Linköping University | Department of Computer and Information Science
Bachelor's thesis, 18 ECTS | Cognitive Science
2021 | LIU-IDA/KOGVET-G--21/007--SE

Building high-quality datasets for abstractive text summarization

– A filtering-based method applied on Swedish news articles

Julius Monsen

Supervisor : Arne Jönsson
Examiner : Erik Marsja

LINKÖPING UNIVERSITY

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Abstract

With an increasing amount of information on the internet, automatic text summarization could potentially make content more readily available for a larger variety of people. Training and evaluating text summarization models require datasets of sufficient size and quality. Today, most such datasets are in English, and for minor languages such as Swedish, it is not easy to obtain corresponding datasets with handwritten summaries. This thesis proposes methods for compiling high-quality datasets suitable for abstractive summarization from a large amount of noisy data through characterization and filtering. The data used consists of Swedish news articles and their preambles which are here used as summaries. Different filtering techniques are applied, yielding five different datasets. Furthermore, summarization models are implemented by warm-starting an encoder-decoder model with BERT checkpoints and fine-tuning it on the different datasets. The fine-tuned models are evaluated with ROUGE metrics and BERTScore. All models achieve significantly better results when evaluated on filtered test data than when evaluated on unfiltered test data. Moreover, models trained on the most filtered dataset with the smallest size achieves the best results on the filtered test data. The trade-off between dataset size and quality and other methodological implications of the data characterization, the filtering and the model implementation are discussed, leading to suggestions for future research.

Keywords: NLP, abstractive text summarization, dataset quality, encoder-decoder model, BERT

Acknowledgments

I want to thank everyone that has contributed to this thesis. First of all, I would like to thank DN (Dagens Nyheter) for making this work possible by providing the data. Then, a special thanks to my supervisor Arne Jönsson for guiding me along the way and providing knowledgeable insights and helpful feedback. I also want to thank everyone in my seminar group for valuable input and my family and close friends who have continuously supported me during this work.

Contents

Abstract iii

Acknowledgments iv

Contents v

List of Figures vii

List of Tables viii

1 Introduction 1
- 1.1 Abstractive summarization . . . 2
- 1.2 Aim . . . 3
- 1.3 Research questions . . . 3
- 1.4 Thesis outline . . . 3

2 Theory 4
- 2.1 Summarization datasets . . . 4
- 2.2 Language modeling . . . 5
  - 2.2.1 Text representation . . . 5
  - 2.2.2 Sequence-to-sequence . . . 6
  - 2.2.3 Attention . . . 7
- 2.3 The Transformer . . . 9
  - 2.3.1 Encoder . . . 9
  - 2.3.2 Decoder . . . 9
  - 2.3.3 Self-attention . . . 10
  - 2.3.4 Multi-head attention . . . 11
  - 2.3.5 Positional encoding . . . 11
- 2.4 BERT . . . 12
  - 2.4.1 Architecture . . . 12
  - 2.4.2 Input representation . . . 12
  - 2.4.3 Pre-training BERT . . . 13
  - 2.4.4 Sentence-BERT . . . 14
- 2.5 Warm-starting seq2seq models with BERT . . . 14
  - 2.5.1 Model architecture . . . 14
  - 2.5.2 Weight sharing . . . 15
  - 2.5.3 Fine-tuning for summarization . . . 15
- 2.6 Evaluation . . . 15
  - 2.6.1 ROUGE-measures . . . 16
  - 2.6.2 BERTScore . . . 16
  - 2.6.3 Cosine similarity . . . 18

3 Building summarization datasets 19

3.1 Compiling the DN data . . . 19
3.1.1 Data characterization . . . 19
3.1.2 Article categories . . . 20
3.2 Filtering . . . 21

4 Using the datasets for abstractive summarization 24
4.1 Model implementation . . . 24
4.1.1 The pre-trained model . . . 24
4.1.2 Preparing the data as model input . . . 24
4.1.3 Warm-starting the encoder-decoder model . . . 25
4.1.4 Fine-tuning the model . . . 25
4.2 Model evaluation . . . 26
4.2.1 BERTScore . . . 26
4.2.2 Evaluation results . . . 26
4.3 Model generated examples . . . 27

5 Discussion 30
5.1 Building summarization datasets . . . 30
5.2 Using the datasets for abstractive summarization . . . 31
5.2.1 Test data differences . . . 31
5.2.2 Model differences . . . 31
5.2.3 Measure differences . . . 32
5.3 Model implementation . . . 32
5.3.1 Model fine-tuning . . . 32
5.3.2 Evaluation measures . . . 32
5.4 Future research . . . 33
5.5 Ethical considerations . . . 33

6 Conclusion 34

Bibliography 35

Appendix A Low-quality summarization data 38

List of Figures

2.1 Encoder-decoder example 7
2.2 Attention example 8
2.3 Transformer architecture 10
2.4 Example of a sequence being processed to yield the input embeddings for BERT . . 13
2.5 BERTScore example 17

3.1 Data characteristics for the unfiltered DN data 21
3.2 Data characteristics for the filtered DN-sn subset 23

List of Tables

2.1 Statistics for the MLSUM dataset and the CNN/Daily Mail dataset 5

3.1 Statistics for the DN data and the CNN/Daily Mail dataset 20
3.2 Statistics for the filtered DN datasets 22

4.1 Training parameters for the models fine-tuned on the different datasets 25
4.2 Evaluation results for the models trained on the different datasets 26
4.3 Evaluation scores for Example 1 28
4.4 Evaluation scores for Example 2 28
4.5 Evaluation scores for Example 3 29

1 Introduction

In this digital age, vast amounts of information are produced on the internet every day. A significant proportion of this information is unstructured texts from various sources such as social media, encyclopedias and news sites. In principle, lengthy texts from many of these sources can be condensed into short summaries capturing the essential points. However, on a large scale in practice, manually summarizing texts is a time-consuming, costly and gruelling task. This is where automatic text summarization comes in as a valuable tool to provide the most important content of a given text in a comprehensible format, sensitive to the user's needs, in a fast and efficient way. Utilizing automatization in this way could have a tremendous positive impact as more information can be made accessible and easier to consume for everyone. For example, automatic text summarization could, together with other language technology tools, automatically adapt texts based on different needs and make certain texts more readily available to people with reading difficulties due to dyslexia, intellectual disabilities or other causes. Today, most easy-to-read texts are manually produced in a time-consuming and expensive process. The potential benefits for automatic summarization are thus promising, both cost-wise and from the perspective of readers who get access to a broader selection of texts. Currently, TextAd[1], a research project at Linköping University, aims to investigate these aspects and whether automatic adjustments of text, such as automatic text summarization, can facilitate reading comprehension for readers with certain difficulties.

There are two main types of automatic text summarization: extractive and abstractive [13]. Extractive summarization involves extracting the most relevant sentences from a given document and concatenating them into a summary. Abstractive summarization is about creating an intermediate semantic representation of the document from which novel sentences that capture the most salient points can be generated in a linguistically fluent manner. There are also hybrid approaches that combine both extractive and abstractive techniques. Although summarizing a text is relatively simple for humans, it is complex for computers to do. This is especially true for the abstractive summarization approach. It requires complex language abilities, including text understanding and production, and the ability to distinguish what information is most important in a given document. All these aspects are subject to the very active research field of natural language processing (NLP).

Most state-of-the-art text summarization systems today, both abstractive and extractive, are based on neural networks[2]. When training and evaluating neural network models, it is crucial to have datasets of sufficient size and quality so that the models can learn and generalize to unseen data. Existing summarization datasets vary in terms of size, from tens of thousands to millions of document-summary pairs. Regarding quality, a summary should be concise, objective and provide a clear and precise picture of the contents in the paired document. This facilitates learning and ensures that the model learns what it is supposed to. In general, it is also vital to have benchmark datasets that allow for comparison between different models.

In most natural language processing tasks, and so even in text summarization, most available datasets are in English, and thus most research efforts focus on the English language. One of the most widely used datasets for summarization is the CNN/Daily Mail dataset[21], which

[1] `https://liu.se/forskning/textad`

[2] See for instance `https://paperswithcode.com/task/text-summarization`

consists of news articles with manually-written highlights for each article. For text summarization in other minor languages availability of such benchmark datasets are limited. In Swedish, for instance, there is currently no established dataset corresponding to the CNN/Daily Mail dataset that can be used when training and evaluating abstractive summarization models. Many other languages face similar problems. One reason for this may be that it is very resource-demanding to write summaries to large sets of texts. It is much more convenient to gather and utilize already existing data. That is what is done in this thesis. The utilized data consists of Swedish news articles published in DN (Dagens Nyheter), Sweden's largest morning newspaper. However, for these articles, there are no written summaries, only preambles. Unfortunately, the preamble is not always a good summary of an article since its primary purpose is to capture the reader's interest. This may be done not only by highlighting the essential points of the article. It can also be the case that the central information of the story is written in the preamble but not directly mentioned again in the article. In Appendix A, examples of different low-quality article-summary[3] pairs are presented. In this thesis, a method for filtering out such low-quality article-summary pairs will be proposed as a way of improving model performance in the abstractive summarization task.

1.1 Abstractive summarization

Over the last few years, progress with Deep Neural Networks (DNNs) has contributed to a rising interest in abstractive text summarization among the research community and advanced natural language processing as a field. Language models trained on large amounts of text data, such as ELMO [24], ULMFit [11], BERT (Bidirectional Encoder Representations from Transformers) [8] and GPT [26] and its successors GPT-2 [27] and GPT-3 [6] has pioneered and enabled groundbreaking results in many NLP tasks, including abstractive summarization. A contributing factor to these developments has been the effectiveness and power that lies in transfer learning. It allows for language models to be pre-trained, unsupervised, on a data-rich task to develop general-purpose language abilities. Then they can be fine-tuned on a downstream task of interest, such as text summarization. This enables pre-trained model checkpoints to be saved, shared and used for various purposes without much effort or cost of training language models from scratch.

Rothe et al. [31] point out that improvements on benchmarks continue as an increasing amount of models building on language models such as BERT and GPT-2 emerges. Nevertheless, there has been a lack of efforts to explore the possibility of utilizing pre-trained models to warm-start sequence-to-sequence (seq2seq) models commonly used in abstractive text summarization. In their paper, a new seq2seq approach based on the Transformer [38] is developed that can make use of publicly available checkpoints from pre-trained models such as BERT. In experiments, they examined a range of model combinations and demonstrated state-of-the-art results on multiple tasks, including abstractive text summarization, comparable with state-of-the-art results from large models such as T5 [28], BART [15], PEGASUS [41] and ProphetNet [25] at a fraction of the training cost. Liu and Lapata [17] used a similar technique based on BERT, both for extractive and abstractive summarization.

The popularization of BERT has incentivized companies and institutions around the world to create language-specific pre-trained BERT models. This enables fine-tuning text summarization models in minor languages with the latest approaches. In 2020, The National Library of Sweden [19] released a pre-trained Swedish BERT model. There has nevertheless been little work within the Swedish NLP community to make use of this to tackle abstractive text summarization. As already mentioned, one reason for this may be the lack of large enough high-quality datasets that can be used to train and evaluate models.

[3] From now on, summary will be used to denote the preamble.

1.2 Aim

The purpose of this thesis is to explore methods for compiling high-quality datasets suited for abstractive text summarization by filtering out data of lower quality from a more considerable amount. This amount will consist of Swedish news articles published in and provided by DN. With the formulated problem in mind, these methods must distinguish between good article-summary pairs and those of lower quality. Here, the CNN/Daily Mail dataset with its characteristics will be used as a point of reference. Furthermore, the aim is to implement and fine-tune abstractive text summarization models based on the approach proposed by Rothe et al. [31]. Checkpoints from the Swedish pre-trained BERT model provided by the National Library of Sweden [19] will be used to warm-start a seq2seq model, which will then be fine-tuned and evaluated on the compiled datasets. By examining and evaluating these models fine-tuned on the different datasets, the effects of the applied filtering can be determined. The evaluation results from these models will also be compared to state-of-the-art results on the CNN/Daily Mail dataset.

1.3 Research questions

- How can datasets for abstractive text summarization with high-quality article-summary pairs be created from a large set of news articles with associated preambles, some of which are of low quality?
- How can a seq2seq model based on the method proposed by Rothe et al. [31] be implemented and realized for abstractive text summarization in Swedish?
- How do the implemented models, fine-tuned on the compiled datasets, perform with regards to commonly used metrics, compared to each other and model performance on the CNN/Daily Mail dataset?

1.4 Thesis outline

This thesis is structured in six chapters. In Chapter 2, the theoretical background and the essential concepts and techniques underlying abstractive summarization will be presented. Chapter 3 consists of a description of the filtering methods used to compile the datasets as well as the results of this filtering. Chapter 4 presents the methods used to implement the abstractive summarization models and the evaluation results for these models. Furthermore, the results, the used methodology, suggestions for future research and ethical considerations are discussed in detail in Chapter 5. A conclusion follows this chapter in Chapter 6.

2 | Theory

In this chapter, the theoretical foundations for abstractive text summarization will be presented. First, a short description of summarization datasets and useful data characteristics will be given. Then the general approach to model and represent language in computers will be described, followed by an explanation of the sequence-to-sequence framework and the concept of attention. The chapter continues by introducing the Transformer model with a presentation of the architecture and its most essential features. Subsequently, BERT will be explained in detail. After that, the process of warm-starting sequence-to-sequence models with BERT will be described. Finally, the theory behind evaluating summarization systems will be explained.

2.1 Summarization datasets

As mentioned in Chapter 1, the CNN/Daily Mail dataset [21] is one of the most used datasets in text summarization. The CNN/Daily Mail dataset consists of 311,971 news articles in English with 3-4 hand-written bullet-point highlights for each article. These highlights are usually concatenated into a single summary. Previous efforts have been made to address the problem of limited availability of datasets in languages other than English by creating datasets corresponding to the CNN/Daily Mail dataset. Scialom et al. [33] introduced the MLSUM dataset as the first large-scale MultiLingual SUMmarization dataset. They highlighted the possibility of this dataset effectively serving as a multilingual extension of the CNN/Daily Mail dataset, as it was similarly built from newspapers. MLSUM consists of more than 1.5 million article-summary pairs in five different languages - namely, French(FR), German(DE), Spanish(ES), Russian(RU) and Turkish(TR). The articles with their paired summaries, published between 2010 and 2019, were obtained from online newspapers covering various topics. Every language, except for Russian, is similar in terms of size to the CNN/Daily Mail dataset.

