

Based on all the results in the above tables, several interesting patterns emerge. Firstly, we see that several of our fine-tuned summarization models beat the baseline models' ROUGE scores for each of the datasets. In some cases, by even 20 and 30 points as for the **DaN-Ext** dataset. In fact, the best performing summarization model is consistently one we have fine-tuned. This allows us to claim state of the art results for Danish summarization on each of the datasets. We see that for all but the **DaN-Abs** dataset, that the mBERT based summarization models that use the **mixed** summarization strategy are the best performing models. This follows similar findings from [Liu and Lapata, 2019]. The fact that mBERT based summarization

models consistently outperform the baseline models for all ROUGE scores and all datasets speaks to the strength of the summarization approach developed by [Liu and Lapata, 2019] and the utility of pre-trained BERT models.

All our extractive models beat the Lead3 baselines for each dataset; however, they are all quite close. This supports the idea that extractive summarization models have some positional bias towards selecting the first few sentences when generating the summaries. We will look further into this in section 7.2.4.

All abstractive mBERT summarization models beat their extractive counterpart, which indicates that the abstractive models have an advantage that allows them to generate better summary sentences. We look further into the abstractiveness of our models and the difference in scores between mBERT and DaBERT models in section 7.2.5. However, this is not the same for DaBERT, where all abstractive models perform significantly worse than the extractive DaBERT summarization models. We further see that the extractive DaBERT summarization models perform comparatively well or sometimes better than the mBERT extractive models. These are quite interesting findings as they provide an important insight into the performance differences between the mono-lingual DaBERT model and the multi-lingual mBERT model on different types of down-stream tasks. We investigate the implications of this finding in greater detail in 7.2.3.

We see an interesting pattern that the ROUGE scores for summarization models fine-tuned on datasets where the **reference summaries** are less extractive (Dan-Abs, Dan-Mix, and Tv2), are generally lower than for the more extractive reference summary datasets. This finding suggests that for datasets with more abstractive reference summaries, the generation of very high scoring summaries is challenging. This is likely a result of the more abstractive reference summaries being generally shorter in length. Thus, if the candidate summaries are longer the ROUGE scores tend to decrease as using the F1 measure tends to penalize instances where there are larger differences in the size of candidate and reference summaries. The tables below containing the precision, recall and F1 score for a series of the summarization models fine-tuned on datasets with more abstractive reference summaries, corroborate this.

	ROUGE-1			**ROUGE-2**			**ROUGE-L**		
Summarization model	Precision	Recall	F1	precision	Recall	F1	Precision	Recall	F1
mBERT-Abs-Abs	19.40	29.76	22.16	5.73	9.06	6.59	14.75	23.39	17.03
mBERT-Abs-Ext	13.78	43.83	19.81	3.41	11.72	5.00	10.82	35.34	15.65
mBERT-Abs-Mix	19.21	29.86	22.07	5.60	9.06	6.50	14.56	23.40	16.91
DaBERT-Abs-Abs	9.00	34.49	13.64	2.30	9.43	3.52	6.55	26.22	10.00
DaBERT-Abs-Ext	14.45	45.61	20.84	3.65	12.38	5.34	11.33	36.54	16.34
DaBERT-Abs-Mix	9.30	33.31	13.96	2.43	9.30	3.66	6.85	25.32	10.24

Table 7.3: Recall, Precision and F1 scores for subset of **DaN-Abs** summarization models

We see clearly that the recall scores are much higher than the precision scores, indicating that generally, the candidate summaries are significantly longer than the reference summaries leading to lower ROUGE F1 scores. Interestingly, we see from the results in table 7.3 that the mBERT based summarization models suffer less from this issue than those based on the DaBERT pre-trained model. This suggests that the DaBERT models might struggle with generating too long summaries and, as a result, have lower ROUGE scores. We analyze this finding further in section 7.2.3.

7.2.2 Summary examples

In order to gain a deeper understanding of the structure of the language of the summaries we automatically generate, we analyze a handful of the candidate summaries in the paragraphs below. We showcase and analyze candidate summaries that both highlight some strengths and weaknesses of the different models (See summary examples in appendix A.2).

While the extractive summaries have the advantage of always being grammatically coherent, the same cannot be said for the abstractive summaries. The fluency of the abstractive summaries is good; however, in some summaries, a reappearing pattern occurs; the abstractive summarization models struggle with interpreting pronouns, locations, and objects. For example, the `mBERT-Abs-Abs` summary in figure 7.4 below, where the first sentence is good, coherent, and conveys the information from the source text.

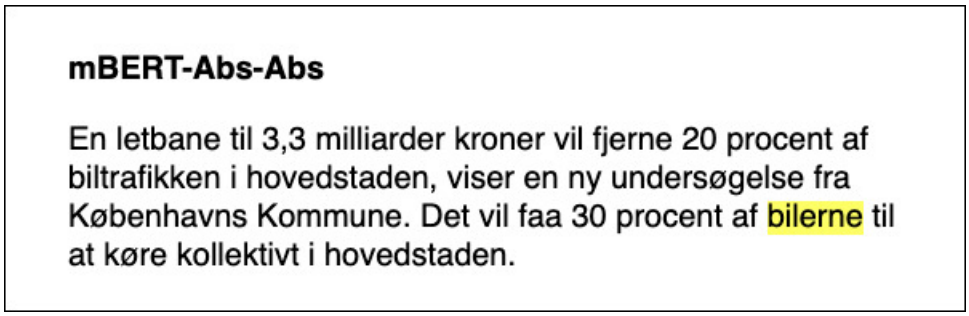


Figure 7.4: Abstractive summary example (full figure in A.2.5)

But in the second sentence, it messes up by referring to the cars even though the source document refers to citizens/people. If instead of generating the word *'bilerne'* the summarizer would have generated the word *'borgerne'* or *'bilisterne'*, this would be a near perfect summary, both in terms of content and fluency and grammatical coherence. In another example from the `mBERT-Abs-Mix` summarizer in figure 7.5 we see a similar thing.

mBERT-Abs-Mix

Den kommende letbane mellem Gladsaxe og København vil fjerne 20 procent af biltrafikken paa Frederikssundsvej og Gladsaxe, viser en undersøgelse fra Metroselskabet, Tetraplan og Cowi.

Figure 7.5: Mixed summary example (full figure in A.2.5)

The summary is fluent and grammatically coherent, but it messes up location names, thus not conveying correct information in its entirety.

Some models also have difficulties with negation, which we see in the mBERT-Full-Mix summary from figure 7.6. The source text clearly states that "Radikale Venstre" **does not** support the proposal of the "SF" party. However, the Mix model has interpreted it as they **do** support the proposal, which creates a fluent grammatical coherent summary, but also a summary that does not convey the information of the source text. Critically the summary states the opposite of the truth in this matter.

mBERT-Full-Mix

Det er et samfundsanliggende, at alle, der bor i Danmark, er bevidste om de frihedsrettigheder, de har, mener De Radikale, der bakker op om et forslag fra SF om, at højst hver tredje elev paa en folkeskole maa have anden etnisk baggrund end dansk

Figure 7.6: Mixed summary example (full figure in A.2.1)

This supports the findings from [Ettinger, 2019] that based on a case study of the strengths and weaknesses of BERT models found that BERT models generally struggle to demonstrate a general understanding of negation in language [Ettinger, 2019]. This can be considered a risk when using BERT for summarization tasks, as it may influence the summarization models' ability to deal with negation.

Even though this is something BERT struggles with, it still is able in some cases to interpret it correctly, as shown in the abstractive summary example from figure 7.7.

Figure 7.7: Abstractive summary example (full figure in A.2.1)

In this case, the summarization model correctly interprets the information that "Radikale venstre" **does not** support the proposal from "SF". The end result is a concise and fluent summary that also conveys the correct information from the source text. More summary examples for the rest of the models can be found in the appendix A.2.

Besides highlighting the generated summaries, we also want to look into the data used to fine-tune, generate, and evaluate the summarization models. The source document and reference summary is an essential part of the summarization task and can both positively and negatively impact the output and evaluation thereof. The reference summaries in figures below are clearly flawed in that they are cut short and does not summarize the important content of the source.

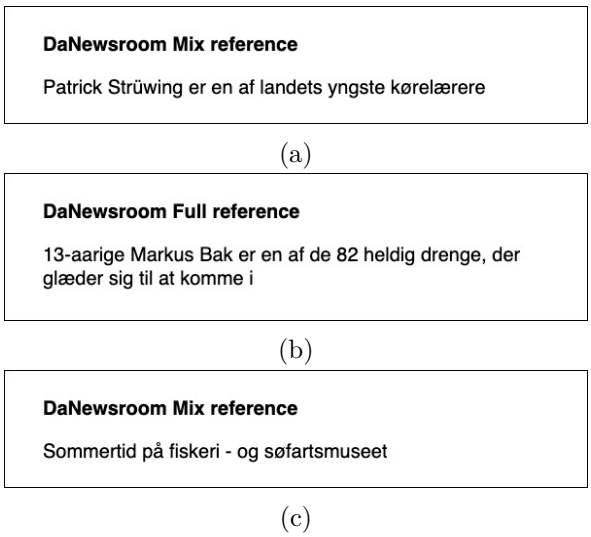


Figure 7.8: Reference summary examples

Holding our generated summaries up against these references is somewhat problematic since the reference summary is supposed to constitute a standard for what

constitutes a good summary of the source document. This has a negative impact on ROUGE scores and, if used for fine-tuning, might hold the model back from improving. Some reference summaries are less explicit but still problematic. An example of that is the reference summary c above. This is not cut short, however, it is unable to capture the most important information from the source text. Hence, evaluating any candidate summary against this does not provide a true insight into the quality of the candidate summary.