When building datasets for summarization, it is helpful to characterize the data with regards to its properties. The length of articles and summaries, vocabulary size, novelty between articles and summaries (a proxy for attractiveness) are essential characteristics often reported for summarization datasets. Scialom et al. [33] characterized the MLSUM dataset and compared it with the CNN/Daily Mail dataset regarding these and other properties. In Table 2.1 the characteristics of the MLSUM dataset is presented as it was presented in the paper. Article and summary lengths were computed in words, and Compression ratio was computed as the ratio between article and summary length. Novelty was the percentage of words in the summary that was not in the paired article. Total Vocabulary was the total number of different words and Occurring 10+, the total number of words occurring 10+ times.

Although MLSUM serves as a multilingual extension of the CNN/Daily Mail dataset, few attempts were made by Scialom et al. [33] to ensure that the summaries summarized the articles well, i.e. that the data maintained high quality, and that each language-specific dataset had similar characteristics as the CNN/Daily Mail dataset. The only filters that were applied were that article-summary pairs where the article was shorter than 50 words or where the summary was shorter than ten words were discarded. In this thesis, additional characteristics and filters are computed and applied to obtain similar characteristics as the CNN/Daily Mail dataset. This process is described in Chapter 3.

	FR	DE	ES	RU	TR	EN
Dataset size	424,763	242,982	290,645	27,063	273,617	311,971
Training set size	392,876	220,887	266,367	25,556	249,277	287,096
Mean article length	632.39	570.6	800.50	959.4	309.18	790.24
Mean summary length	29.5	30.36	20.71	14.57	22.88	55.56
Compression Ratio	21.4	18.8	38.7	65.8	13.5	14.2
Novelty (1-gram)	15.21	14.96	15.34	30.74	28.90	9.45
Total Vocabulary Size	1,245,987	1,721,322	1,257,920	649,304	1,419,228	875,572
Occurring 10+ times	233,253	240,202	229,033	115,144	248,714	184,095

Table 2.1: Statistics for each language in the MLSUM dataset and for the CNN/Daily Mail dataset(EN), as presented by Scialom et al. [33].

2.2 Language modeling

Abstractive summarization, as well as many other NLP tasks, hinges on finding good and efficient ways to model language. Language modelling is about predicting upcoming words[1] or sequences of words based on the prior context. The simplest language model is the n-gram[2] model, which relies on the assignment of probabilities to words or sequences of words [12]. A common way to estimate these probabilities is with maximum likelihood estimation (MLE), that is, take the number of occurrences of a given n-gram followed by a new word divided by the number of occurrences of the n-gram followed by any word in the given corpus. Although the n-gram model performs adequately in some respects and is computationally efficient, it has its shortcomings. Neural language models that use neural networks as probabilistic classifiers tackle the main problems with the n-gram model. Among other things, they can handle longer sequence histories and generalize over contexts of similar words, which allows for much better predictive accuracy.

2.2.1 Text representation

Language and, more specifically, the meaning of words can be represented in several ways. In the n-gram model and more traditional language models, the meaning of words is often represented by the words themself, i.e. the string of letters the word consists of [12]. Nowadays, the general approach to represent the semantic meaning of words is to use vector semantics.

The main idea behind vector semantics is to represent words as vectors in a multidimensional semantic space [12]. These vectors are based on the distributions of word neighbours. The underlying assumption is that frequently co-occurring words, close to each other in the multidimensional semantic space, share some semantic meaning. Consequently, the similarity between two words can easily be calculated as the angular distance between their respective vector (see Section 2.6.3). One approach to represent how often words co-occur is to use a so-called term-term matrix. This matrix builds upon a predefined vocabulary consisting of all unique words in a given corpus. For every word in the vocabulary, there is a row and a column in the matrix. Every cell m_{ij} in the term-term matrix m can, for instance, represent how many times the target word at row i co-occurs in some narrow context with the word at column j. To represent the association between words based on raw frequency is not optimal since frequencies can be skewed and not very discriminative. Another approach is, therefore, to use a weighted representation of associations based on how much more often two given words co-occur than what would be expected a priori by chance. This variant is called PPMI (positive pointwise mutual information). Regardless of what the cells in the term-term matrix represent, the row vectors constitute the representations of words in the semantic space. However, since words may only co-occur with a small number of different words from the vocabulary,

[1] In NLP, punctuation and parts of words are often referred to as tokens. Henceforth, "words" and "tokens" will be used interchangeably to denote this broader conception.

[2] An n-gram is a sequence of n words.

many cells in each row in the matrix will have zero values, bearing no information whatsoever. These long vectors with dimensions corresponding to the number of words in the vocabulary, with mostly zero counts or functions of counts, are called sparse vector representations.

It is preferable to represent words with so-called embeddings, which are short, dense vectors. It reduces the number of trainable parameters in a model significantly, which can help the model generalize better and avoid overfitting [12]. Instead of having word vectors with as many dimensions as the vocabulary size, e.g. $50,000$, with a lot of redundancy, dense vector representations usually have $50 - 1000$ dimensions with continuous values. Two widely used and efficient methods for creating embeddings has been Word2Vec [20] and Glove [23]. Rather than counting co-occurrences of words, Word2Vec uses the weights of a trained binary classifier as embeddings. This classifier is trained to predict whether words from the corpus are likely to show up near a given target word, but since the weights for given words constitute the embeddings, the predictions themselves are unnecessary. Advantageously, the training can be done in a self-supervised manner, meaning that the text itself can be used as training data and as a gold standard for the classification task since the labels (whether a given word show up near the target word or not) are inherent in the text.

Compared to neural language models, which learn embeddings from a word prediction task (and will be central in coming sections), Word2Vec is a much simpler approach [12]. This is in part because it reformulates the problem of creating word embeddings as a binary classification task. As a consequence, the architecture is also much simpler and more efficient to train. However, the main drawback with embedding representations created with methods such as Word2Vec and Glove is that these representations are static and context insensitive. Only one fixed embedding for each word in the vocabulary is learned. This matters a lot since language is often ambiguous, and meaning often depends on the context. For instance, the word *left* in the sentence *"My friend left the party."* has a different meaning than the same word in the sentence *"I waved with my left hand."*.

There have been significant advances in neural network architectures and language modeling approaches that have allowed for more dynamical contextual embeddings. When creating contextual embeddings, whole sequences are used as the context for a word, not only other words in its proximity. This provides a richer representation of contextual relationships between words in a given context, and the vector for each word depends on the context. Incorporating context into word embeddings in this way, as done in models such as ELMo [24], GPT-2 [27], and BERT [8], has led to state-of-the-art performances on virtually every NLP task. The embeddings in these models, including Word2Vec, are learned during pre-training and can thus be utilized and fine-tuned for different tasks. In Section 2.4 BERT will be accounted for in more detail.

2.2.2 Sequence-to-sequence

Abstractive summarization and other similar tasks where the input and output are sequences, possibly of different lengths, from separate domains (e.g. long text/summary) with some mapping between them, has largely been enabled by the sequence-to-sequence (seq2seq) framework first presented by Google in 2014 [36]. The main idea behind a seq2seq model is the so-called encoder-decoder architecture [12]. Simply put, an encoder's purpose is to turn the input sequence into an intermediate contextualized representation, usually called the context, preserving all the information in the input sequence. The purpose of the decoder is then to map this representation to an output sequence generated in the target domain. In the early years, the predominant approach to implement seq2seq models was to use recurrent neural network architectures such as long short-term memory networks [10]. In these models, the words in the input sequence are turned into embeddings and processed by the encoder in a sequential manner [12]. The first input embedding is fed to a recurrent unit which outputs a so-called hidden state. Then the next input embedding is fed to a recurrent unit together with the previous hidden state. This continues until all words in the input sequence has been processed

and the last hidden state has been outputted. This final hidden state serves as the context c. Formally, a sequence of hidden states h_i^e are generated by the encoder as a function f of the previous hidden state h_{i-1}^e and the input word x_i at position i according to equation 2.1.

$$h_i^e = f(x_i, h_{i-1}^e) \tag{2.1}$$

The context c is subsequently fed into the decoder, which uses it as its initial state [12]. The decoder will sequentially generate outputs y_i, as well as new hidden states (see equation 2.2) as a function g of the previously generated output y_{i-1} at position $i-1$ and the outputted hidden state h_{i-1}^d, until an end-of-sequence marker is generated.

$$h_i^d = g(h_i^d, y_{i-1}, c)^3 \tag{2.2}$$

In Figure 2.1 it is illustrated, with a machine translation example, how a sequence is first encoded and then decoded as described above.

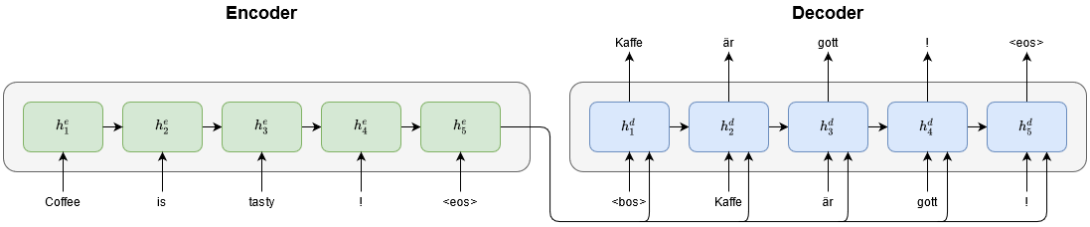

Figure 2.1: Encoder-decoder example

The output y_i is based on a probability distribution of all possible words in the vocabulary, given by a softmax calculation of each decoder hidden state respectively. One way for the model to choose the output is to pick the word with the highest probability at every position i [12]. This may not be optimal when considering the context of the whole sequence since there can be long dependencies and complex word relations between words that may get lost when picking the most probable word for the current position only. A solution to this is to pick the sequence with the highest combined probability. However, it is infeasible to search through all possible combinations of words, given an extensive vocabulary. A search algorithm called beam search is most often used. Instead of choosing the most probable word to generate at each position, the beam search algorithm keeps the k most probable words at each position and reduces the search space. These initial k outputs, called hypotheses, are then extended incrementally, each with k additional words, at subsequent steps by being passed to distinct decoders. The parameter k is often called beam-width.

A problem with the encoder-decoder model presented so far is that the context vector, i.e. the last hidden state of the encoder, acts as a bottleneck [12]. In other words, the context is all the information about the input sequence that the decoder gets, and thus it must represent every aspect of the input sequence. It may tough be the case, especially for long sequences, that the beginning of the sequence is not equally represented in the context vector due to the many computation steps between the first and the last hidden state.

2.2.3 Attention

The attention mechanism, introduced by Bahdanau et al. [3] and Luong et al. [18], is a solution to the bottleneck problem of handling long sequences that further improves performance significantly. Instead of passing just the last encoder hidden state to the decoder, with

[3]The context can be used by the decoder in different ways. Here the context is added as a parameter when calculating the hiddent states. This prohibits the influence of the context to wane as the output sequence is generated.

attention, a weighted sum of all the encoder hidden states is passed to the decoder [12]. The weights enable the decoder to focus only on the most relevant parts of the input sequence for the output that is being produced. Unlike the previously static context vector c based on the last hidden state, the attention-based context vector c_i is dynamic and changes for each position i in the decoding process. The output y_i and the decoder hidden state h_i^d is calculated as in equation 2.2 with the only difference that the constant context c is replaced with the dynamic context c_i.

To calculate c_i at each position in the decoding process, it must be decided how relevant each encoder state h_j^e is given the decoder state h_{i-1}^d [12]. This ability to compare one hidden state of interest to all other hidden states in a way that reveals their relevance in the current context is at the core of the attention-based approach. The relevance is embodied in a score function. The simplest score is the dot-product attention that measures how similar the given decoder state h_{i-1}^d is to an encoder state by taking the dot product between them. An alternative approach is to parametrize the score with its own set of learnable weights so that the network can learn which aspects of similarity between the states are important. Attention weights α_{ij}, that reflects the relevancy of each encoder state j to the prior hidden decoder state h_{i-1}^d, are then computed with a softmax of all scores for the current time step i according to equation 2.3.

$$a_{ij} = softmax(score(h_{i-1}^d, h_j^e)) \tag{2.3}$$

The context for the current decoder state is calculated as a weighted average over all the encoder hidden states according to equation 2.4.

$$c_i = \sum_j (\alpha_{ij} h_j^e) \tag{2.4}$$

In Figure 2.2 it is illustrated how the context vector c_i, for $i = 3$, is calculated with the help of the attention weights computed with equation 2.3.

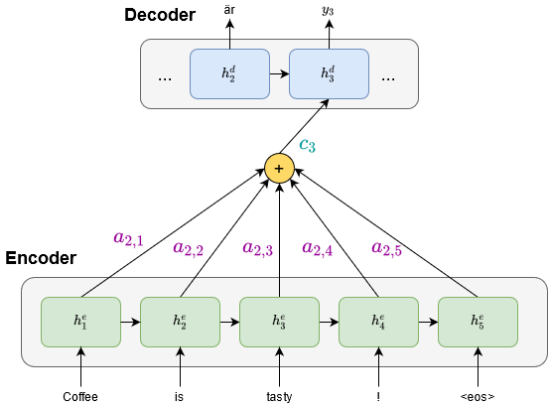

Figure 2.2: Attention example

Attention, as presented so far, has still been encapsulated in recurrent models. However, solving seq2seq problems with recurrent models, with or without attention, has its drawbacks. The inherently sequential nature of binding the positions to steps in computation time, where hidden states are dependent on previous hidden states, prevents parallelization within training examples [38]. This becomes an even more critical problem as the input sequences are long since memory constraints impede batching possibilities. Tackling these limitations, the Transformer, presented in the next section, has emerged as a better alternative for seq2seq tasks and language modeling in general, with state of the art results in many NLP tasks and significantly faster training times.

2.3 The Transformer

The Transformer, introduced by Vaswani et al. in 2017 [38], adopts an encoder-decoder architecture that evades recurrence and solely relies on the concept of attention to draw global dependencies between input and output. In the following subsections, the Transformer architecture and its key innovation - self-attention - will be described.

2.3.1 Encoder

On a high level, the encoder component in the Transformer consists of six individual encoders stacked on each other, all identical in structure but with separate weights. Each encoder comprises two modules: a self-attention layer (multi-head attention) and a feed-forward neural network [1]. The self-attention mechanism applied in the self-attention layer enables the encoder to look at different words in the input sequence that may lead to a better encoding for the word that is being processed, and thus it provides a rich context [12].

The first step when encoding a sequence is, as in the recurrent models described in Section 2.2.2, to transform all words in the input sequence to embeddings. By having the same dimension (512 in the Transformer implementation) for these input embeddings as for the output vector of each encoder, several encoders can be stacked as each output serves as the input for the next encoder [1]. In addition, positional encodings (see Section 2.3.5) are added to the embeddings. The resulting vectors from these additions are first passed through the self-attention layer (see Section 2.3.3) in the first encoder. The output from this self-attention layer is then fed to the feed-forward neural network, which yields the output of the encoder, one vector for each input embedding. These output vectors are used as input to the next encoder. This repeats until it reaches the top of the encoder stack. Here the output is a set of attention vectors (referred to as K and V in Section 2.3.3), corresponding to the context that will be passed to the decoder. It is worth noting that each input embedding can more or less flow through each encoder independently. In the self-attention layer, there are some dependencies, but in the feed-forward neural network, there are not. This allows for significantly more parallelization and faster training times than in recurrent encoder-decoder models.

2.3.2 Decoder

Similar to the encoding component, the decoding component consists of a stack of six decoders. These are identical to the encoders, with a few exceptions. Firstly, the self-attention layer is modified to prevent the decoder from looking at subsequent words [1]. This is done by masking future positions (before the softmax step in the self-attention calculation described in Section 2.3.3). Secondly, another self-attention layer called the encoder-decoder attention is added. This layer performs multi-head attention over the weights K and V outputted from the encoder stack together with the Query matrix Q from the preceding self-attention layer. All this helps the decoder attend to the most relevant parts of the input sequence as new words are generated without looking into subsequent words.

As with the encoder inputs, positional encodings are added to the decoder inputs at each decoding step to indicate the position of each word [1]. The inputs are processed and bubbled up through all six decoders, just like in the encoder stack. At the end of each decoder step, the decoder stack produces a probability distribution over all words in the vocabulary by first feeding the processed input to a fully connected neural network that maps the stack output onto a vector with the size of the vocabulary. Then a softmax is applied to this vector. As is described in Section 2.2.2, the words generated can result from choosing each word with the maximum probability or by choosing the words with the maximum combined probability. Furthermore, the output of each decoder step is fed to the bottom decoder in the next step until the end-of-sequence marker is generated.

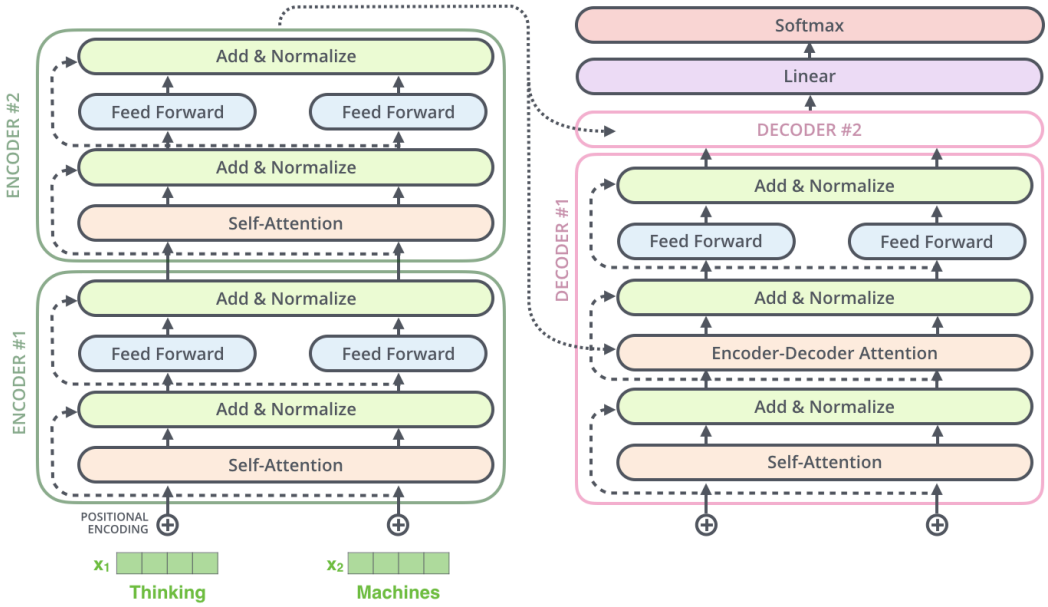

Figure 2.3: Transformer architecture in the context of processing the input sequence "Thinking Machines". In this example, there are only two encoder and decoders, respectively, unlike six as in the original Transformer implementation. Figure adopted from The Illustrated Transformer blog post by Alammar, J. [1]

In Figure 2.3 the architecture is displayed in the context of an example. As can be seen, there are also residual connections [9] in the form of dashed lines. This means that inputs to a module bypasses and is added to the output of the same module. Furthermore, the result of this addition is then passed to a Layer normalization function [2]. All this is encapsulated in the ”Add & Normalize”-block in Figure 2.3. The purpose of these is mainly to facilitate and make the training more efficient.

2.3.3 Self-attention

The first step in the process of calculating self-attention is to create three vectors: the Query vector q_i, the Key vector k_i and the Value vector v_i for each input embedding x_i [1]. These vectors are derived from multiplying the input embedding x_i with the matrices W^Q, W^K and W^V that are learned during training. The q_i, k_i and v_i vectors are by design down-scaled in dimension (from 512 dimensions to 64 dimensions) to make the multi-head attention computation easier (see Section 2.3.4). In the actual implementation, all inputs x_i are assembled in the matrix X which is multiplied with the trainable weight matrices W^Q, W^K and W^V to get the Q, V and K matrices packed with each q_i, v_i and k_i vector. This allows for more parallelization and faster computations. Nevertheless, for understanding, it is easier to consider the computations on the vector level. The keys and the values can be seen as the encoder hidden states drawing parallels to the recurrent seq2seq model presented earlier. Jufarsky [12] describes the intuition behind the q, v and k vectors in terms of different roles they play during the attention process.

- The **Query (Q)** plays the role as the current focus of attention when being compared to all of the other preceding inputs,
- the **Key (K)** plays the role as the preceding input being compared to the current focus of attention (the Query), and

- the **Value (V)** plays the role of a value used to compute the output for the current focus of attention.

The next step in calculating self-attention is to score the current focus of attention in the input sequence x_i against the other words x_j [1]. This scoring is analogous to the attention scoring described in Section 2.2.3 as it determines how much focus to place on other parts of the input sequence when processing a particular word. The scores are the dot product between the query vector q_i for the current word and the key vectors k_j for the other words. In the Transformer, this dot product is furthermore down-scaled to stabilize the gradients and avoid numerical issues. This is done by dividing them by 8 (the square root of the dimension of the key vectors, $\sqrt{d_k}$).

As before, the scores are passed to a softmax that normalizes them to add up to one [1]. These corresponds to the a_{ij} weights in equation 2.3. The weighting will highlight the importance of some words and drown out irrelevant words once again. The proportional relevance of each input x_j to the current focus of attention x_i can then be determined by multiplying each value vector v_j with the outputted softmax score for each k_j. Then as before, the weighted value vectors are summed to get the self-attention output z_i for position i that can be passed on to the feed-forward neural network. Zooming out, this happens for all queries and on the matrix level. The output of the self-attention calculation is thus the matrix Z containing all z_i vectors. These vectors contain information about how much each word in the input sequence should attend to the other words in the sequence. All this is encapsulated in equation 2.5 that calculates the output of the self-attention with matrices.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.5}$$

2.3.4 Multi-head attention

The multi-head attention, which serves as the actual self-attention layer in the Transformer [38], is a combination of eight different parallel self-attention heads, each with its own set of randomly initialized weights W_i^K, W_i^Q and W_i^V for head i. In other words, self-attention is computed eight times in parallel. The reason for doing this is that it can be challenging for one single attention head to account for parallel word relationships in a sequence [12]. To have several heads with separate weights allows for a more nuanced representation of word relationships in a sequence as the different heads can focus on different aspects of word relationships. It also expands the ability to focus on multiple word positions. The outputs Z_i from the different heads i are concatenated into a single vector that is multiplied with yet another weight matrix W^o jointly trained with the model to give the final output of the multi-head attention layer. Since each output vector is of dimension 64, the resulting vector from the concatenation and the linear transformation of the 8 outputs is in dimension 512. The equation for the multi-head attention is as follows:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_8)W^o \tag{2.6}$$

where

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{2.7}$$

2.3.5 Positional encoding

In recurrent models, the order of the input words is baked into the nature of the model since these inputs are processed in a sequential manner [12]. However, this is not the case for the Transformer as it parallelizes the processing. This makes it impossible to distinguish between the positions of elements in the input sequence. To handle this problem, the Transformer utilizes positional encodings, vectors with the same dimensions as the input embeddings, that

is supposed to capture the relationships among the positions. These vectors are unique to each position in an input sequence, and in the original Transformer, they were computed by combining sine and cosine functions with differing frequencies. The positional encodings are added to the input word embeddings, and the resulting vector serves as the input for further processing.

2.4 BERT

As the name suggests, BERT (Bidirectional Encoder Representations from Transformers) builds upon the Transformer architecture. More specifically, BERT utilizes the encoder stack of the Transformer. As a consequence of only utilizing encoders, BERT learns bidirectional representations during training [8]. This means that both the left preceding context and the right upcoming context are considered when looking at positions. Contextual relations between all the words in a sequence are learned. This is a significant difference from unidirectional models such as GPT-2 [27], which consist of Transformer decoders only, and thus solely consider the left previous context when predicting words during training.

Since subsequent words are considered in BERT, it would be cheating in the task of language modeling since the model already knows all words in the sequence that it is supposed to predict [8]. This is because the bidirectional connections indirectly allow subsequent words to be accessed without learning anything. To tackle this problem, BERT uses two pre-training tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) (see Section 2.4.3). As mentioned, BERT is an example of where transfer learning is applied. It can learn general representations of language during pre-training on large amounts of unlabelled data. Then the pre-trained model with learned representations can be fine-tuned on a specific task that often requires labelled data by modifying the final output layer. During pre-training, this output layer performs the MLM and NSP tasks.

2.4.1 Architecture

In the BERT paper [8], 12 ($BERT_{base}$) and 24 ($BERT_{large}$) Transformer encoders were used respectively compared to 6 encoders in the original Transformer implementation. The feed-forward neural networks are also bigger in BERT, 768 and 1024 hidden units, respectively, compared to the original Transformer implementation with 512 hidden units. Additionally, 12 and 16 self-attention heads are used compared to 8 in the original Transformer implementation. BERT processes input throughout the stack in the same way as the Transformer, with self-attention layers and feed-forward networks. What differs is the input to the first layer and the output from the final layer.

2.4.2 Input representation

To be able to fine-tune BERT on a range of different tasks, the input representation must unambiguously capture both representations of single sentences[4] and representations of pairs of sentences in one token sequence[5] [8]. Several special tokens are used for this. The first token in every input sequence is always the **[CLS]**[6] token. The final output embedding for this token is used as an aggregate representation of the whole sequence when classifying sequences. It is learned in the NSP task and is meaningless without fine-tuning.

Furthermore, sentence pairs in a single sequence are separated by the **[SEP]** token. If the sequence only contains one sentence, the **[SEP]** token marks the end of the input sequence. Additionally, learned embeddings, called segment embeddings, are added to every token to

[4] A sentence in this context can be an arbitrary span of contiguous text and not necessarily a linguistic sentence.

[5] A token sequence may be a single sentence or two sentences packed together.

[6] CLS stands for classification.

indicate whether the token belongs to sentence A or B. In the case of one single sentence, these will be identical. The input sequences to BERT must also be of the same length. This length can be defined manually and depends on how long the sequences are in the used dataset. However, BERT has a maximum length of 512 tokens. The **[PAD]** token is used to fill out shorter sequences to reach the defined length. As in the original Transformer implementation, positional encodings (position embeddings) are also added to the embeddings indicating the relative positions of tokens in the sequence. In Figure 2.4 an example of an input sequence being processed is illustrated.

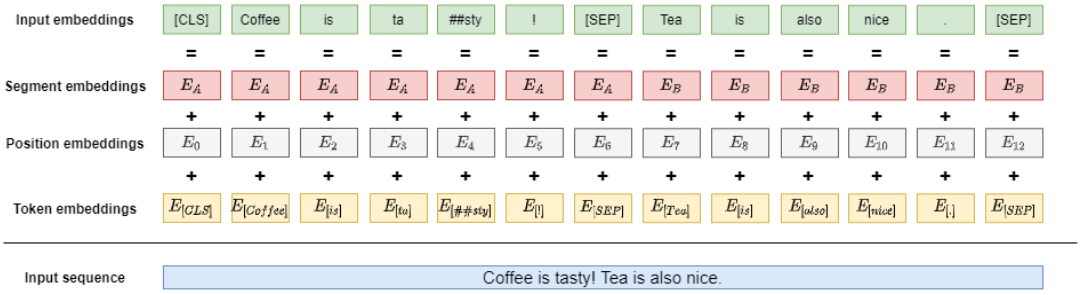

Figure 2.4: Example of a sequence being processed to yield the input embeddings for BERT

The BERT model's tokenizer first tokenizes the sequences that are passed into the model. In doing this, the sequences are split into so-called WordPieces embeddings [39] and special tokens are added. The WordPiece embeddings are based on words or smaller sub-word units defined in the model's vocabulary, which reflect the data used during pre-training. The reason to break words down into smaller sub-word units is to account for unknown words that would not be defined in a vocabulary of actual words. Assuming the word "tasty" is unknown in such a vocabulary, it could be defined in a wordPiece vocabulary based on the same data since the word can be broken down into, for example, "ta" and "##sty"[7]. With these wordPiece tokens and the special tokens, unique token embeddings can be constructed for each word in the input sequence, even for words not in the defined vocabulary. The vocabulary for the $BERT_{base}$ model in the BERT paper consists of $30,000$ wordPiece tokens.