All of this mostly comes down to the way the DaNewsroom dataset is scraped using meta-tags from articles. These are all created by humans, but as the authors of DaNewsroom found, Danish news websites have not adapted fully to the use of meaningful meta-tags for summaries [Varab and Schluter, 2020]. The authors try to filter out the problematic reference summaries, but some are getting through since this process is all automated. The meta-tags are also often just the subheadings of the articles, which brings its own problems, also seen in the TV2 dataset. These subheadings do not always summarize the article's body but are more about expanding the article title and/or preface to the article. We discuss the implications of the structure of the reference summaries further in the discussion.

7.2.3 Monolingual vs. Multilingual

In this section, we further analyze the performance differences of the summarization models based on the of the two pre-trained BERT models, DaBERT and mBERT, used in this project. As described earlier, the two BERT models did not perform on par when using the same hyper-parameters. Therefore the parameters have been optimized for DaBERT summarization models to bring them closer to the performance levels of the mBERT summarization models.

In table 7.2, the results clearly show that the multilingual based abstractive and mixed summarization models generally outperform those based on the Danish BERT model. The same does not apply to the extractive models, where DaBERT and mBERT are relatively equal in performance. By examining the summaries generated by the DaBERT models using abstractive and mixed summarization (See figure 7.9 7.10 below), it quickly becomes clear why we see these low scores. DaBERT generated summaries are typically a lot longer than summaries generated with mBERT models, which significantly impacts the ROUGE scores. DaBERT seems to have issues knowing when to stop and, in some cases, generates sequences of random tokens at the end. This also impacts the general readability and fluency of the summaries. This pattern repeats for most abstractive/Mix DaBERT models.

When examining DaBERT summaries, we encounter another significant issue. Scattered within most of the summaries are `[CLS]` tokens, which are not supposed to be part of the summary.

DaBERT-Abs-Abs

michal jensen var tæt på at vinde sin første kamp i bordtennisturneringen, men blev slået med 4-2. [CLS] [CLS] [CLS] danskeren blev slået i to sæt og er dermed klar til at spille sig videre i kvartfinalen. [CLS] se klippet her. [CLS] - sport [CLS] [CLS]dk [CLS] [CLS] vært : peter rosenind (s) [CLS] [CLS] ([CLS] [CLS]) [CLS] (: [CLS] [CLS])

Figure 7.9: Example of `DaBERT-Abs-Abs` summary (full figure in A.2.6)

DaBERT-Abs-Mix

den danske bordtennisspiller bobi simion var tæt på at blive klar til kvartfinalen i bordtennisturneringen i rio. [CLS] [CLS] [CLS] peter rosenmeier og peter rosenindmeier måtte se sig slået af peter rosenindmeier i kvartfinalen i bordtennis. [CLS] - sport [CLS]dk [CLS] [CLS]dk [CLS] [CLS] - danmark [CLS] [CLS]dk bull ; dr sportens peter rosenind (s) [CLS]) : [CLS] [CLS] se video

Figure 7.10: See full figure with source A.2.6 - Example of DaBERT-Abs-mix summary

After investigating, we see that the summarization models directly predict these tokens in the decoding phase. This problem does not occur for the abstractive and mixed mBERT based summarization models. Furthermore, the input data for both DaBERT and mBERT summarization models is pre-processed and split in the exact same way, with the only difference being the BERT models' wordpiece tokenizers. The code used for fine-tuning and evaluation is also the same; the only difference again being the BERT model used and the hyper-parameter values. Thus, the most significant difference between the mBERT based and DaBERT based summarization models is the pre-trained BERT model.

We hypothesize that two key factors contribute to the generation of `[CLS]` tokens in the candidate summaries generated by DaBERT based abstractive and mixed summarization models. Firstly, and most significantly, the **pre-training of the DaBERT model** and the pre-training data. DaBERT is pre-trained on various sources from the **Danish common-crawl**, but also scraped data from Danish debate forums like **heste-nettet** and **dindebat**. In general, the structure of the language in these sources are sub-optimal for BERT model pre-training, as they

are short-form containing many small sequences of text, with the naturally occurring language idiosyncrasies of debate fora, such as grammatical errors and slang. Furthermore, the Danish **Common-crawl** data also contains significant portions of non-Danish data (primarily Norwegian) [Strømberg-Derczynski et al., 2020]. These factors combined result in the DaBERT model being pre-trained on less data of lower quality than the mBERT models.

We argue that as a result, the DaBERT model struggles with generating text, as it has not been pre-trained on data that allows for the generation of a comprehensive understanding of the long-term sequential dependencies in natural language. The authors of BERT state that the structure, in particular the length, of the sequences in the pre-training data is crucial in order for the BERT model to develop a proper understanding of language [Devlin et al., 2019].

The second reason as to why the DaBERT based summarization models predict the `[CLS]` tokens we argue is the BERTSUM encoder model laid out by [Liu and Lapata, 2019] that we use in our summarization models. BERTSUM inserts a `[CLS]` token in front of every sentence in the input text. DaBERT might see the `[CLS]` token enough to use it in its prediction, reasoning that it is important for the source text. In the figures 7.9 and 7.10 the `[CLS]` tokens mostly appear at the very beginning of a sentence or after a period (.), thus indicating that the models are heavily influenced of the `[CLS]` token at the start of every sentence in the source. Given that the mBERT based summarization models do not produce `[CLS]` tokens as output, we hypothesize that the stronger pre-training of the mBERT model means that it is able to see past these in the input when generating the summaries.

Thus, we see that for the simpler extractive summarization task, the difference in pre-training of the DaBERT and mBERT models does not significantly impact performance. For the more complex abstractive and mixed summarization tasks, the difference in pre-training between the two BERT models used has a massive effect on performance. This difference is likely a result of extractive being a relatively simple classification task, thus not relying on as complex a language understanding as the one needed for language generation and abstractive summarization. Given the better performance of the `mBERT` based summarization models, we utilize a subset of these for the remainder of the experiments.

7.2.4 Layout bias in news data

In this section, we present the results of the experiments investigating the degree to which our extractive-based summarization models rely on the first portion of sentences in the source text when generating summaries. Firstly we investigate the positions from which a subset of our extractive models selects sentences for candidate summaries. See figure 7.11 below for a visualization of this.

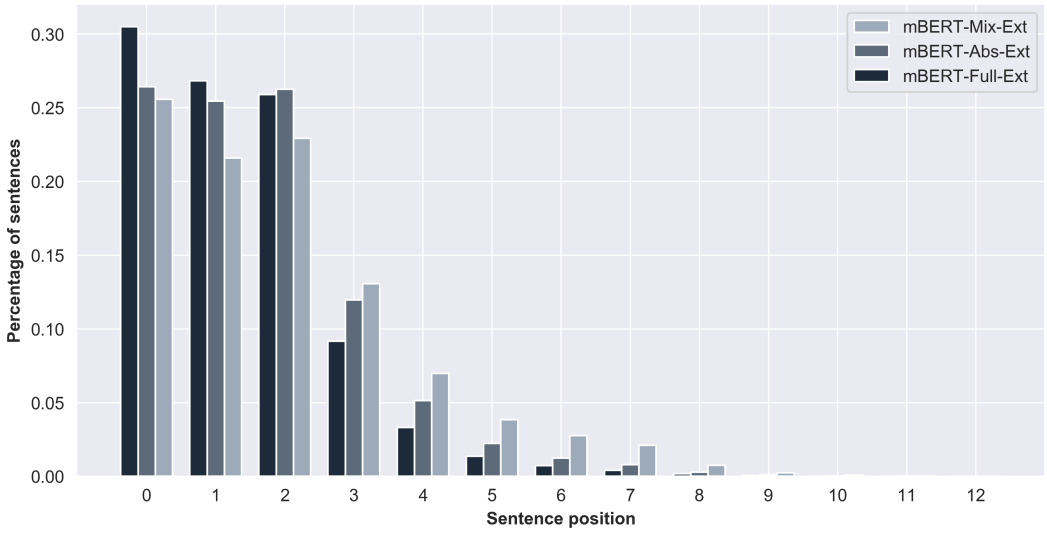


Figure 7.11: Position of selected sentences for extractive models

The figure shows that the extractive models extract the first 3 sentences to a much larger degree than the other sentences in the source text. This indicates that our models have been affected by the layout of the news articles they are trained on. This reaffirms similar findings from [Kedzie et al., 2018] and [Liu and Lapata, 2019], that the learning signal for deep learning-based news summarization models is heavily influenced by sentence position [Kedzie et al., 2018]. The likely cause of this follows from the structure of news articles, where the first sentences contain key pieces of information regarding the article[Kryscinski et al., 2019].