2.4.3 Pre-training BERT

During the pre-training of bidirectional embedding representations, BERT is, as mentioned, trained unsupervised on two tasks: MLM and NSP. In the MLM task, a certain percentage of the tokens in an input sequence is masked randomly, and the objective is to predict those masked words [8]. This type of task is often referred to as a "Cloze task" [37]. In the BERT implementation, 15% of the words in the input sequence are masked. When masking tokens, the special token **[MASK]** is used, and the corresponding output embeddings from the top encoder are passed to a softmax that outputs a probability distribution over the vocabulary as in standard language modeling. This task makes the model learn what words fit in a particular context since the context is the only information available when predicting words. Because no tokens are masked during fine-tuning, however, this causes a mismatch between the pre-training and the fine-tuning [8]. To alleviate this problem and make the model more robust, tokens are not always replaced with the actual **[MASK]** token when masked. This is done in 80% of the cases. In 10% of the cases, a random token from the vocabulary is used as the mask, and in 10% of the cases, the token is not replaced.

In the NSP task, the model's task is to predict, given two sentences A and B, if B is likely to be the sentence that follows sentence A [8]. In 50% of the cases, B is the actual sentences following A, and in 50% it is a randomly selected sentence from the corpus. This method

[7]The ## symbol indicates that these wordPieces are suffixes to another wordPiece.

makes BERT better at handling relationships between sentences which is essential in many downstream tasks.

2.4.4 Sentence-BERT

One disadvantage with the BERT architecture is that independent sentence embeddings can not be computed. This makes it hard to derive good representations of sentences. As described in Section 2.4.2 and 2.4.3 the input for BERT in sentence tasks consist of sentence pairs separated by the **[SEP]** token. To use this setup in sentence tasks is computationally expensive [30]. For example, if one were to find the two most similar sentences among $10,000$ sentences through pair-wise comparisons, it would require $49,995,000$ inference computations (due to too many possible combinations) which would take about 65 hours on a modern GPU. A common way to bypass this problem has been to pass single sentences through BERT and derive a sentence representation either by averaging the output embeddings or by using the output corresponding to the **[CLS]** token [30]. These can then be mapped onto a vector space such that semantically similar sentences are close to each other by using different clustering and semantic search techniques. However, this approach yields rather bad sentence embeddings.

To address this issue, Reimers and Gurevych [30] developed a novel modified version of BERT, which they called Sentence-BERT (SBERT). First, they added a pooling operation to the output of BERT in order to derive sentence embeddings. Different pooling strategies were deployed, but the default one was to compute the mean of all output vectors. Then they fine-tuned this model on Natual Language Inference data (NLI)[8] data using siamese and triplet networks [32] to update the weights such that the produced sentence embeddings were close to each other in vector space if the sentences were semantically similar. The sentence embeddings yielded from this fine-tuned model significantly outperformed other state-of-the-art sentence embedding methods. This method expands the possibilities of using BERT in tasks that were not applicable before, such as large-scale semantic similarity comparisons and information retrieval via semantic search. With measures such as cosine similarity (see Section 2.6.3), the semantic similarity search example with $10,000$ sentences could be performed in approximately five seconds.

2.5 Warm-starting seq2seq models with BERT

The general language representations from a pre-trained BERT model can, in general, be fine-tuned for various downstream tasks. This is often done by just adding an output layer that is then jointly trained with the pre-trained parameters during the fine-tuning [8]. For example, a classification layer could be added on top of BERT to perform sentence classification. With its encoder-based architecture, BERT was primarily developed for encoding text representations for Natural Language Understanding tasks. On the other hand, GPT-2 [27], as a decoder-only architecture, was intended for text generation tasks. Based on these assumptions, arguments have been that BERT is not well suited for text generating tasks that require decoding, such as abstractive summarization [40]. Fine-tuning BERT for abstractive summarization poses a challenge since the task requires both understanding and language generation capabilities. On the contrary, Rothe et al. [31] proved that it is, in fact, useful and efficient to warm-start seq2seq models with BERT checkpoints.

2.5.1 Model architecture

The Transformer-based seq2seq architecture of Rothe et al. [31] enables initializing both the encoder and the decoder weights with pre-trained checkpoints. The architecture is compatible with publicly available pre-trained BERT, GPT-2 and RoBERTa checkpoints and combinations

[8] NLI is the task of determining whether a hypothesis is true (entailment), false (contradiction), or undetermined (neutral) given a premise.

of these checkpoints can be used, in any way, to warm-start an encoder-decoder model. For example, the encoder and the decoder can be initialized with pre-trained checkpoints from an encoder-only model such as BERT. The BERT2BERT (BERT-initialized encoder paired with a BERT-initialized decoder) configuration is the one that is relevant in this thesis.

More specifically, this seq2seq model consists of an encoder component that is inherited from BERT. All weight parameters in the seq2seq model are initialized with corresponding BERT parameters, and before any further fine-tuning, this encoder behaves precisely like the pre-trained BERT model. The decoder component in the seq2seq model is similar to the BERT encoder component but differs in three ways. The first difference is that the right context is masked in the self-attention layers. The second difference is that encoder-decoder attention layers are added. The weights in these layers are randomly initialized. The third difference is that the output layer is changed to be suitable for text generation. These changes are analogous to how the Transformer is used for seq2seq modeling as described in Section 2.3.2. When using BERT for seq2seq tasks, there is furthermore no need for the weights corresponding to BERT's **[CLS]** output.

2.5.2 Weight sharing

One possibility when warm-starting an encoder-decoder model is to initialize the encoder and the decoder with the same weights as proposed by Raffel et al. [28]. This means that all the layers in the encoder with corresponding layers in the decoder have the same weights. With this approach, a model with shared BERT weights (BERTshare) can be warm-started. As shown by Rothe et al. [31] this reduces the number of parameters significantly, from 221M in BERT2BERT to 136M in BERTshare, and yields similar results as with BERT2BERT after fine-tuning. In some tasks, it yields even slightly better results.

2.5.3 Fine-tuning for summarization

After warm-staring an encoder-decoder model, the weights can be fine-tuned on a seq2seq downstream task, such as summarization. Rothe et al. [31] fine-tuned several model configurations for abstractive summarization. Here the CNN/Daily Mail dataset and two other summarization datasets were used. When fine-tuning on the CNN/Daily Mail dataset, articles longer than 512 tokens were truncated, as 512 tokens are the maximum input length for BERT. Similarly, the length of the summaries was limited to 128 tokens. A batch size of 128 was used for the CNN/Daily Mail dataset, and the model was fine-tuned for 300k steps. These hyper-parameters differed slightly between the different datasets due to dataset size and other characteristics such as article and summary lengths. One of the best performing model configurations was the BERTshare configuration which achieved ROUGE-1, ROUGE-2 and ROUGE-L (see Section 2.6.1) of 39.09, 18.10 and 36.33 respectively.

All in all, this method is effective since it does not require training seq2deq models from scratch, and it demonstrates state-of-the-art results. Moreover, one of the main advantages is that it can be used for seq2seq tasks in languages with no pre-trained decoder-only models such as GPT-2, but only pre-trained encoder-only models.

2.6 Evaluation

When evaluating a trained model on a task such as abstractive text summarization, it is crucial to apply adequate measures. In text summarization, one wants to evaluate how good a generated summary is regarding various factors. Unfortunately, there is no single ideal measure in this regard; rather, several measures, all with pros and cons. Methods for evaluating summaries can broadly be divided into intrinsic and extrinsic measures [35]. Extrinsic measures judge the summary quality based on the helpfulness of the summaries in a given task, while intrinsic measures judge the summary quality based on an analysis of summaries

themselves. This is often done by comparing the generated candidate summary to an "ideal" reference summary or to the source document measuring how many main ideas of the source document are covered by the summary. Intrinsic measures can furthermore be divided into measures that focus either on text quality or the content of the text. Text quality concerns the text's readability, grammar, clarity, coherence and structure and is hard to judge automatically. Content measures judge the ability to identify key topics in the given document. Two commonly used ways to do this is by applying co-occurrence statistics between words in the different summaries or measuring the cosine similarity between some composed vector representation of the candidate and the reference summary. A good metric for evaluation should have high correlations with human judgments.

2.6.1 ROUGE-measures

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [16] constitute a family of commonly used evaluation metrics based on co-occurrence statistics. The main variants are ROUGE-N, ROUGE-L, ROUGE-W and ROUGE-S. Each automatically measures the similarity between a generated candidate summary and the reference summary by looking at overlapping n-grams and word sequences, although in somewhat different ways. In this thesis, only ROUGE-N and ROUGE-L will be utilized since these are most commonly reported in papers.

ROUGE-N is an n-gram recall between the candidate summary and the reference summary, i.e. the percentage of n-grams in the reference summary present in the generated summary. Formally this can be described as in equation 2.8 below, where n denotes the length of the n-gram $gram_n$, $Count_{match}(gram_n)$ the number of n-grams co-occurring in the candidate summary and the reference summary, and $Count(gram_n)$ the total number of n-grams in the reference summary. It is also worth noting that even though this measure is recall-oriented, it can easily be modified to measure precision and F-score.

$$\text{ROUGE-N} = \frac{\sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{gram_n \in S} Count(gram_n)} \tag{2.8}$$

On the other hand, ROUGE-L is a measure based on the longest common subsequence (LCS) of words between a candidate summary and a reference summary. The idea behind LCS is that the longer the LCS of two summary sentences is, the more similar the two summaries are. Lin et al. [16] uses the F-measure of ROUGE-L based on recall (see equation 2.9) and precision (see equation 2.10). It is calculated as in equation 2.11 where Y is the generated candidate summary, X the reference summary, n and m their respective lengths, and $LCS(X,Y)$ is the length of the longest common subsequence of X and Y. $\beta = \frac{P_{lcs}}{R_{lcs}}$ when $\frac{F_{lcs}}{R_{lcs}} = \frac{F_{lcs}}{P_{lcs}}$.

$$R_{lcs} = \frac{LCS(X,Y)}{m} \tag{2.9}$$

$$P_{lcs} = \frac{LCS(X,Y)}{n} \tag{2.10}$$

$$F_{lcs} = \frac{(1+\beta^2) R_{lcs} P_{lcs}}{R_{lcs} + \beta^2 P_{lcs}} \tag{2.11}$$

2.6.2 BERTScore

In text generation tasks, n-gram-based metrics such as ROUGE has some drawbacks regarding semantic equivalence and content overlap between generated candidate sentences and reference sentences. They rely on surface-form similarity and do not account for meaning-preserving lexical and compositional diversity [42]. Hence, they often fail to match paraphrases robustly. For instance, given the reference sentence *"people like foreign cars"* the candidate sentence

"people like visiting places abroad" would receive higher ROUGE scores than the sentence *"consumers prefer imported cars"*. In other words, surface form is not equivalent to semantic similarity. They also disregard distant dependencies and penalize semantically critical ordering changes. For example, given the reference sentence *"I like apples, bananas and oranges"*, the candidate sentence *"I like oranges, apples and bananas"* would receive a low ROUGE-2 score just because the ordering is different.

Zhang et al. [42] addressed these pitfalls with BERTScore, which correlates better with human judgments than existing metrics. Unlike measures that use n-gram matching, BERTScore utilizes contextualized token embeddings from BERT, effectively capturing distant dependencies and ordering. More precisely, BERTScore computes the similarity between a reference sentence X and a candidate sentence Y as a sum of cosine similarities between their tokens' embeddings. A greedy-matching strategy is deployed, in which each token is matched to the most similar token in the other sentence to maximize the matching similarity score. Cosine similarity is defined in Section 2.6.3, but in the below equations of BERTScore precision 2.13, recall 2.12 and F-score 2.14 pre-normalized vectors are used. Thus, the cosine similarity is $x_i^T \hat{x_j}$.

$$R_{BERT} = \frac{1}{|x|} \sum_{x_i \in x} \max_{y_j \in y} x_i^T y_j \tag{2.12}$$

$$P_{BERT} = \frac{1}{|y|} \sum_{y_j \in y} \max_{x_i \in x} x_i^T y_j \tag{2.13}$$

$$F_{BERT} = 2\frac{P_{BERT} R_{BERT}}{P_{BERT} + R_{BERT}} \tag{2.14}$$

BERTScore also enables importance weighting with inverse document frequency (IDF) scores computed from the test corpus [42]. This is based on the rationale that rare words can be more indicative of sentence similarity than common words. In Figure 2.6.2 the process of calculating BERTScore between two sentences is illustrated.

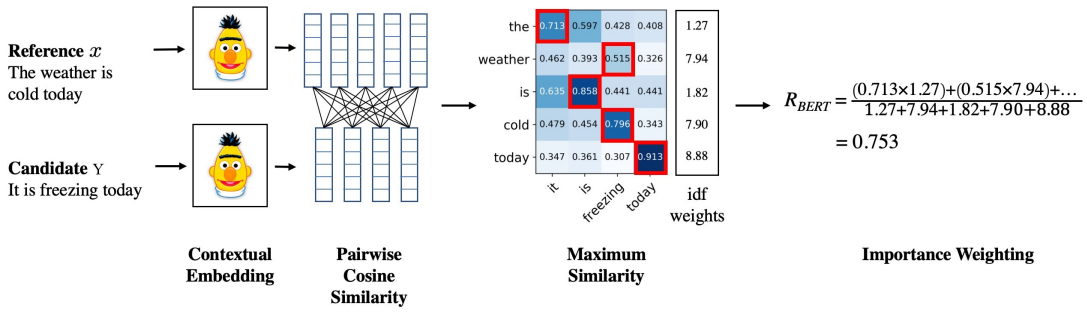

Figure 2.5: BERTScore calculated between two sentences. Figure adopted from Zhang et al. [42]

In practice, scores for some models end up in a very limited span within the range of −1 to 1 [42]. This does not affect scoring capabilities, but it makes the scores harder to interpret. Therefore a language and model-specific "baseline rescaling" can be utilized to adjust the output scores. This is done by creating 1M randomized candidate-reference pairs from the given corpus by grouping two random sentences. This way, each pair has very low lexical and semantic overlapping. The baseline b is computed by averaging the BERTScore on all these sentence pairs. The actual scores are then rescaled according to equation 2.15[9]. Zhang et al.

[9]The same rescaling is applied to precision and recall as well.

[42] noted that the ranking ability and human correlation of BERTScore were not affected by rescaling.

$$F_{BERT_{rescaled}} = \frac{F_{BERT} - b}{1 - b} \tag{2.15}$$

BERTScore was originally developed as a metric for machine translation and focused on comparing single sentences. Abstractive summarization share similarities with machine translation but differs in the sense that summaries often consist of more than one sentence. However, it has been shown by Paraschiv and Cercel [22] that BERTScore also can be useful for evaluating reference summaries with candidate summaries in abstractive summarization. To evaluate summary quality, they deployed both SBERT (see Section 2.4.4) and BERTScore and examined how well they correlated with human judgment. BERTScore proved to be more in line with the human evaluators.

2.6.3 Cosine similarity

In many NLP applications there is a need to compare two or more vector representations of words or sequences in a larger vector space. As described in Section 2.2.1, embeddings that are close to each other in vector space can be considered to be semantically similar. One way of measuring the similarity of vectors in such a space is to use cosine similarity. The cosine similarity measure can formally be defined as follows:

$$cos(X, Y) = \frac{\sum_i x_i y_i}{\sqrt{\sum_i (x_i)^2}\sqrt{\sum_i (y_i)^2}} \tag{2.16}$$

where X and Y are vectors in a given vector space model and x_i and y_i denotes their respective components [35]. The similarity score is always between 0 and 1 and the higher the score the more similar the given vectors are.

3 | Building summarization datasets

This chapter will present the methods used for compiling and filtering the DN data. First, in 3.1, the process of compiling and characterizing and the data is described. Then in Section 3.2 the filtering methods will be explained, and the results of this filtering will also be presented.

3.1 Compiling the DN data

The datasets used in this thesis were compiled from $1,963,576$ news articles provided by DN (Dagens Nyheter)[1] stored in 21 JSONL files, one for each year from 2000-2020. Each data row consisted of the article's headline, summary and body[2] as well as the publication time, URL address and a unique id. Initially, data rows where the article was shorter than 25 words or where the summary was shorter than 10 words were removed, as well as duplicates. In MLSUM [33] thresholds of 50 and 10 words were used for article and summaries, respectively, but as further filtering mechanisms were to be used, a lower threshold was set to include more data. Furthermore, data rows where the ratio between the number of words in the summary and the number of words in the article, also called compression ratio[3], was higher than 0.4 were removed. This limit was based on the rationale that a summary should be rather short relative to the article and on the fact that very few article-summary pairs in the CNN/Daily Mail dataset had a compression ratio higher than 0.4. An additional filter applied was that article-summary pairs with articles longer than 2500 words or summaries longer than 200 words were removed to avoid extreme outliers. After this filtering and the removal of duplicates, $821,405$ data rows remained.

3.1.1 Data characterization

The $821,405$ remaining article-summary pairs after the initial filtering were characterized further with respect to several aspects. For comparison purposes, the CNN/Daily Mail dataset was characterized and examined the same way to understand what characteristics make a good dataset for abstractive summarization.

In characterizing the data, article and summary lengths (both with respect to words and sentences), novelty (n-gram), compression ratio, vocabulary size and word occurrences were computed. The novelty (n-gram) measure was meant to be a proxy for the attractiveness of the summary, as it is the fraction of n-grams in a given summary that does not appear in the paired article [33]. In calculating novelty, stop words were removed, and the rest of the words were stemmed so that grammatical inflexions would not matter. These properties presented hitherto are surface properties and do not account for the underlying semantics in the article-summary pairs.

To get a sense of how well the summaries captured the semantic meaning in the articles, semantic textual similarity was computed for each article-summary pair by utilizing

[1] `https://www.dn.se/`

[2] The body will henceforth be referred to as the article.

[3] The reason for not measuring compression ratio as article length divided by summary length as was done by Scialom et al. [33], but rather the other way round, was to get more interpretable values between 0 and 1.

SBERT [30]. This was done in two ways. First, cosine similarity between the embedding representations yielded from SBERT for the complete summary and the complete article was computed (doc/doc). Secondly, semantic textual similarity between the embedding representation yielded from SBERT for the complete summary and each sentence in the associated article was computed (doc/sent). Here the score for the most semantically similar sentence was used. The reason for computing semantic similarity this way was to make the characterization more robust and account for the possibility that some sentences in the article could be semantically similar to the summary without the complete article being semantically similar. The SBERT model that was used was the `distiluse-base-multilingual-cased-v2` model. All these characteristics for the DN data are presented in Table 3.1 together with the characteristics for the CNN/Daily Mail dataset. Compared to the properties of the CNN/Daily Mail dataset presented in Table 2.1, the properties presented here are slightly different due to calculation differences.

	DN	CNN/DM
Dataset size	821,405	311,971
Vocabulary size	2,605,651	803,487
Occurring 10+ times	392,904	161,820
Article words	371.71	677.21
Article sentences	24.07	28.52
Summary words	29.37	48.34
Summary sentences	2.20	3.70
Compression ratio	0.13	0.09
Novelty (uni-gram)	0.41	0.14
Novelty (bi-gram)	0.79	0.57
Novelty (tri-gram)	0.93	0.77
Semantic sim (doc/doc)	0.49	0.65
Semantic sim (doc/sent)	0.52	0.67

Table 3.1: Statistics for the DN data and the CNN/Daily Mail dataset. Dataset size is the number of article-summary pairs. Vocabulary size is the total number of different words in the corpus and Occurring 10+ times, the total number of words occurring 10+ times. Article/Summary words/sentences are the number of words/sentences in the articles/summaries. Compression rate, novelty, and semantic similarity are presented in the text. All values except Dataset size, Vocabulary size and Occurring 10+ times are mean values across all article-summary pairs.

To get a better sense of the distributions for these characteristics among the data, they are presented in graphs in Figure 3.1. As can be seen in these graphs and Table 3.1, the DN dataset is much larger than the CNN/Daily Mail dataset. It also shows the difference in terms of article and summary lengths with significantly shorter articles and summaries. The DN data also has a higher novelty, mainly uni-gram and bi-gram novelty. This indicates that there are many lower-quality article-summary pairs where the summary does not contain the contents from the article or that the summaries are very abstract and use different words than the article. In terms of semantic textual similarity, the DN dataset contains less semantically similarly article-summary pairs (doc/doc) and less semantically similar article sentences relative to summaries (doc/sent). This also points to the current problem of low-quality article-summary pairs.

3.1.2 Article categories

There are many different types of new articles in a typical newspaper. To get a sense of the types of articles the DN data contained, the URL address for each article was utilized. The URL addresses consisted of pieces of text separated with /. The last piece reflected the article's headline, and the other pieces were subcategories and categories. For example a URL address could look like: */sport/fotboll/manchester-city-overklagar-inte-avstangning/*. The categories

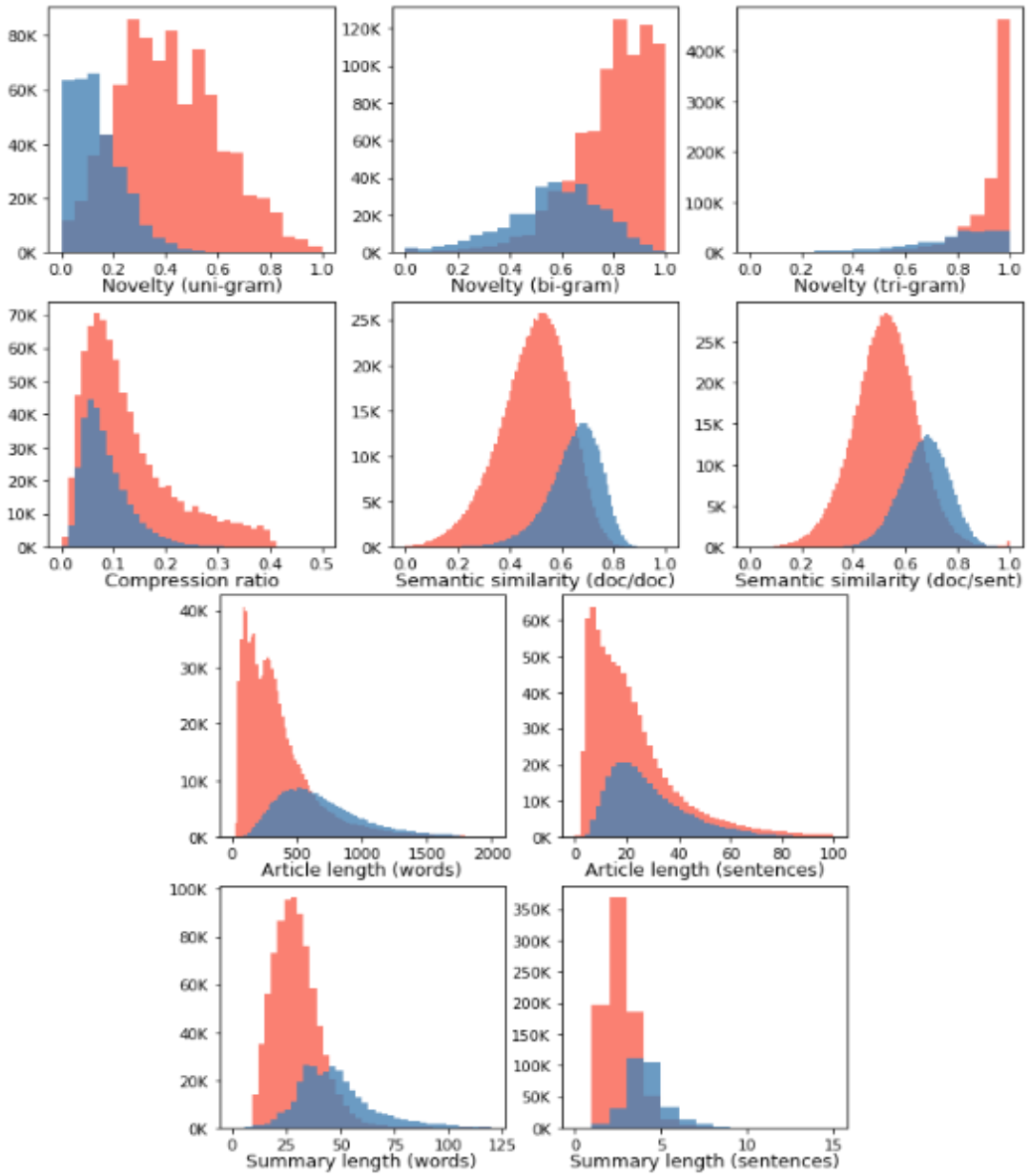

Figure 3.1: Distributions for the DN data (red) and the CNN/Daily Mail dataset (blue) characteristics. The y-axis in all graphs is the number of article-summary pairs.

were extracted by removing the last piece of text for each URL address. This resulted in 240 unique categories. These were manually examined and categorized into five broad categories: sports, economy, culture, domestic news and others (mostly very small subcategories). Of the 821,405 articles, 42% of were categorized as domestic news, 22% as sports, 16% as economy, 10% culture and 9% other.

3.2 Filtering

With the characteristics for the DN data and the CNN/Daily Mail dataset at hand, the goal was to create high-quality datasets that resembled the properties of the CNN/Daily Mail dataset. This was done by applying additional filtering thresholds on novelty and semantic similarity,

both separately and combined. Those article-summary pairs that were not semantically similar or concrete enough, based on novelty, were thus removed. This type of filtering was not done with MLSUM [33].

First of all, $19,000$ article-summary pairs were set aside as test data. $10,000$[4] of these pairs were randomly sampled from the DN data and had thus similar characteristics. The other $9,000$ were sampled from a filtered subset of the DN dataset with similar characteristics as the CNN/Daily Mail dataset, the same way as DN-sn was filtered (described below). These test subsets are referred to as test-lc and test-sn, respectively.

The DN dataset filtered on length and compression ratio, test data deducted, consisted of $802,405$ article-summary pairs. This subset is henceforth referred to as DN-lc. Four additional datasets were compiled from DN-lc by applying further filtering. These are referred to as DN-s, DN-n, DN-sn and DN-s∪n. DN-s was filtered on semantic similarity first using the doc/doc similarity measure, then the doc/sent similarity measure. The threshold values for filtering were iteratively adjusted to obtain similar distributions as the CNN/Daily Mail dataset. DN-n was filtered to have similar distributions as the CNN/Daily Mail dataset regarding novelty (uni-gram, bi-gram and tri-gram). It turned out that filtering on uni-gram novelty also provided similar bi-gram and tri-gram distributions since they are strongly correlated. DN-sn was filtered by both semantic similarity and uni-gram novelty by the same principles as above. The threshold values were again iteratively adjusted to obtain a similar distribution as the CNN/Daily Mail dataset. The final dataset DN-s∪n was simply the union of DN-s and DN-n. The results of this filtering are shown in Table 3.2, with the characteristics of the CNN/Daily Mail dataset as reference. The distribution among news categories was approximately identical to the original DN data for all subsets, except for DN-n, which had 40% domestic news, 18% other, 17% culture, 13% economy, 12% sports.