We study the degree to which our models have been affected by this bias comparing the sentence selection distribution before and after shuffling the source text sentences for the `mBERT-Full-ext` model on the evaluation set for `DaN-Full`. The results of this are shown in figure 7.12 below.

The results showcase that there is some form of improvement to how the extractive models select their sentences. Generally, we see that the first 3 sentences are selected less, in favor of later sentences. However, it still shows a heavy bias towards the first three sentences in general. This suggests that the two-layered transformer that `PreSumm` applies atop of `BERTSUM` embeddings in their extractive summarization model to incorporate positional information of sentences have been biased towards selecting the first three sentences during training.

Given that the reference summaries for the extractive models are generated by greedily selecting the three sentences that maximize the ROUGE-2 score wrt. the original reference summary, the above finding further indicates that the reference summaries include similar information as the first couple of sentences. To investigate this fur-

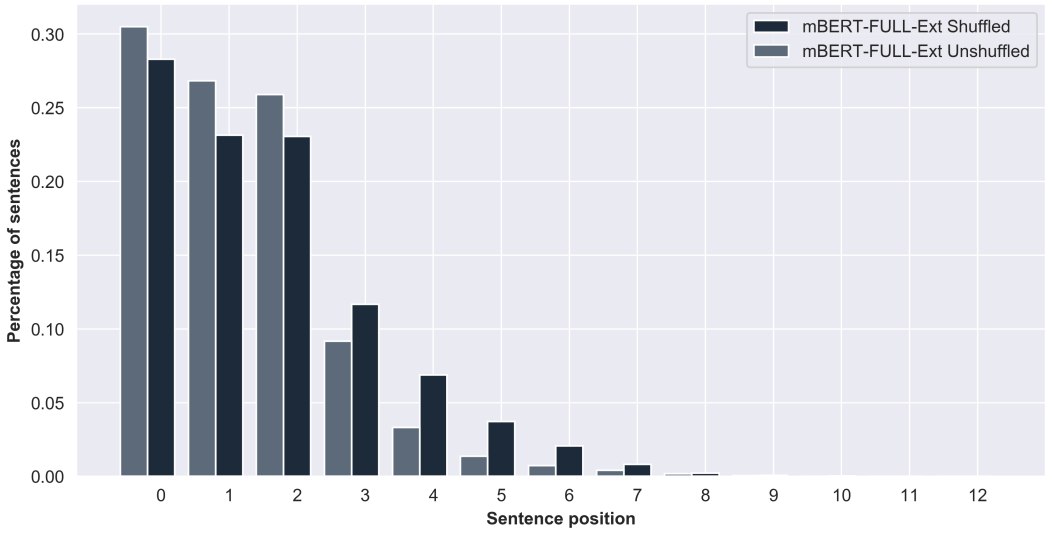


Figure 7.12: Position of selected sentences for extractive models - post shuffling

ther, we compare the ROUGE scores of a subset of our summarization models before and after we shuffle the sentences of the evaluation dataset we apply them on. The results from this are found in the table below.

	Evaluation set DaN-Full-Unshuffled			Evaluation set DaN-Full-shuffled		
Model	R1	R2	RL	R1	R2	RL
Lead3	43.02	35.41	40.82	21.32	8.00	17.63
mBERT-FULL-Abs	48.33	39.98	44.98	23.75	9.80	17.77
mBERT-FULL-Ext	43.93	36.20	41.72	22.94	9.62	19.19
mBERT-FULL-Mix	49.99	41.62	46.67	24.90	10.92	18.99

Table 7.4: ROUGE F1 Comparison of Shuffled vs. Unshuffled source sentences

The results clearly show that we see a significant drop in the ROUGE scores after shuffling the sentences. For the extractive based models, this is an indication of the fact that the extractive reference summaries generated often include the first three sentences. The abstractive models also show a large drop in rouge scores after shuffling, which indicates that the model is unable to adapt and heavily relies on the positional biases of sentences in news articles. This likely also means that our summarization model will not fare well when applied to other domains of summarization than news articles.

Although this may be seen as a weakness of the summarization models, one could argue that it is instead the result of the models having learned to incorporate an important feature of the data during training.

7.2.5 Measuring abstractiveness of summaries

We have conducted some smaller experiments on the output of a subset of the summarization models to measure how abstractive their outputs really are, that is how many unique n-grams are included in the candidate summary that are not in the source text. For each candidate summary and source document, we split all words into subsets of n-grams. By doing so, we can compare the difference in n-grams of the candidate summary with n-grams of the source document. See the figures on the following page.

Figure 7.13 showcases that the proportion of unique n-grams is generally relatively low. Thus there are very few n-grams in the candidate summaries that do not also appear in the source document. This indicates that many of the summaries generated by the `mBERT-FULL-Abs` and `mBERT-FULL-Mix` models stay close to the wordings of the source documents, especially if we compare these results to that of figure 7.14.

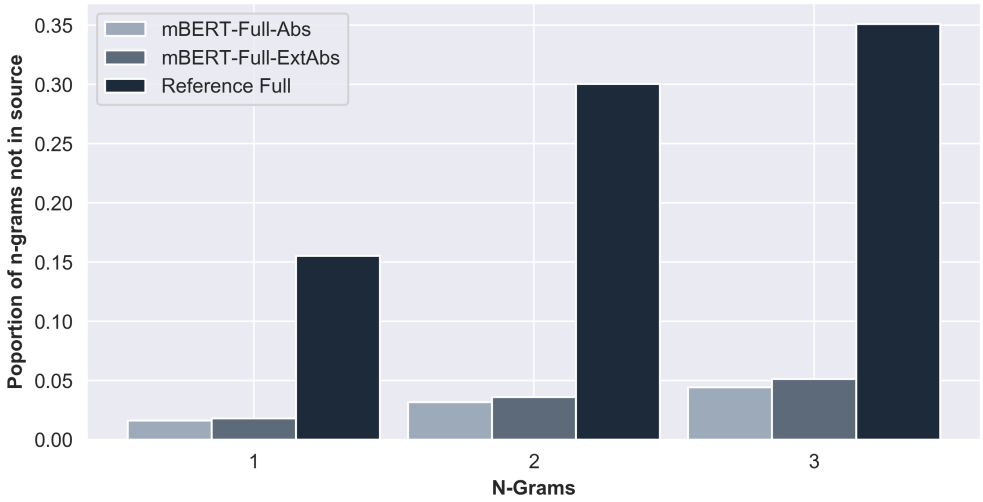


Figure 7.13: Unique n-grams in the candidate summaries produced by `mBERT-Full` models

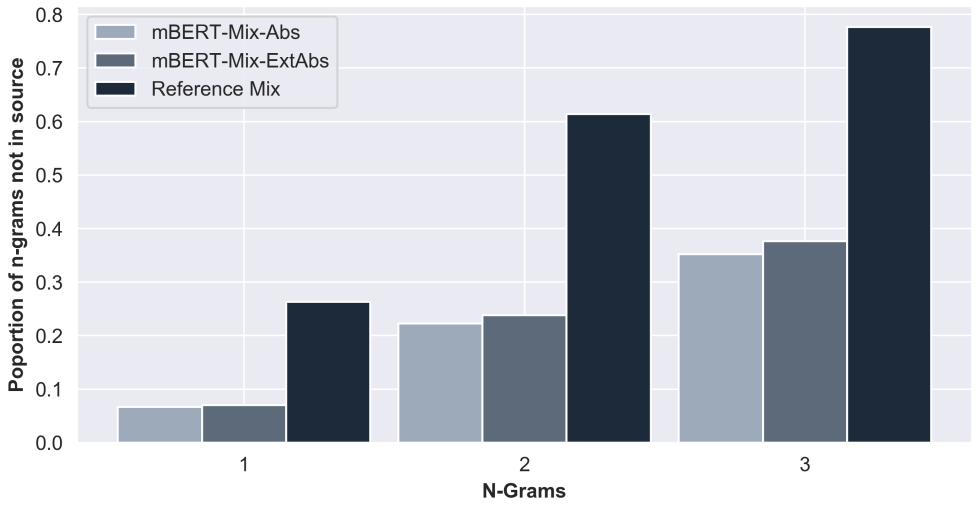


Figure 7.14: Unique n-grams in the candidate summaries produced by `mBERT-Mix` models

The `mBERT-MIX-Abs` and `mBERT-MIX-Mix` models contain a higher proportion of unique n-grams, which indicates that the summaries generated using the `mBERT-MIX` models are a lot more abstractive than summaries generated by the `mBERT-Full` models. A likely cause for this follows from the fact that for the `mBERT-FULL` models more than 60% of the dataset used to train these consist of extractive reference summaries, while the rest is a combination of Mix and abstractive.