	DN-lc	DN-s	DN-n	DN-sn	DN-s∪n	CNN/DM
Dataset size	802,405	122,419	124,105	38,151	215,286	311,971
Vocabulary size	2,575,130	727,406	1,070,351	435,412	1,291,347	803,487
Occurring 10+ times	387,370	118,66	171,100	70,450	205,451	161,820
Article words	370.41	362.52	630.36	512.43	496.62	677.21
Article sentences	24.00	22.51	40.27	31.71	31.53	28.52
Summary words	29.31	32.15	33.16	35.67	32.30	48.34
Summary sentences	2.19	2.38	2.51	2.62	2.43	3.70
Compression ratio	0.13	0.13	0.07	0.10	0.10	0.09
Novelty (uni-gram)	0.42	0.32	0.14	0.14	0.24	0.14
Novelty (bi-gram)	0.80	0.73	0.57	0.57	0.66	0.57
Novelty (tri-gram)	0.93	0.89	0.78	0.78	0.84	0.77
Semantic sim (doc/doc)	0.49	0.65	0.52	0.65	0.57	0.65
Semantic sim (doc/sent)	0.52	0.67	0.60	0.67	0.63	0.67

Table 3.2: Statistics for the filtered DN datasets

Furthermore, the distributions among the characteristics for the DN-sn dataset are displayed in Figure 3.2 below. One can see that the distributions for this filtered dataset are more similar to the CNN/Daily Mail dataset than the original DN data. At the same time, it is also significantly smaller. One can also see that the threshold values for filtering were about 2.5 for novelty (uni-gram) and 5.6 for semantic similarity.

[4] $1,000$ of these were used as validation data to evaluate the models during fine-tuning.

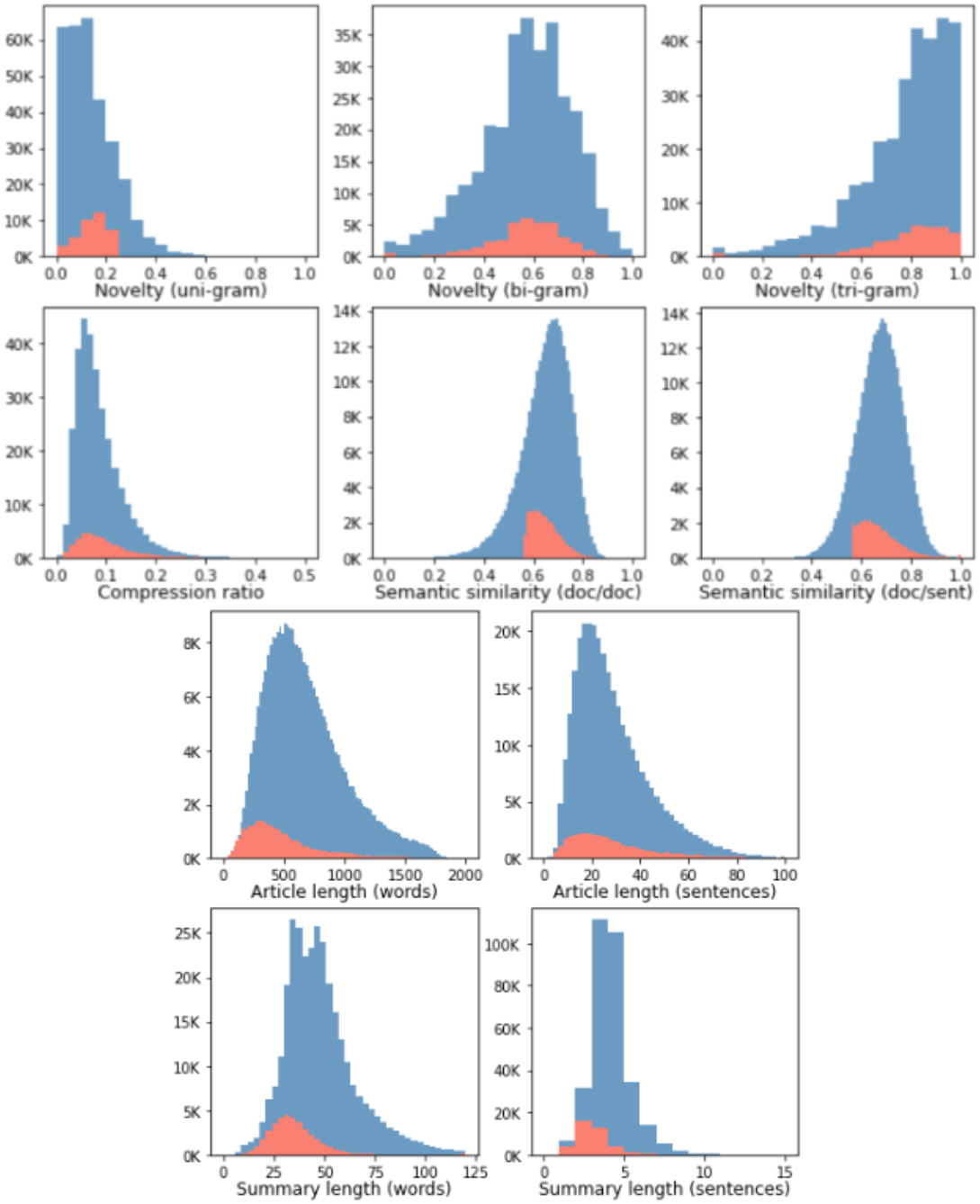

Figure 3.2: Distributions for DN-sn (red) and the CNN/Daily Mail dataset (blue) characteristics. The y-axis in all graphs is the number of article-summary pairs.

4 Using the datasets for abstractive summarization

In this chapter, the usage of the produced datasets from Chapter 3 is described. This consisted of implementing and evaluating abstractive summarization models trained on the different datasets, respectively. In Section 4.1 the implementation of summarization models will be described in detail. In Section 4.2 it is explained how the fine-tuned models were evaluated. Subsequently, the evaluation results for the fine-tuned models are presented in Section 4.2.2. In order to illustrate the models´ capabilities, some examples of generated summaries (in Swedish) are presented in Section 4.3.

4.1 Model implementation

This section concerns how the abstractive summarization models were implemented. First, the Swedish pre-trained BERT model is described, followed by an explanation of how the data was prepared as input for the model. Then the process of warm-starting, fine-tuning and evaluating the encoder-decoder models is described. For implementing the models, i.e. warm-starting and fine-tuning the encoder-decoder models for the different datasets, the Python Huggingface Transformers library was used[1].

4.1.1 The pre-trained model

The pre-trained BERT model used in this thesis was released by The National Library of Sweden/KBLab [19]. It was trained on approximately 15-20GB of text (200M sentences, 3000M tokens) from various sources such as books, news, government publications, Swedish Wikipedia and internet forums with the aim to provide a representative BERT model for Swedish text. More specifically, the `bert-base-swedish-cased` model was used, which corresponds to the original $BERT_{base}$ model in terms of size and has a vocabulary size of $50,325$ tokens.

4.1.2 Preparing the data as model input

In preparing the data as input to the encoder-decoder model, all unnecessary attributes were removed, i.e. all attributes except the summary and the article. The summaries and articles were then tokenized with the pre-trained model's tokenizer by utilizing the `BertTokenizerFast` class and the `from_pretrained()` method to load the tokenizer. Since the maximum input length to BERT is defined as 512 tokens (as described in Section 2.4), all articles with more than 512 tokens were truncated to 512 tokens. This applied to about a third of all articles in the DN data with $821,405$ article-summary pairs. In practice, this means that some information in these articles was disregarded, which may not have been optimal. However, this approach was, used by Rothe et al. [31] with good results on the CNN/Daily Mail dataset, based on the reasoning that the most important information is found at the beginning of articles most often. The articles in the CNN/Daily Mail dataset are even longer on average than those in

[1] `https://github.com/huggingface/transformers`

the DN data as illustrated in Figure 3.1 which indicate that this had even less impact on the models fine-tuned on the DN data.

Since the length for the summaries had already been limited to 200 words and very few summaries were longer than that (as can be seen in Figure 3.1), there were virtually no summaries with more than 200 tokens. Nevertheless, the maximum output limit was set to 200 tokens for the decoder. Input ids and attention masks were also computed, and padding was applied so that shorter articles and summaries would be of the same length. This was done both for the encoder inputs and the decoder inputs.

4.1.3 Warm-starting the encoder-decoder model

Since the BERTshare model, proposed by Rothe et al. [31] yielded good results on the CNN/Daily Mail dataset (a ROUGE-1 score of 39.09 and a ROUGE-2 score of 18.10) and had significantly fewer parameters than the BERT2BERT implementation, the Swedish seq2seq model in this thesis was initialized with the pre-trained model checkpoints from `bert-base-swedish-cased`, shared between the encoder and the decoder. This was done using the `EncoderDecoderModel` class from the Huggingface library and the `from_encoder_decoder_pretrained()` method.

Since BERT has no special tokens for the decoder that indicates the start and end of sentences, these were set to be the **[CLS]** and **[SEP]** respectively. Additionally, the padding token for the encoder-decoder model was set as the BERT padding token, and the vocabulary size was configured as the `bert-base-swedish-cased` vocabulary. The beam-search settings were defined based on what usually is used on the CNN/Daily Mail dataset. However, since the summaries in the DN data are a bit shorter, the settings were adapted to that. A beam-width of 4 was utilized, and the max output length was set to 100 tokens and minimum length to 35 tokens. Furthermore, a length penalty of 2.0, early stopping and a no-repeat-n-gram size of 3 was applied to force summaries not to be longer than necessary and to avoid repetition.

In addition to the Swedish warm-started encoder-decoder model, an English encoder-decoder model based on the `bert-base-cased` model was warm-started the same way to be fine-tuned on the CNN/Daily Mail dataset. The reason for this was mainly to verify the results from Rothe et al. [31] and also to enable evaluation with BERTScore. In the coming sections, this is the model referred to when talking about the CNN/Daily Mail dataset.

4.1.4 Fine-tuning the model

The fine-tuning was done using the Huggingface `Seq2SeqTrainer` class. When fine-tuning the warm-started encoder-decoder model on the different datasets presented in Table 3.2 slightly different hyperparameters were used, accounting for the fact that the datasets differed in size. For all datasets, a batch size of 8[2] was used. Most other parameters were set to default. The main parameters that differed between the datasets were the number of training steps and warm-up steps. These are presented in Table 4.1. The fine-tuning of all models was conducted on a Tesla V100-SXM2-16GB GPU in Google Colab.

	DN-lc	DN-s	DN-n	DN-sn	DN-s∪n	CNN/DM
Training steps	300,000	100,000	100,000	50,000	150,000	150,000
Warm-up steps	4,000	2,000	2,000	1,000	2,000	2,000

Table 4.1: Training parameters for the models fine-tuned on the different datasets

[2] A larger batch size would have been preferable, but since Google Colab was used, the GPU memory was limited.

4.2 Model evaluation

To evaluate how well the fine-tuned models could summarize articles, they were tested on test-lc and test-sn. The articles in these test sets were used as input to the model that would then generate candidate summaries. ROUGE-scores, as well as BERTScores, were calculated. ROUGE-1, ROUGE-2 and ROUGE-L were calculated as described in Section 2.6.1 by utilizing the Huggingface `datasets.Metric` class.

4.2.1 BERTScore

BERTScore was computed with the bert_score python package[3]. Unfortunately, no Swedish pre-trained BERT model was available for BERTScore. Hence the multilingual model `bert-base-multilingual-cased` was used instead. This also meant that there was no rescaling baseline available for Swedish to make the scores more interpretable. So to create a Swedish baseline the `get_rescale_baseline` script was modified to use sentences from the DN data. Since summaries on average had about two sentences, 2M sentences were paired randomly, creating 1M examples used to rescale the baseline. When calculating the scores, IDF-weights, computed from the test sets, were also applied to value rare words. The same model but with an already existing English rescaled baseline was used to evaluate the model fine-tuned on the CNN/Daily Mail dataset. IDF-weights was computed from the CNN/Daily Mail test set.

4.2.2 Evaluation results

In Table 4.2 the evaluation results for all fine-tuned models are presented. In the leftmost column, it is indicated which dataset the model was fine-tuned on. Note that the BERTScore for the model trained on the CNN/Daily Mail dataset may not be comparable to the BERTScores for the Swedish models due to baseline differences and because embedding representations may differ between languages.

	ROUGE-1		ROUGE-2		ROUGE-L		BERTScore	
CNN/DM	39.89		18.18		27.54		17.52	
	test-lc	test-sn	test-lc	test-sn	test-lc	test-sn	test-lc	test-sn
DN-lc	29.08	35.46	9.67	14.71	20.23	24.71	19.41	25.34
DN-s	28.44	36.96	8.98	15.79	19.48	25.71	18.61	27.39
DN-n	27.42	36.96	8.42	16.17	18.74	25.95	17.11	26.89
DN-sn	26.48	37.22	7.44	16.32	17.84	26.10	15.96	27.33
DN-s∪n	28.44	37.04	9.26	16.17	19.71	25.96	18.84	27.23

Table 4.2: Evaluation results for the models trained on the different datasets. Both ROUGE-scores and BERTScore are F-measures.

As can be seen, the results differ significantly between the different test sets with higher scores on test-sn than on test-lc for all models. The scores on test-sn is more on par with the scores yielded from the CNN/Daily Mail dataset although somewhat lower on all measures. On the test-lc set, there is some difference between the models. Here it seems that the DN-lc model trained on the largest dataset not filtered on novelty or semantic similarity performed the best on all measures followed by DN-s∪n, DN-s, DN-n and DN-sn in that order. On the contrary, DN-sn performed better than all models on the test-sn set with ROUGE-1 and ROUGE-2 scores of 37.22 and 16.32, respectively. As described in Section 3.2, this dataset was the smallest of all datasets with only 38,151 article-summary pairs. Furthermore, the results for the DN-s∪n, DN-n and DN-s models were broadly similar on test-sn, while the results for DN-lc was slightly worse than all other models. One additional observation is that the DN-n

[3] `https://github.com/Tiiiger/bert_score`

model performed somewhat better regarding ROUGE-scores than the DN-s model, while the DN-s model performed somewhat better with regards to BERTScore.

4.3 Model generated examples

In this section, three examples of model generated summaries will be given for illustration purposes. These are presented with their ROUGE-scores as well as BERTScore in Tables 4.3, 4.4 and 4.5 respectively. As can be noted, Example 1 and Example 2 illustrate good candidate summaries with rather high evaluation scores, while Example 3 illustrate inaccurate candidate summaries with low scores.