Due to the extractiveness of `mBERT-Full` summarization models, we have also looked at the proportion of whole sentences from the candidate summaries that also appear in the source document. It turns out that 71% and 66% of all whole sentences generated by `mBERT-Full-Abs` and `mBERT-Full-Mix` are also sentences that appear in the source document while this is only the case for 7% and 6% of the sentences for `mBERT-Mix-Abs` and `mBERT-Mix-Mix`. This further supports the idea that even though the `mBERT-Full` summarization models perform better wrt. ROUGE scores, they also generate very extractive summaries even for summarization strategies that are not purely extractive. The implication of this is that for summarization models trained mostly on extractive data will generally be biased towards replicating the specific phrasings of the source document.

7.3 Analysis of TSAuBERT

We present here the results of the experiments assessing the performance of `TSAuBERT` as a summarization evaluation metric. We measure performance by the correlation between `TSAuBERT` versions and the three human evaluation metrics in the TAC dataset.

Firstly we present the results of the random search parameter optimization ex-

periments, describing the best parameter-combinations found. Based on these, we present the results from a series of smaller experiments investigating the performance differences resulting from changing `TSAuBERT` parameters like; `BERT-layer` and `n-gram size`. We conclude by comparing the performance of `TSAuBERT` with the ROUGE metrics, highlighting their strengths and weaknesses, reflecting on the capabilities of `TSAuBERT` as a summarization evaluation metric.

7.3.1 Random search results

We report the best overall results from all of the stages of the random search in table 7.5 below.

	Parameters				Pearson correlations			
Model	Layer	Wordpiece pooling	n-gram	Scoring	Pyramid	Readability	Responsiveness	avg.
bert-base-uncased	12	no	1	argmax	0.490	0.267	0.436	0.398
bert-large-uncased	19	no	1	argmax	0.483	0.255	0.429	0.392
mbert-base-uncased	12	no	1	argmax	0.509	0.248	0.439	0.399
sentence-bert	8	no	1	argmax	0.527	0.212	0.436	0.392
bert-base-uncased	9	-	sentence	argmax	0.216	0.300	0.309	0.275
bert-large-uncased	20	-	sentence	argmax	0.251	0.291	0.346	0.296
mbert-base-uncased	12	-	sentence	argmax	0.261	0.262	0.289	0.271
sentence-bert	10	-	sentence	argmax	0.268	0.238	0.322	0.276

Table 7.5: Random Search - Best combinations found

The results clearly indicate that **argmax** based scoring performs better than **mean** based scoring. This is likely a result of when taking the average *cosine similarity* we smooth/average out any significant similarities and/or dissimilarities in individual vector comparisons making a final score less indicative of an overall level of similarity.

We further see that the best instances have an n-gram size of 1, indicating that combining several word-level vectors might not produce better representations of the underlying tokens. This may be a result of the fact that the contextually dependent embedding-vectors already incorporate information on the neighbouring tokens, following the self-attention mechanism inherent in the BERT models. In relation to this, we see that when using 'sentence-level' vectors, `TSAuBERT` performs significantly worse. This reaffirms similar results from other textual similarity studies [Reimers and Gurevych, 2019]. These findings suggest that when using contextually dependent embedding vectors for similarity assessment tasks, using individual token vectors provides the best results overall. We explore this further in 7.3.2.2.

In regards to pooling of vectors for tokens split by the wordpiece tokenizer, the best combinations found during the random search all had this set to 'False'. The intuition explaining these results follows the one laid out above for n-gram size; pooling of individual vectors, whether those of split words or not, does not provide a better representation for comparison using cosine similarity.

In regards to the layer of the BERT model from which we retrieve the vector, the last layer seems most preferable for the mBERT and BERT-base models, whereas, for

sentence-BERT and BERT-large, layers around the upper-middle seem to perform best. We explore this further in section 7.3.2.1.

An interesting result is that the `TSAuBERT` version based on the mBERT pre-trained model performs slightly better on pyramid and responsiveness compared to BERT-base, which is trained only on English data. However, given that their difference in correlation scores is rather insignificant, we are not comfortable claiming that either is clearly favorable. An important note to be made is that, in general, we do not see strong correlations between the different versions of `TSAuBERT` and the human evaluations. Although there exists some positive correlation, it is not substantial. We discuss this finding further in relation to the correlation scores of the ROUGE metrics in section 7.3.3, and further elaborate on the complexity of summarization evaluation in the discussion of this project.

7.3.2 Probing analysis of TSAuBERT parameters

In order to further understand the performance implications of changing the parameter values for `TSAuBERT`, we conduct a series of smaller experiments using the `TSAuBERT` versions derived from the random search. Specifically, we test the effect of changing the encoder layer of the BERT model where we retrieve the embedding vectors and the effect of increasing the n-gram size..

The aim is to strengthen our understanding of the relationship between changes in the parameters and the `TSAuBERT` metrics performance as a summarization evaluation metric. All experiments in this section are done using all 4311 candidate summaries in the TAC dataset. The `TSAuBERT` score for a given candidate summary is the average f1 score of the *argmax* scoring approach, wrt. the four reference summaries.

7.3.2.1 Layer-wise probing

The intuition of the layer-wise probing experiments follows that as the embeddings flow through the transformer layers in the BERT models, different degrees of contextual information are encoded, yielding different input representations. Note that we only probe a subset of the upper layers of each of the BERT models, as previous research has indicated better performance for similarity assessment tasks from these [Zhang* et al., 2020]. We hold all other parameters constant, and only change the layer of the BERT model from which the vectors are retrieved.

The figures below show the change in **Pearson correlation** between the `TSAuBERT` score and the human evaluation metrics when changing the BERT layer. (For all correlation results, see A.4.1 in the appendix)

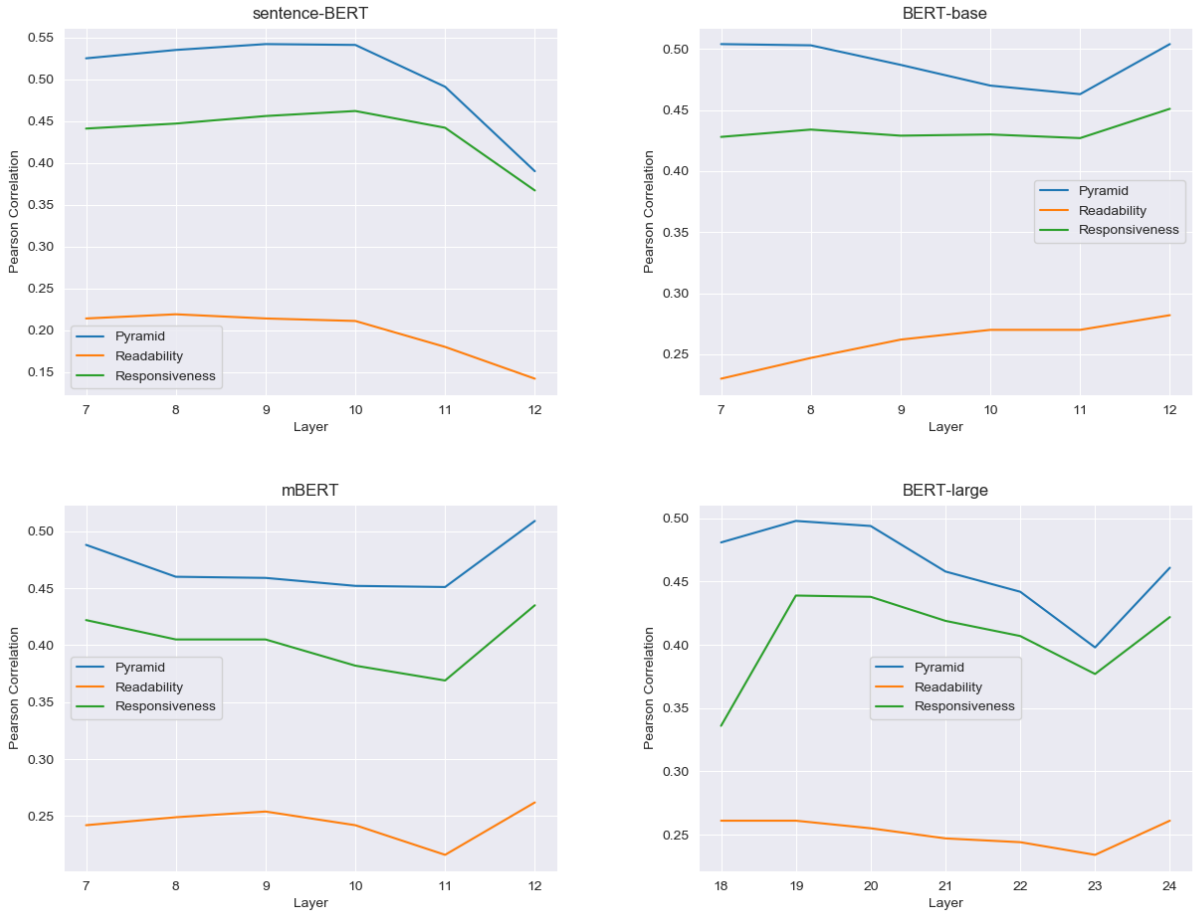


Figure 7.15: Layerwise Pearson correlation of TSAuBERT metrics and human evaluation metrics

The patterns in the figures show that for both the `TSAuBERT` versions using the BERT-base and mBERT models, there is a downward trend from layers 7-11 with a sharp uptick at the 12th layer. For the version using the BERT-large we see a similar sharp uptick from the second-to-last to the last layer, however, the best performing representations are derived from the 19th layer.

Interestingly, we see that for the `TSAuBERT` version based on the sentence-BERT pre-trained model, the trend is somewhat opposite than those of the other models. There is a sharp drop-off in the correlation scores for all three human evaluations after the 10th layer of the sentence-BERT model. Given that the sentence-BERT model has been fine-tuned on a sentence-level semantic similarity task [Reimers and Gurevych, 2019] when the embedding vectors flow towards the final output layer of the sentence-BERT model, the embeddings are likely encoded with information relevant to the fine-tuning task as suggested by [Xiao, 2018]. This may bias the embedding vectors towards that task, causing the drop in performance when used

in TSAuBERT.

An additional finding of the layerwise probing experiments is the 10th layer of the sentence-BERT based TSAuBERT version seems to be overall better performing than the 8th layer found during the random search. We use the 10th layer for the $TSAuBERT_{sent}$ version for the remaining experiments.

In general, we see that for different BERT models, different layers provides the best performance when used in the TSAuBERT framework. The results from the layerwise probing analysis reaffirmed, for 3 out of 4 of the models, the findings of the random search.

7.3.2.2 Increasing n-gram size

We test the effect of increasing n-gram size on the performance of different versions of TSAuBERT. The intuition of doing follows that by combining the embeddings vectors, we derive, akin to increasing the size of N for ROUGE-N. See figure 7.16

There is a clear tendency in all the below figures showing decreased correlation when increasing the n-gram size. We also see a general tendency towards a flattening of the curve as the n-gram size increase above 3. For the *readability* human metric, the trend is overall more flat and, in some cases, slightly increasing, however, the differences are minute. Thus, there is evidence to support the claim that increasing the n-gram size and pooling together embedding vectors does not improve the representation of language for comparison with cosine similarity. As previously mentioned, this is likely a result of the contextual nature of the embedding vectors derived from BERT models, i.e., there is already information regarding the surrounding words encoded into the embeddings.

The results suggest that further pooling does not provide a richer contextual representation of the input text. An additional factor influencing this is the choice of pooling technique we employ. As we are pooling together embedding vectors by taking their mean, we might be averaging away distinguishing information from the individual embeddings. Further study of the performance differences between different pooling strategies is needed in order to assert this.
Overall, when comparing the results of the probing analysis with the best parameter combinations found during the random search, we see that in only one instance, we were able to find a better parameter, being the 10th rather than the 8th layer for the sentence-BERT based TSAuBERT version. This finding speaks to the general strength of random search as a search strategy, even with a reduced search space as explained in 6.2.1.1.

From the above experiments, we derive the final four versions of TSAuBERT based on the best parameter sets found. We use these versions as the basis for the comparison with the ROUGE metrics. The table below provides an overview of the final

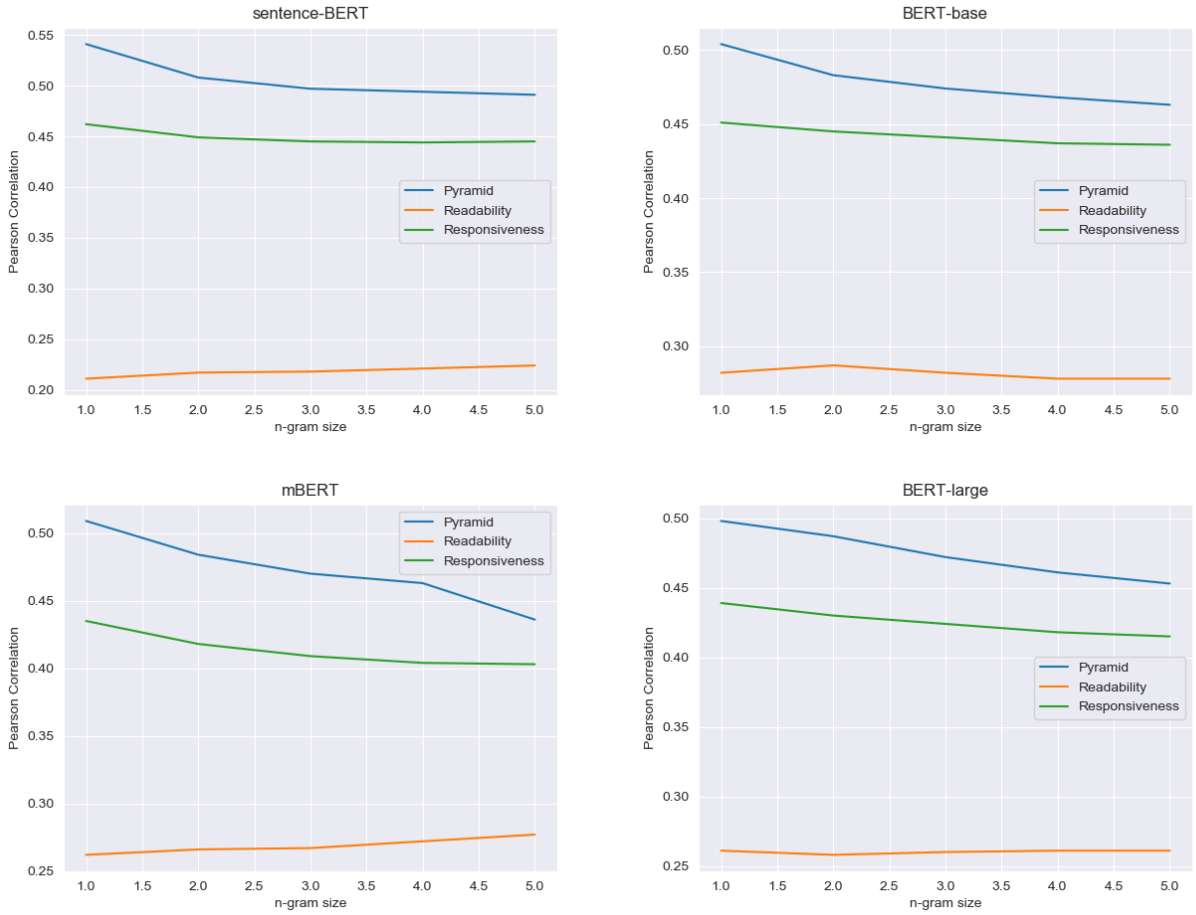


Figure 7.16: ngram Pearson correlation of TSAuBERT metrics and human evaluation metrics

versions.

	Parameters				
Metric	Model	Layer	Wordpiece	n-gram	scoring
$TSAuBERT_{base}$	bert-base-uncased	12	no	1	argmax
$TSAuBERT_{large}$	bert-large-uncased	19	no	1	argmax
$TSAuBERT_{mult}$	mbert-base-uncased	12	no	1	argmax
$TSAuBERT_{sent}$	sentence-bert	10	no	1	argmax

Table 7.6: Final TSAuBERT metrics

7.3.3 TSAuBERT vs. ROUGE

We compare the correlation with the human evaluation metrics of the best performing versions of the `TSAuBERT` metrics (table 7.6) and the ROUGE scores for all the candidate summaries in the TAC. The table below shows all correlation scores, with

the highest scores in bold.

	Pearson			Spearman			Kendall		
Metric	Pyramid	Readability	Responsiveness	Pyramid	Readability	Responsiveness	Pyramid	Readability	Responsiveness
$TSAuBERT_{base}$	0.504	**0.282**	0.451	0.513	**0.254**	0.451	0.375	**0.192**	0.348
$TSAuBERT_{large}$	0.498	0.261	0.439	0.502	0.236	0.440	0.367	0.178	0.339
$TSAuBERT_{mult}$	0.509	0.262	0.435	**0.518**	0.233	0.441	0.378	0.176	0.339
$TSAuBERT_{sent}$	**0.541**	0.211	**0.537**	0.462	0.208	**0.467**	**0.394**	0.156	**0.360**
ROUGE-1	0.525	0.164	0.420	0.503	0.157	0.418	0.368	0.119	0.322
ROUGE-2	0.482	0.157	0.384	0.454	0.162	0.390	0.329	0.122	0.299
ROUGE-L	0.460	0.154	0.386	0.444	0.157	0.391	0.321	0.118	0.301

Table 7.7: `TSAuBERT` F1 & ROUGE F1 scores - Correlation scores with human evaluations

The Pearson's correlation coefficient is used to measure the general correlation between two variables. The assumptions are that both variables are normally distributed and that their relationship is linear. The results of the Pearson's correlations show that overall `TSAuBERT` correlates better with the human metrics than ROUGE. The only instance where a ROUGE metric is better correlated than some of the `TSAuBERT` metrics is the *Pyramid* correlation of ROUGE-1. For both *Readability* and *Responsiveness* `TSAuBERT` shows, comparatively, larger improvements in correlations over ROUGE. This indicates that `TSAuBERT` may, in fact, be better at assessing the linguistic quality of the generated language compared to ROUGE. The caveat to this is that generally, the correlation scores are quite low. Although there are positive relationships established, these are not significantly large. That being said, the fact that `TSAuBERT` generally outperforms ROUGE is a positive result, indicating its use as a summarization evaluation metric.