Example 1

Article: *Inspelningen av "Falsk identitet 5" har just startat i Paris. Serieskaparen och regissören Erich Rochant tar denna gång hjälp av sin franska kollega Jacques Audiard ("En profet", "Rust and bone" och "The sisters brothers"). Audiard, som Guldpalmsbelönades i Cannes 2015 för sitt migrantdrama "Dheepan" ska regissera delar av den femte säsongen av den hyllade franska spionserien som visats i SVT. Han ska också vara med och skriva manus, enligt det franska produktionsbolaget Canal plus. Även Mathieu Kassovitz, som spelar seriens huvudroll Malotru, kommer att slå sig ner i registolen inför kommande säsong. Läs mer: Erich Rochant intervjuad i Filmfredag Bland nytillkomna namn för femte säsongen nämns Louis Garrel som tidigare bland annat spelat Jean-Luc Godard i "Redoutable". "Falsk identitet 5" ska omfatta tio entimmesavsnitt som får premiär under nästa år. Säsong 1-4 ligger på SVT Play, säsong fyra kan ses ytterligare en vecka. Läs mer: Nicholas Wennö ger säsong 4 högsta betyg.*

Reference summary: *Den Guldpalmsbelönade Jacques Audiard regisserar nästa säsong av den hyllade, franska spionserien "Falsk identitet" som kommer tillbaka för femte gången 2020.*

DN-lc: *Falsk identitet - skaparen Erich Rochant tar hjälp av sin franska kollega Jacques Audiard för att ta sig an den femte säsongen av den franska spionserien, rapporterar Filmfredag.*

DN-s: *Falsk identitet - regissören Erich Rochant tar hjälp av sin franska kollega Jacques Audiard för att skapa den femte säsongen av den franska spionserien som sänds i SVT nästa år.*

DN-n: *Franska filmskaparen Erich Rochant tar hjälp av sin franska kollega Jacques Audiard för att göra den femte säsongen av den franska spionserien Falsk identitet 5.*

DN-sn: *Den franska franska spionserien Falsk identitet får en ny säsong. Under nästa vecka börjar inspelningen av den femte och sista säsongen av den hyllade spionserien, rapporterar Filmfredag.*

DN-s∪n: *Den franska spionserien Falsk identitet 5 är klar för femte säsongen. Den här gången är det den franska regissören Erich Rochant som ska regissera delar av den femte säsongen, enligt franska medier.*

Example 2

Article: *De tidigare ryktena att Microsoft tänkte rekrytera sin nya vd internt visade sig stämma. Det blir Microsofts molnchef Satya Nadella som tar över som vd för it-företaget efter Steve Balmer, det framgår av bolagets hemsida. – Satya Nadella är en stark intern*

	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore
DN-lc	46.43	18.52	25.00	28.40
DN-s	52.46	20.34	29.51	35.26
DN-n	50.91	22.64	32.73	39.33
DN-sn	51.72	28.57	27.59	32.16
DN-s∪n	43.75	22.58	31.25	34.88

Table 4.3: Evaluation scores for Example 1

rekrytering som har gjort bra ifrån sig tidigare.Utmaningen framåt för Micorsoft är framförallt att komma framåt med sina mobila satsningar. Kanske behöver företaget rekrytera någon talang som verkligen förstår den utmaningen, säger David Cearly, analytiker på Tech Gartner. Nadella som är född 1967 i Indien har varit verksam i en rad roller på Microsoft sedan 1992. Senast var han chef för bolagets datormoln och företagsgrupp. ”Under denna tid av förändring finns det ingen annan som kan leda företaget på ett bättre sätt”, säger grundaren och styrelseordföranden Bill Gates i en kommentar. Microsoft meddelade samtidigt att Bill Gates avgår som styrelseordförande och kommer stötta Nadella som teknisk rådgivare i bolaget. Ny styrelseordförande blir John Thompson som är vd för Virtual instruments och sedan tidigare sitter med i styrelsen. Satya Nadella själv säger i pressmeddelandet att han känner sig hedrad av uppdraget. ”En stor del av mitt arbete är att påskynda vår förmåga att ta fram innovativa produkter till våra kunder”, säger han.

Reference summary: *Microsoft utser Satya Nadella till ny vd för it–jätten. Bill Gates avgår som styrelseordförande och blir teknisk rådgivare.*

DN-lc: *Microsofts molnchef Satya Nadella blir ny vd för it - företaget Micorsoft. Han ersätter Steve Balmer, som sparkades efter bara två månader på posten. Jag känner mig hedrad, säger Nadella.*

DN-s: *Microsofts molnchef Satya Nadella tar över vd - skapet för it - företaget Virtual instruments. Det framgår av ett pressmeddelande från Microsoft. Vi behöver rekrytera någon som kan ta fram innovativa produkter, säger styrelseordföranden Bill Gates i ett pressmeddelande.*

DN-n: *Microsofts molnchef Satya Nadella tar över som vd för it - företaget Micorsoft efter Steve Balmer, som tidigare varit vd för Virtual instruments.*

DN-sn: *Microsofts molnchef Satya Nadella blir ny vd för it - företaget Microsoft. Det framgår av ett pressmeddelande från bolaget. Det finns ingen annan som kan leda företaget på ett bättre sätt, säger grundaren David Cearly i ett pressmeddelande.*

DN-s∪n: *Microsofts molnchef Satya Nadella blir ny vd för it - företaget Micor Gartner. Han ersätter Steve Balmer, som avgår som styrelseordförande för Microsoft.*

	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore
DN-lc	30.51	17.54	27.12	16.50
DN-s	34.78	11.94	26.09	26.36
DN-n	32.65	17.02	32.65	25.24
DN-sn	31.88	14.93	26.09	21.38
DN-s∪n	57.69	36.00	50.00	39.26

Table 4.4: Evaluation scores for Example 2

Example 3

Article: *Rekordtemperaturer uppmättes i New York och delar av östra USA på tisdagen. I Central Park på Manhattan uppmättes 39,4 grader vilket var mer än det tidigare rekordet för den 6 juli från 1999. Värmeböljan slog också till i Washington DC, Richmond, Virginia, Philadelphia och Pennsylvania, liksom i Montreal i Kanada. Elförbrukningen ökade kraftigt på grund av flitigt användande av luftkonditionering, fläktar och annat avkylande. Nationella vädertjänsten utfärdade varning för att hettan och den höga luftfuktigheten kan vara skadlig för små barn och gamla. Invånarna fick också rådet att dricka mycket vätska, hålla sig i luftkonditionerade rum och vara försiktig med att vistas ute i solen. Befolkningen uppmanades också att titta till släktingar och grannar.*

Reference summary: *Nära 40 grader uppmättes i Central Park på Manhattan i går när östra USA nådde rekordvärme. Nationella vädertjänsten utfärdade varning för att hettan och befolkningen uppmanas att titta till släktingar.*

DN-lc: *Värmeböljan slog till med full kraft på tisdagen. I New York uppmättes 39, 4 grader, den högsta temperatur som uppmätts på över 20 år. I Washington, Virginia och Philadelphia uppmanades invånarna att hålla sig inomhus.*

DN-s: *En värmebölja med temperaturer på 39, 4 grader har slagit till mot New York, Washington DC och andra delar av östra USA. Invånarna uppmanas att hålla sig borta från solen och hålla sig inomhus.*

DN-n: *Värmeböljan och den höga luftfuktigheten kan vara skadlig för små barn och gamla. I Washington, Pennsylvania och Philadelphia fick invånarna uppmanas att dricka mycket vätska och vara försiktiga med att vistas ute i solen.*

DN-sn: *Värmeböljan slog till kraftigt i USA : s huvudstad Washington DC på tisdagen. Invånarna uppmanades att hålla sig till luftkonditionerade rum och hålla sig i luftkonditionerade Rum.*

DN-s∪n: *Värmeböljan och den höga luftfuktigheten i delar av USA har slagit till mot stora delar av landet. I New York, Washington DC, Philadelphia och Pennsylvania fick invånarna rådet att vara försiktiga.*

	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore
DN-lc	26.51	2.47	14.46	17.09
DN-s	24.69	5.06	12.35	21.41
DN-n	21.95	5.00	14.63	9.67
DN-sn	22.22	0.00	11.11	8.49
DN-s∪n	23.08	0.00	10.26	6.76

Table 4.5: Evaluation scores for Example 3

5 | Discussion

This chapter will go into more detail and discuss the results of the filtering and the evaluation. Furthermore, methodological implications, suggestions for future research and ethical considerations are discussed.

5.1 Building summarization datasets

As already pointed out in the introduction and Section 5.2.1, both the size and quality of datasets matter a lot when training machine learning models. The filtering methods applied to the data in this thesis did, as showed in Section 4.2.2, improve model performance. However, at the same time, there was a trade-off between dataset size and more strict filtering thresholds. As could be seen, the DN-sn dataset, which was compiled by filtering on both semantic similarity and novelty, was the smallest of the datasets with only $38,151$ article-summary pairs. Moreover, the model fine-tuned on this dataset achieved the best results on the filtered test data as discussed in Section 5.2.2.

The implications of dataset size and the relationships between quality and size are hard to pin down. There are summarization datasets of various sizes in the literature, suggesting that it does not matter to some degree. For example the BillSum dataset [14] has about $20,000$ training examples while the CNN/Daily Mail dataset[21] has around $280,000$ training examples and the BIGPATENT [34] has around 1.2M training examples. At the same time, it does probably matter to some degree since it is necessary for generalization. With these aspects in mind, the threshold values used for filtering could possibly have been set differently to yield more suitable datasets.

That brings us to another aspect, namely that it is difficult to judge the quality of datasets automatically. In other words, the characteristics that are important and signify a good article-summary pair are hard to quantify. Article and summary lengths and the other surface properties do probably matter to some degree. Moreover, one of the most crucial aspects is that a summary should capture the essential content of the article. Although surface-level n-gram matching gives some hint to the degree of content overlap, it has drawbacks, as pointed out by Zhang et al. [42]. Consequently, it is possible that some high-quality article-summary pairs were excluded, whereas some low-quality pairs were included when filtering on novelty.

On the other hand, filtering on semantic similarity is more promising since it is the overall semantic meaning of an article that is supposed to be formulated in the summary. In this thesis, SBERT was used for assessing semantic similarity. The SBERT model that was used was furthermore multilingual. Multilingual models have been proved to be very effective [7]. Nevertheless, fine-tuning SBERT on Swedish data would probably have had positive effects and resulted in better embedding representations. Furthermore, it is also hard to draw a specific limit for what level of semantic similarity is sufficient to say that an example is of high quality from a quantitative perspective. Therefore, it may have been the case that some article-summary pairs of high quality were excluded, and some low-quality article-summary pairs were included when filtering on semantic similarity.

SBERT is, as the name indicates, aimed to create sentence embeddings. However, in the SBERT documentation[1] it is also stated that SBERT is usable for longer texts with multiple sentences. In this thesis, SBERT was used, as described in Section 3.2, to get embeddings for the complete articles. It is unclear to what extent such article representations capture the most critical content in the article. In a similar vein as with language, it may have been beneficial to fine-tune SBERT on longer texts as well.

5.2 Using the datasets for abstractive summarization

In this section, the evaluation results of the models will be discussed. The main aspects that will be discussed are differences between the two test datasets, differences between all the fine-tuned models, and the performance on the two used evaluation measures.

5.2.1 Test data differences

It is commonly known that low-quality data can harm model performance [29]. This problem is often characterised by the saying ”garbage in garbage out”. As pointed out in Chapter 1 and as illustrated in Appendix A low-quality article-summary pairs was a noticeable problem with the set of data provided for this thesis. Although using such data may have been suboptimal for summarization purposes, it was demonstrated that it might be a great alternative with applied filtering techniques when data is scarce.

The data quality was clearly reflected in the evaluation results. The evaluation scores were significantly higher for all fine-tuned models on the test data that had been filtered on novelty and semantic similarity (test-sn) compared to the test data that had not been filtered as much (test-lc). This is reasonable since it is probably difficult for the models to get high scores if the reference summary has little semantic similarity with the article from which summaries are generated. The same goes for novelty; if a reference summary contains mostly novel words that do not appear in the article, it may be hard for the models to draw inference and generate a candidate summary similar to the reference summary. These results suggest that it is crucial to have high-quality test data to obtain adequate evaluation results and that the suggested filtering methods improve the data quality to some degree.

5.2.2 Model differences

Furthermore, the evaluation results showed that the models fine-tuned on the filtered data performed better on the test-sn set than the model fine-tuned on DN-lc. More specifically, the model fine-tuned on the smallest dataset (DN-sn) with similar characteristics as the CNN/Daily Mail dataset achieved the best results. In other words, more training data did not seem to produce a better model in this case. This may be the case because there are a considerable amount of low-quality article-summary pairs in the larger DN-lc dataset that acts as noise and makes the model learn aspects that are not relevant for summarization.

On the contrary, larger dataset size seemed to matter more when it came to test-lc only filtered on length and compression ratio. Here the DN-lc model achieved the best results while the DN-sn performed the worst in all regards. There may be a reasonable explanation for this stemming from the fact that larger datasets create conditions for better generalization capabilities. So, in this case, the model fine-tuned on DN-lc seems to generalize better to a larger variation of article-summary pairs, including those of low quality. In other words, the model probably learns to ”summarize” articles in a way that yield good results on low-quality article-summary pairs.

Nevertheless, the difference between the fine-tuned models on the different test data is not that big compared to the difference between the results from the different test datasets. Thus,

[1] `https://www.sbert.net/examples/applications/computing-embeddings/README.html`

applying filtering methods as proposed in this thesis is most effective when applying them on the test data but may also improve model performance when applied to the training data. However, this might hamper performance on low-quality examples and negatively affect the generalization ability.

5.2.3 Measure differences

A third aspect of the evaluation results is regarding the difference between ROUGE-scores and BERTScore. The model trained on DN-s, which had only been filtered on semantic similarity computed from SBERT, yielded somewhat better performance regarding BERTScore than the other models, especially DN-n and DN-lc. On the other hand, the model trained on DN-n yielded slightly better performance than the DN-s regarding ROUGE scores on the test-sn set. It is reasonable to suspect that models fine-tuned on a dataset filtered on novelty will also yield better ROUGE-scores since novelty and ROUGE-scores are complementary and measure surface characteristics. Likewise, it is reasonable to suspect that a model trained on data filtered on semantic similarity will yield better BERTScores since BERTScore is a measure based on semantic similarity, although somewhat different from SBERT.