The Spearman's correlation coefficient measures the correlation between the ranks of two variables and is useful for measuring the correlations where the relationship might not be linear in nature. The results of the Spearman's correlations show similar tendencies as those seen in the Pearson's correlations. `TSAuBERT` generally correlates better than ROUGE, with a larger difference for *Readability* and *Responsiveness*. The only notable difference is lower scores for the sentence-BERT based version.

Lastly, the Kendall's correlation coefficient measures correlations between ranks of the variables similar to Spearman's. Although the correlation values are generally lower than for the other two coefficients, the same patterns emerge. Thus, reaffirming the findings described in the paragraphs above.

Based on the above analysis, we argue that the performance of the `TSAuBERT` framework as a summarization evaluation metric shows promise. We base this on the comparatively good correlation scores reported in table 7.7, and the fact that we demonstrate a clear improvement over the ROUGE metrics in terms of correlation with *Readability* and *Responsiveness* and that we see the same patterns of correlation across three distinct metrics. Although the correlation values do not indicate a

significant positive correlation, we argue that the general improvements compared to ROUGE, suggests that `TSAuBERT` is, in fact, a useful addition to the summarization evaluation protocol.

Although the results may indicate that the usability of BERT embedding vectors for unsupervised similarity assessment is limited, the fact that the correlation scores are generally on par or better than those for ROUGE is encouraging. It should inspire further research into how to improve `TSAuBERT` for similarity assessment tasks. This includes investigating the fine-tuning of the underlying BERT models on the summarization evaluation task, as well as testing the limits of `TSAuBERT` for textual similarity assessment, providing a deeper understanding of the utility of BERT embedding vectors to measure the semantic similarity between sequences of text.

7.4 Key findings

We present here an overview of the key findings from our analysis of the results, providing concise answers to the research questions presented in section 1.2
The key findings of the analysis are:

- **General results**
  The summaries generated by most of our summarization models were able to beat all baseline ROUGE scores for all datasets, thereby achieving new state of the art in Automatic text summarization for Danish and providing a strong baseline of scores for future Danish summarization systems to beat.

- **Reference summaries**
  Using article subheadings as reference summaries can negatively impact how candidate summaries are evaluated wrt. ROUGE scores. We argue that this explains the comparatively lower ROUGE scores for all summarization models developed on the **Dan-Abs**, **Dan-Mix** and **Tv2** datasets.

- **Mono-lingual Vs. Multi-lingual**
  The DaBERT based summarization models using *extractive* summarization perform equally well or even outperforms the *extractive* mBERT summarization models, whereas the mBERT summarization models performs significantly better than DaBERT based ones for all *abstractive* and *mixed* summarization strategies. We hypothesize that this is likely a result of the DaBERT based models produces longer summaries and, in general, has trouble generating language as a cause of sub-par pre-training. This finding suggests that when using BERT for more complex NLP tasks such as language generation, the quality of the data for the pre-training of the BERT model has a significant impact. This is less so for simpler classification type tasks.

- **Positional Bias**
  We found that our extractive summarization models generally have a bias towards extracting the first three sentences of the source document. The models

were not able to adapt when applied to a dataset that did not have these domain-specific biases.

- **Measuring abstractiveness**
  The summarization models based on mBERT and fine-tuned on the DaN-Full dataset produce very extractive summaries, most likely due to the number of extractive reference summaries in the dataset. 71% and 66% of the sentences generated by the mBERT-Full models were also found in the source text. This was only 7% and 6% of the sentences for the mBERT-mix models. This indicates the when the reference summaries are more extractive in nature, abstractive and mixed summarization models generate sequences that often follow the wording of the source text.

Summarization evaluation metric

- **Results** Based on the random search results, a scoring approach that involved argmax, unigrams, no word pooling, and a combination of top and upper-middle layers proved to produce the best results. Surprisingly mBERT performs better on pyramid and responsiveness compared to other BERT base models. However, the changes are not significant enough to favor one model over another.
- **TSAuBERT vs ROUGE**
  We find that when comparing the correlation with human evaluations, different versions of `TSAuBERT` outperforms ROUGE in terms of correlation with human assessments, especially in correlations with *Readability* and *Responsiveness*. We highlight that even though the correlation is not larger for neither metric of the two, the fact that we still think that `TSAuBERT`, by outperforming ROUGE (although in some cases only slightly), shows promise as an addition to the summarization evaluation protocol.
- **Summarization evaluation metrics in general**
  Developing good metrics for evaluating the quality is a complex task. Further research is needed in order to cultivate a better summarization evaluation protocol.

8. Discussion

In this section, we discuss the implications of using BERT for summarization as well as some of the structural challenges with the summarization datasets. We further evaluate the design of `TSAuBERT` and the challenging task of summarization evaluation. The section concludes with reflections on further research.

8.1 Using BERT for summarization

Our experiments demonstrate, using the approaches developed by [Liu and Lapata, 2019], the ability to fine-tune pre-trained BERT models for summarization on a lower resource language, but also highlight some issues in using the BERT architecture on this task. Most prominently, our results showed that the DaBERT model pre-trained only on Danish data struggled in generating summaries and often inserted irrelevant tokens. Given that we did not observe these issues in the mBERT model, the struggles of the DaBERT abstractive summarization models is most likely due to the pre-training of DaBERT. mBERT is pre-trained using Wikipedia articles from 104 different languages. This means that even though mBERT is pre-trained on less Danish text, the overall corpus is much larger and contains more structured data than that of DaBERT. This likely improve the overall language understanding of the model. We hypothesize that the model achieves better results by also being trained on languages very similar to Danish like Swedish, Norwegian, and German. These languages are not the same as Danish but share many words and grammatical structures, and therefore learning from those can be beneficial to understand Danish better. The fact that mBERT performs better than DaBERT is also a testament to the quality of the pre-training corpus playing a large role.
The implication of this is that when using BERT models in language generation tasks, the data used for pre-training the model significantly affects down-stream performance. Even more so than for simpler classification style tasks such as extractive summarization. This presents a challenge for lower resource languages such as Danish, as suitable pre-training data is less available. In order to improve the performance on Danish language generation tasks using BERT, including abstractive summarization, a better pre-trained Danish BERT model is needed.

A further reason why our DaBERT based abstractive summarization models performed worse than the extractive ones lies in the pre-training tasks of BERT (described in section 2.6.2). The fine-tuning strategy for our extractive summarization models is much simpler; it is a binary classification task that involves predicting whether a sentence is a summary sentence or not. On the other hand, the abstractive task involves predicting every word in the output summary and requires a much deeper understanding of language. The extractive fine-tuning task is also somewhat similar to the pre-training task of next sentence prediction. Recent research has indicated that having a pre-training task that closely resembles the fine-tuning task often leads to significantly improved results.

As neither of the pre-training tasks for BERT are directed at improving auto-regressive language generation, BERT as a framework is perhaps less suited for the abstractive summarization task compared to other pre-trained transformer-based models such as BART [Lewis et al., 2019] or PEGASUS [Zhang et al., 2019a]. That being said, our results from the mBERT based abstractive summarization models do show that the BERT architecture can be fine-tuned to language generation tasks with reasonable degrees of success, even though these abstractive summarization models tend to stick closely to the wordings of the source text.

8.2 Danish summarization datasets

Both the DaNewsroom and TV2 datasets often rely on subheadings of articles for reference summaries, which, as our results showed, does not always lead to optimal outcomes. The primary purpose of an article subheading is not to summarize the text body of the article but rather expand on the title of an article. Hence, the use of these for reference summaries in the Tv2 dataset and several of the instances in the DaNewsroom dataset(s) introduces a central challenge when evaluating the quality of a candidate summary. As highlighted in section 7.2.1, when there are larger differences between the sizes of the reference and candidate summaries, the final ROUGE scores for the candidate summary tends to drop. This occurs regardless of the quality of the candidate summary wrt. summarizing the source text, as the F1 measure used in ROUGE scores penalizes length differences between candidate and reference. This indicates that as reference summaries vary greatly in quality, relying solely on these to evaluate the quality of a candidate is somewhat problematic. The current protocol for evaluating summarization systems cannot incorporate any other information than the reference summary. Thus with the varying qualities of reference summaries in datasets, there is a need to develop evaluation metrics that also take into account the source text when evaluating a candidate summary. This seems a particularly interesting avenue for future research.

It also is important to point out that DaNewsroom is not a poor dataset. In fact, it is quite good, and we could not have achieved the results reported in this thesis had it not been for the work of [Varab and Schluter, 2020] in generating it. However, the way it has been created introduces some structural challenges that have to be acknowledged. DaNewsroom is a start and can help expand the research within Danish text summarization and Danish NLP in general. However, the ultimate goal for danish summarization should be to create a danish dataset with human reference summaries that have specifically been produced to summarize articles. This task is both expensive and time-consuming, but will likely enable further advance of the field of automatic text summarization for Danish.

8.3 Evaluation metric datasets

The general lack of available datasets for developing summarization evaluation metrics further adds to the task's complexity. While the TAC 2011 AESOP dataset used in this thesis for assessing the quality of `TSAuBERT` is the most popular dataset, it is not without flaws. Firstly, the dataset is outdated in terms of the automatic summarizers used to produce the candidate summaries. Secondly, it is extremely difficult to get a hold of (you need to have been signed up for the 2011 TAC conference, of which our supervisor was) and is presented in a distributed format difficult to work with.
There is a need for the creation of a better dataset designed for the development of more sophisticated evaluation metrics. Furthermore, in order to facilitate better comparisons of potential summarization evaluation metrics' performance, there is a need for the establishment of a more transparent process on how to assess the quality of a metric.

8.4 TSAuBERT as an evaluation metric

In developing the concept for `TSAuBERT`, a key assumption is that the embedding vectors generated by various BERT models are, in fact, useful representations of words for comparing semantic similarity. Although previous work has found evidence to support this [Zhang* et al., 2020], others have highlighted the limited capabilities of using BERT embeddings in unsupervised similarity tasks [Reimers and Gurevych, 2019]. Thus there seem to be conflicting results in the literature, dependent on how the semantic similarity task is phrased. Our results show that, in fact, the BERT embeddings can be used to measure the quality of generated summaries. Thus indicating at least some utility in BERT embeddings in measuring textual similarities. Although `TSAuBERT` versions generally perform as well and, in most cases, slightly better than ROUGE in terms of correlation with human metrics, the correlation scores reported are not significantly large. To further improve these scores, one would have to fine-tune a BERT model on semantic similarity tasks and used this BERT model in the `TSAuBERT` framework.

While the fine-tuning of the BERT models would likely yield a better evaluation metric, improve correlation results, the cost of doing this is decreased flexibility in terms of languages applied on. As previously mentioned, one of the key benefits of ROUGE is that it is agnostic towards the language it is used on, making it easy to use it and compare performance on different languages. Having to fine-tune a monolingual BERT model in order for `TSAuBERT` to perform optimally on a different language, reduces the flexibility of the framework significantly. Furthermore, for low resource languages, the availability of data for fine-tuning BERT models for semantic text similarity is likely scarce.

This ties together with the findings from the Danish summarization experiments, as the quality of pre-trained BERT models for low resource languages is a challenging

task in and of itself. This exposes a weakness in the TSAuBERT framework in that in order for it to function well as an evaluation metric, the BERT model used needs to be well pre-trained. For English, this is not a problem; however, it is a challenge for smaller languages like Danish. However, the general popularity of the BERT framework in the NLP community means that several mono-lingual BERT models have been pre-trained, and new and improved models continue to be developed. Thus there is potential for implementing and analyzing the performance of TSAuBERT for several languages in future research.

Another of the findings from the summarization experiments relevant to future developments on TSAuBERT are the results related to the short reference summaries. As we argue for the development of summarization evaluation metrics that rely not only on a reference summary to assess the quality of a candidate, TSAuBERT would need to be updated in order to accommodate the inclusion of the source text in the evaluation process.
The task of conceptualizing and developing a new approach towards summary-evaluation is not trivial. There is no objectively "best" summary for a given source text, and the quality of language can be measured in a multitude of manners. The current reliance on ROUGE metrics to define state of the art for summarization systems is problematic, as the results of several experiments in this thesis show. Although the ROUGE metrics certainly have a place, they only assess summary quality in terms of lexical overlap. With the introduction of TSAuBERT we present an evaluation metric that attempts to address this by measuring semantic similarity on a level deeper than the lexical.

8.5 Future research

With the findings from this thesis, there are several interesting avenues for future research to explore. It would be relevant to further optimize the hyper-parameters of our summarization models, testing the limits if the PreSumm summarization framework from [Liu and Lapata, 2019] for summarization on a low resource language. For improved results on Danish abstractive summarization, the pre-training of a better Danish BERT model is an important step. The development of this model would not only improve results for summarization but likely several Danish NLP tasks. Furthermore, given the procurement of better Danish pre-training data, it would also be quite interesting to pre-train and fine-tune a Danish BART model for language generation, as this framework has shown impressive results for abstractive summarization on English [Lewis et al., 2019]. In terms of future research in TSAuBERT it seems pertinent to fine-tune a BERT model on the summarization evaluation task to see if further improvements in correlations occur. One could also explore the pooling strategy we used to combine vectors and see for other pooling strategies that would yield better results. It would also be relevant to explore the development of a new and better dataset for assessing the quality of summarization evaluation metrics. Aiding the development in this field would also likely improve the development of new and better automatic summarization models.

9. Conclusion

The aim of this thesis was to develop and evaluate the performance of Danish summarization models using the BERT architecture. Furthermore, the aim was to investigate the performance difference for Danish summarization when using a monolingual Danish BERT model (DaBERT) vs. a multilingual BERT model (mBERT). Using the framework for fine-tuning BERT for summarization developed by [Liu and Lapata, 2019] we fine-tuned 27 Danish summarization models on different datasets. Our results showed that our summarization models consistently beat the baseline models in terms of ROUGE F1 scores, establishing a new state of the art for the Danish datasets.

The results further showed that for the more simple *extractive* summarization task, the different summarization models based on DaBERT and mBERT performed equally well. However, this was not the case for the *abstractive* and *mixed* summarization models, where those based on mBERT performed significantly better than those based on DaBERT. These findings suggest that the relatively poor quality of the pre-training data used for DaBERT causes the model to struggle when fine-tuned for language generation. Thus emphasizing the importance of pre-training data quality for more complex language generation tasks, such as abstractive summarization. We suggest that future research in summarization for Danish focus on the development of a better pre-trained Danish BERT model in order to improve results, especially for abstractive summarization.

Our experiments further showed that the pervasive use of news-article sub-headings for reference summaries in some of the Danish summarization datasets has a negative impact on the evaluation scores of generated summaries. This exposes a weakness in the ROUGE metrics currently used to define the state of the art for summarization, in that they rely entirely on reference summaries for evaluating the quality of generated summaries. We, therefore, emphasize the importance of the development of evaluation metrics that incorporate more than the reference summary, providing an interesting avenue for further research.

Additionally, the aim of the thesis was to develop and test a summarization evaluation metric capable of measuring textual similarity on a semantic level. In doing so, we developed a metric that utilizes the *cosine similarity* between contextually dependent embedding vectors derived from BERT models to measure semantic similarity between two words. We name this framework **T**extual **S**imilarity **A**ssessment using **B**ERT or `TSAuBERT` for short. Our analysis of `TSAuBERT` as a summarization evaluation metric showed that it correlated as well and, in many instances, better than the ROUGE metrics, with human evaluations. Thus indicating that `TSAuBERT` is, in fact, a usable addition to the current summarization evaluation protocol. However, our results also indicated that neither ROUGE nor `TSAuBERT` correlates significantly well with human assessments, with the highest Pearson correlation being **0.542**. In

order to improve upon these results we suggest that future research focus on fine-tuning a BERT model for the summarization evaluation task, and focus on building a better and accessible dataset for developing and testing summarization evaluation metrics.

In general, our findings demonstrate the usefulness of fine-tuning BERT models for summarization on low resource languages. The results presented in this thesis further showed that monolingual BERT models pre-trained on sub-optimal data does comparatively well when fine-tuned for simpler classification tasks, but struggle when fine-tuned for more complex language generation tasks. We argue that the development of a better pre-trained Danish BERT model would improve future efforts for not only Danish summarization, but Danish NLP in general.

Bibliography

[Aggarwal, 2018] Aggarwal, C. C. (2018). *Neural Networks and Deep Learning*. Springer, Cham.

[Alammar, 2018a] Alammar, J. (2018a). The illustrated bert, elmo, and co.

[Alammar, 2018b] Alammar, J. (2018b). The illustrated transformer.

[Allahyari et al., 2017] Allahyari, M., Pouriyeh, S. A., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B., and Kochut, K. (2017). Text summarization techniques: A brief survey. *CoRR*, abs/1707.02268.

[Bergstra et al., 2011] Bergstra, J., Bardenet, R., Kégl, B., and Bengio, Y. (2011). Algorithms for hyper-parameter optimization.

[Bergstra and Bengio, 2012] Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305.

[BotXO, 2019] BotXO (2019). Botxo has trained the most advanced danish bert model to date.

[Chauhan, 2019] Chauhan, K. (2019). Unsupervised text summarization using sentence embeddings.

[Chopra et al., 2016] Chopra, S., Auli, M., and Rush, A. M. (2016). Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, San Diego, California. Association for Computational Linguistics.

[Conroy and O'leary, 2001] Conroy, J. M. and O'leary, D. P. (2001). Text summarization via hidden markov models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, page 406–407, New York, NY, USA. Association for Computing Machinery.

[Deng and Liu, 2018] Deng, L. and Liu, Y. (2018). *Deep learning in natural language processing*. Springer Nature.

[Devlin, 2019] Devlin, J. (2019). Multilingual bert.

[Devlin et al., 2019] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). *Proceedings of the 2019 Conference of the North*.

[Edmundson, 1969] Edmundson, H. P. (1969). New methods in automatic extracting. *J. ACM*, 16(2):264–285.

[Egonmwan and Chali, 2019] Egonmwan, E. and Chali, Y. (2019). Transformer-based model for single documents neural summarization. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 70–79, Hong Kong. Association for Computational Linguistics.

[Ettinger, 2019] Ettinger, A. (2019). What bert is not: Lessons from a new suite of psycholinguistic diagnostics for language models.

[Freitag and Al-Onaizan, 2017] Freitag, M. and Al-Onaizan, Y. (2017). Beam search strategies for neural machine translation. *Proceedings of the First Workshop on Neural Machine Translation*.

[Gambhir and Gupta, 2016] Gambhir, M. and Gupta, V. (2016). Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, 47.

[Ganesan, 2015] Ganesan, K. (2015). Rouge 2.0: Updated and improved measures for evaluation of summarization tasks.

[Gehrmann et al., 2018] Gehrmann, S., Deng, Y., and Rush, A. M. (2018). Bottom-up abstractive summarization.

[Grusky et al., 2018] Grusky, M., Naaman, M., and Artzi, Y. (2018). Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies.

[Hennig et al., 2010] Hennig, L., De Luca, E. W., and Albayrak, S. (2010). Learning summary content units with topic modeling. In *Coling 2010: Posters*, pages 391–399, Beijing, China. Coling 2010 Organizing Committee.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[i2tutorials, 2019] i2tutorials (2019). What is the difference between bidirectional rnn and rnn?

[Inderjeet, 2009] Inderjeet, M. (2009). Summarization evaluation: an overview.

[Informatik et al., 2003] Informatik, F., Bengio, Y., Frasconi, P., and Schmidhuber, J. (2003). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. *A Field Guide to Dynamical Recurrent Neural Networks*.

[Kedzie et al., 2018] Kedzie, C., McKeown, K., and Daumé III, H. (2018). Content selection in deep learning models of summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1818–1828, Brussels, Belgium. Association for Computational Linguistics.

[Khan et al., 2016] Khan, A., Salim, N., and Farman, H. (2016). Clustered genetic semantic graph approach for multi-document abstractive summarization. In *2016 International Conference on Intelligent Systems Engineering (ICISE)*, pages 63–70.

[Kirkedal et al., 2019] Kirkedal, A., Plank, B., Derczynski, L., and Schluter, N. (2019). The lacunae of Danish natural language processing. In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, pages 356–362, Turku, Finland. Linköping University Electronic Press.

[Kryscinski et al., 2019] Kryscinski, W., Keskar, N. S., McCann, B., Xiong, C., and Socher, R. (2019). Neural text summarization: A critical evaluation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 540–551, Hong Kong, China. Association for Computational Linguistics.

[Kupiec et al., 1995] Kupiec, J., Pedersen, J., and Chen, F. (1995). A trainable document summarizer. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 68–73.

[Lewis et al., 2019] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

[Li, 2013] Li, B. (2013). Distance weighted cosine similarity measure for text classification.

[Liddy, 2001] Liddy, E. D. (2001). Natural language processing.

[Lin, 2004] Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

[Lin and Hovy, 2000] Lin, C.-Y. and Hovy, E. (2000). The automated acquisition of topic signatures for text summarization. In *COLING 2000 Volume 1: The 18th International Conference on Computational Linguistics*.

[Lin and Hovy, 2002] Lin, C.-Y. and Hovy, E. (2002). Manual and automatic evaluation of summaries. Citeseer.

[Lipton, 2015] Lipton, Z. C. (2015). A critical review of recurrent neural networks for sequence learning. *CoRR*, abs/1506.00019.

[Liu and Liu, 2008] Liu, F. and Liu, Y. (2008). Correlation between ROUGE and human evaluation of extractive meeting summaries. In *Proceedings of ACL-08: HLT, Short Papers*, pages 201–204, Columbus, Ohio. Association for Computational Linguistics.

[Liu and Lapata, 2019] Liu, Y. and Lapata, M. (2019). Text summarization with pretrained encoders.

[Lloret et al., 2017] Lloret, E., Plaza, L., and Aker, A. (2017). The challenging task of summary evaluation: an overview. *Language Resources and Evaluation*, 52.

[Luhn, 1958] Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165.

[Mikolov et al., 2013a] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space.

[Mikolov et al., 2013b] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality.

[Nallapati et al., 2016] Nallapati, R., Zhai, F., and Zhou, B. (2016). Summarunner: A recurrent neural network based sequence model for extractive summarization of documents.

[Nenkova and McKeown, 2011] Nenkova, A. and McKeown, K. (2011). *Automatic summarization*. Now Publishers Inc.

[Nenkova et al., 2007] Nenkova, A., Passonneau, R., and McKeown, K. (2007). The pyramid method: Incorporating human content selection variation in summarization evaluation. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(2):4–es.

[Ng and Abrecht, 2015] Ng, J. and Abrecht, V. (2015). Better summarization evaluation with word embeddings for ROUGE. *CoRR*, abs/1508.06034.

[Olah, 2015] Olah, C. (2015). Understanding lstm networks.

[Peyrard, 2019] Peyrard, M. (2019). Studying summarization evaluation metrics in the appropriate scoring range. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5093–5100, Florence, Italy. Association for Computational Linguistics.

[Pouliakis et al., 2016] Pouliakis, A., Karakitsou, E., Margari, N., Bountris, P., Haritou, M., Panayiotides, J., Koutsouris, D., and Karakitsos, P. (2016). Artificial neural networks as decision support tools in cytopathology: Past present and future. *Biomed Eng Comput Biol*, 2016:1–18.

[Reimers and Gurevych, 2019] Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

[Russell and Norvig, 2009] Russell, S. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition.

[See et al., 2017] See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks.

[Sellam et al., 2020] Sellam, T., Das, D., and Parikh, A. (2020). BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

[ShafieiBavani et al., 2018] ShafieiBavani, E., Ebrahimi, M., Wong, R., and Chen, F. (2018). A graph-theoretic summary evaluation for ROUGE. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 762–767, Brussels, Belgium. Association for Computational Linguistics.

[Strømberg-Derczynski et al., 2020] Strømberg-Derczynski, L., Baglini, R., Christiansen, M. H., Ciosici, M. R., Dalsgaard, J. A., Fusaroli, R., Henrichsen, P. J., Hvingelby, R., Kirkedal, A., Kjeldsen, A. S., Ladefoged, C., Nielsen, F. Å., Petersen, M. L., Rystrøm, J. H., and Varab, D. (2020). The danish gigaword project. *CoRR*, abs/2005.03521.

[Sun et al., 2019] Sun, C., Qiu, X., Xu, Y., and Huang, X. (2019). How to fine-tune bert for text classification? In Sun, M., Huang, X., Ji, H., Liu, Z., and Liu, Y., editors, *Chinese Computational Linguistics*, pages 194–206, Cham. Springer International Publishing.

[TAC, 2017] TAC (2017). Tac 2011 aesop task guidelines.

[Thongtan and Phienthrakul, 2019] Thongtan, T. and Phienthrakul, T. (2019). Sentiment classification using document embeddings trained with cosine similarity. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 407–414, Florence, Italy. Association for Computational Linguistics.

[Trinh et al., 2018] Trinh, T. H., Dai, A. M., Luong, T., and Le, Q. V. (2018). Learning Longer-term Dependencies in RNNs with Auxiliary Losses. In *Proceedings of the 35th International Conference on Machine Learning*, pages 4972–4981. PMLR.

[Varab and Schluter, 2020] Varab, D. and Schluter, N. (2020). DaNewsroom: A large-scale Danish summarisation dataset. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 6731–6739, Marseille, France. European Language Resources Association.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

[Wolf et al., 2019] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. (2019). Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

[Wong et al., 2008] Wong, K.-F., Wu, M., and Li, W. (2008). Extractive summarization using supervised and semi-supervised learning. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 985–992, Manchester, UK. Coling 2008 Organizing Committee.

[Wu et al., 2016] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

[Xiao, 2018] Xiao, H. (2018). bert-as-service. `https://github.com/hanxiao/bert-as-service`.

[Yan et al., 2020] Yan, Y., Qi, W., Gong, Y., Liu, D., Duan, N., Chen, J., Zhang, R., and Zhou, M. (2020). Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training.

[Zhang et al., 2019a] Zhang, J., Zhao, Y., Saleh, M., and Liu, P. J. (2019a). Pegasus: Pre-training with extracted gap-sentences for abstractive summarization.

[Zhang* et al., 2020] Zhang*, T., Kishore*, V., Wu*, F., Weinberger, K. Q., and Artzi, Y. (2020). Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

[Zhang et al., 2019b] Zhang, X., Wei, F., and Zhou, M. (2019b). HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5059–5069, Florence, Italy. Association for Computational Linguistics.

[Zhong et al., 2019] Zhong, M., Liu, P., Wang, D., Qiu, X., and Huang, X. (2019). Searching for effective neural extractive summarization: What works and what's next. *CoRR*, abs/1907.03491.

[Zhou et al., 2018] Zhou, Q., Yang, N., Wei, F., Huang, S., Zhou, M., and Zhao, T. (2018). Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–663, Melbourne, Australia. Association for Computational Linguistics.