5.3 Model implementation

In this section, methodological aspects of the model implementation and the evaluation are discussed.

5.3.1 Model fine-tuning

Regarding the fine-tuning of the warm-started model, it is possible that other hyper-parameter values would have yielded better results. The number of training steps and warm-up steps were, as mentioned, adjusted for the different dataset sizes, but no extensive investigation was done to find the most optimal parameters. The training process, more specifically the training and validation loss, was just monitored to make sure the models were not overfitting. On the other hand, some of the models could probably have been fine-tuned even longer. Furthermore, the relatively small batch size certainly affected training time since less parallelization was enabled, but as Bengio [5] points out, test performance is not affected that much when having small batch sizes. The beam search parameters for the warm-started encoder-decoder model could also have been set differently, which possibly would have yielded better results. No extensive research was done to optimize these parameters either.

The fact that the model fine-tuned on the CNN/Daily Mail dataset yielded better results than in the original implementation by Rothe et al. [31] bolsters the assumption that these hyper-parameters did not affect test results significantly. Either way, the primary purpose of this thesis was not to optimize parameter values for performance but to investigate the proposed filtering methods and their effects on model performance. Therefore, more focus was put on making the different models comparable, to each other, by having approximately[2] the same settings for all models.

5.3.2 Evaluation measures

As pointed out by Zhang [42] automatic metrics based on n-gram matching has severe limitations. They rely on the assumption that content selection is the single thing that makes a good summary. Other text quality aspects such as fluency, grammaticality and coherence are overlooked. Content selection with n-gram based measures does also solely rely on lexical overlap as demonstrated in Section 2.6.2 which is not optimal in the abstractive summarization task.

[2]The only difference was that the English model had higher minimum and maximum length in beam search to account for the fact that summaries were slightly longer in the CNN/Daily Mail dataset.

For these reasons, the validity of ROUGE measures and other similar n-gram based measures is questionable, especially for tracking the progress and claiming state-of-the-art results based on these metrics.

To mitigate some of the problems with ROUGE, BERTScore, was also used to evaluate candidate summaries. BERTScore is still a relatively new metric, and the applicability to abstractive summarization is relatively unexplored. Nevertheless, utilizing contextualized embeddings shows great promise and is probably the way to go since it addresses the most pressing limitations with n-gram matching metrics. With that said, BERTScore may also have limitations. Text quality aspects such as fluency, grammaticality and coherence may still be overlooked, although they might be captured in the embeddings to some degree. The implementation of BERTScore in this thesis was furthermore done with a multilingual model. Once again, a Swedish model, unfortunately not available for BERTScore, might have contributed to more accurate embedding representations.

5.4 Future research

Abstractive summarization is a difficult task. The existing models are by no means perfect and still undergoes constant improvements. They do, however, require high-quality data to be trained and evaluated on. This aspect was the focus of the current work in which the CNN/Daily Mail dataset was used as a reference point regarding characteristics when filtering the DN data. It was assumed that this dataset had high-quality summaries of articles since they were hand-written for summarization purposes, although humans may also have flaws in writing summaries. For this reason, the goal was to obtain similar characteristic as the CNN/Daily Mail dataset. Other high-quality datasets may have different characteristics than the CNN/Daily Mail dataset. Therefore, it would be interesting to thoroughly compare multiple summarization datasets to get a better sense of what constitutes a high-quality dataset in future research. Moreover, other potential characteristics to judge the quality of individual document-summary pairs may also be needed. Here, qualitative work would also be essential to bring new insights and validate automatic methods.

This thesis has also been limited to the domain of news articles, which is convenient for abstractive summarization since the texts are not that long and since it most often exists summaries/highlights or preambles. For future research, it would be interesting to apply the proposed filtering methods to other types of document-summary pairs. These could also be longer, as new model architectures are emerging to tackle the text length limitations of the Transformer and BERT, e.g. the Longformer [4].

Finally, there is also the need for better evaluation measures. Incorporating grammar, fluency, and other textual qualities would be of great interest, just as the improvements of content selection measures. Doing so may lead to more valid results and comparisons between models. They would, of course, need to correlate well with human judgement. Therefore extensive qualitative work with extrinsic evaluation is also needed regarding this.

5.5 Ethical considerations

When training machine learning models on large amounts of text data, there is always a risk of inherent bias in the data. It is crucial to acknowledge this when deploying models for actual use since it could have negative implications if users are exposed to such biases. Furthermore, regarding news, factual correctness may also be an aspect to consider. Therefore, it is important to be wary of the sources of the data. In this thesis, news data from DN was used. This is one of the most renowned newspapers in Sweden, known for its high-quality journalism and independent liberal stance. However, humans are flawed, and one can not completely rule out the possibility of some bias in this data. Text summarization systems are by no means perfect either, which is also important to consider when deploying them.

6 | Conclusion

This thesis aimed to explore methods for filtering out low-quality article-summary pairs from a large set of Swedish news articles provided for this thesis. In doing so, high-quality datasets could be compiled and used for training and testing abstractive summarization models. This was achieved using the CNN/Daily Mail dataset as a reference regarding specific data characteristics and filtering to obtain similar characteristics. The primary filters applied, except article and summary length and compression ratio, were novelty and semantic similarity between articles and summaries. Five datasets were compiled. One was filtered only on length and compression ratio (DN-lc). The others (DN-s, DN-n, DN-sn and DN-s∪n) were further filtered on semantic similarity or novelty or both. DN-sn was the dataset most similar to the CNN/Daily Mail dataset in terms of the used characteristics.

The aim of this thesis was also to implement summarization models to see what effects the filtering had on model performance. This was done by warm-starting an encoder-decoder model with checkpoints from a Swedish BERT model and fine-tuning it on the yielded datasets, respectively. The model was implemented and fine-tuned using the HuggingFace Transformers library. Two test datasets were used for evaluating the fine-tuned models with ROUGE scores and BERTScore. One had been filtered on semantic similarity and novelty (test-sn), and the other had been filtered on length and compression ratio (test-lc).

The evaluation results of the five different models showed that the quality of the test data matters the most for how the models perform. The models' evaluation results differed slightly from each other for each test dataset. For the test-sn, the model fine-tuned on the most filtered DN-sn subset performed the best, while on test-lc, the model fine-tuned on the least filtered subset DN-lc performed the best. This probably has to do with the relationship between dataset size and quality, affecting how the models generalize to unseen data. Compared to model performance on the CNN/Daily Mail dataset, the best Swedish model performed slightly worse regarding all metrics.

The proposed filtering methods seem promising, especially for cases where the data is in some sense suboptimal for the task of summarization and contain many low-quality examples. However, the space of text summarization is still relatively unexplored, and future research is needed to explore further dataset characteristics and filtering techniques as well as evaluation measures.

Bibliography

[1] Jay Alammar. *The Illustrated Transformer [Blog post]*. 2018. URL: http://jalammar.github.io/illustrated-transformer/ (visited on 05/04/2021).

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

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

[4] Iz Beltagy, Matthew E. Peters, and Arman Cohan. *Longformer: The Long-Document Transformer*. 2020. arXiv: 2004.05150 [cs.CL].

[5] Yoshua Bengio. *Practical recommendations for gradient-based training of deep architectures*. 2012. arXiv: 1206.5533 [cs.LG].

[6] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].

[7] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. *Unsupervised Cross-lingual Representation Learning at Scale*. 2020. arXiv: 1911.02116 [cs.CL].

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[10] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory." In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[11] Jeremy Howard and Sebastian Ruder. *Universal Language Model Fine-tuning for Text Classification*. 2018. arXiv: 1801.06146 [cs.CL].

[12] Daniel Jurafsky and James H. Martin. *Speech and Language Processing (3rd edition draft)*. Prentice-Hall, Inc., 2020.

[13] Wafaa S. El-Kassas, Cherif R. Salama, Ahmed A. Rafea, and Hoda K. Mohamed. "Automatic text summarization: A comprehensive survey." In: *Expert Systems with Applications* 165 (2021), p. 113679. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2020.113679. URL: https://www.sciencedirect.com/science/article/pii/S0957417420305030.

[14] Anastassia Kornilova and Vlad Eidelman. *BillSum: A Corpus for Automatic Summarization of US Legislation*. 2019. arXiv: 1910.00523 [cs.CL].

[15] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. 2019. arXiv: 1910.13461 [cs.CL].

[16] Chin-Yew Lin. "ROUGE: A Package for Automatic Evaluation of Summaries." In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. URL: https://www.aclweb.org/anthology/W04-1013.

[17] Yang Liu and Mirella Lapata. *Text Summarization with Pretrained Encoders*. 2019. arXiv: 1908.08345 [cs.CL].

[18] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. *Effective Approaches to Attention-based Neural Machine Translation*. 2015. arXiv: 1508.04025 [cs.CL].

[19] Martin Malmsten, Love Börjeson, and Chris Haffenden. *Playing with Words at the National Library of Sweden – Making a Swedish BERT*. 2020. arXiv: 2007.01658 [cs.CL].

[20] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL].

[21] Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos santos, Caglar Gulcehre, and Bing Xiang. *Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond*. 2016. arXiv: 1602.06023 [cs.CL].

[22] Andrei Paraschiv and Dumitru-Clementin Cercel. "UPB at GermEval-2020 Task 3: Assessing Summaries for German Texts using BERTScore and Sentence-BERT." In: *SwissText/KONVENS*. 2020.

[23] Jeffrey Pennington, Richard Socher, and Christopher Manning. "GloVe: Global Vectors for Word Representation." In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://www.aclweb.org/anthology/D14-1162.

[24] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. *Deep contextualized word representations*. 2018. arXiv: 1802.05365 [cs.CL].

[25] Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. *ProphetNet: Predicting Future N-gram for Sequence-to-Sequence Pre-training*. 2020. arXiv: 2001.04063 [cs.CL].

[26] A. Radford and Karthik Narasimhan. "Improving Language Understanding by Generative Pre-Training." In: 2018.

[27] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. "Language Models are Unsupervised Multitask Learners." In: (2019).

[28] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 2020. arXiv: 1910.10683 [cs.LG].

[29] Thomas C. Redman. "If Your Data Is Bad, Your Machine Learning Tools Are Useless." In: *Harvard Business Review* (Apr. 2, 2018).

[30] Nils Reimers and Iryna Gurevych. *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. 2019. arXiv: 1908.10084 [cs.CL].

[31] Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. *Leveraging Pre-trained Checkpoints for Sequence Generation Tasks*. 2020. arXiv: 1907.12461 [cs.CL].

[32] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A unified embedding for face recognition and clustering." In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2015). DOI: 10.1109/cvpr.2015.7298682. URL: http://dx.doi.org/10.1109/CVPR.2015.7298682.

[33] Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. *MLSUM: The Multilingual Summarization Corpus*. 2020. arXiv: 2004.14900 [cs.CL].

[34] Eva Sharma, Chen Li, and Lu Wang. *BIGPATENT: A Large-Scale Dataset for Abstractive and Coherent Summarization*. 2019. arXiv: 1906.03741 [cs.CL].

[35] Josef Steinberger and Karel Jezek. "Evaluation Measures for Text Summarization." In: *Computing and Informatics* 28 (Jan. 2009), pp. 251–275.

[36] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. *Sequence to Sequence Learning with Neural Networks*. 2014. arXiv: 1409.3215 [cs.CL].

[37] Wilson L. Taylor. "Cloze Procedure: A New Tool for Measuring Readability." In: *Journalism Quarterly* 30.4 (1953), pp. 415–433. DOI: 10.1177/107769905303000401. eprint: https://doi.org/10.1177/107769905303000401. URL: https://doi.org/10.1177/107769905303000401.

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL].

[39] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, ukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. 2016. arXiv: 1609.08144 [cs.CL].

[40] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*. 2020. arXiv: 1906.08237 [cs.CL].

[41] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. *PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization*. 2020. arXiv: 1912.08777 [cs.CL].

[42] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. *BERTScore: Evaluating Text Generation with BERT*. 2020. arXiv: 1904.09675 [cs.CL].

A | Low-quality summarization data

Below, examples of article-summary pairs with low quality are presented (in Swedish). These were hand-picked from randomly sampling among the 821,405 article-summary pairs filtered on length and compression ratio. The common theme among these examples is that it would be challenging for a human to create a summary similar to the written preamble. Thus it would be even more challenging for a computer.

Example 1

Summary: *Med hänsyn till den utdragna återhämtningen och den svaga arbetsmarknaden torde utrymmet för löneökningar vara mycket begränsat.*

Article: *Det sade förste vice riksbankschefen Svante Öberg när han talade i Stockholm på fredagen, enligt ett pressmeddelande från Riksbanken. Öberg pekade även på att det för närvarande råder stor osäkerhet om den framtida ekonomiska utvecklingen. "Det gör att det kan vara svårt för parterna att i avtal binda upp sig för överenskommelser över längre tidsperioder", sade han.*

Example 2

Summary: *De flesta utländska hästarna i Elitloppet har varit på plats i Norden i många veckor i väntan på loppet. När flyget landar på lördag kväll på Arlanda med franske Oyonnax och italienska Ilaria Jet ombord är alla på plats.*

Article: *Amerikanske favoriten Lucky Jim kom i mitten av förra veckan och trivs på Erikssund säteri. De två andra från Nordamerika, Enought Talk och Define The World, kom ännu tidigare. Enought Talk startade både i Finlandia Ajo och Oslo Grand Prix. Define The World var med i Oslo och gjorde en fin insats som nyanländ. Båda har efter det tränats på Solvalla. Franske Nimrod Borealis gick direkt från Oslo Grand Prix till Sverige och är uppstallad på Stefan Melanders gård utanför Enköping. Italienska Lisa America körs av Jorma Kontio och Kontio har haft henne på sin gård i Enköping ett tag i väntan på start i årets största uppgift. Leif Berg*

Example 3

Summary: *De enklaste, billigaste och bästa ångkokarna finns att köpa i butiker som säljer asiatiska livsmedel.*

Article: *Kycklingfilé med ångkokt broccoli och rostade jordnötter Portioner: 4 4 kycklingfiléer (à 150 g) 750 g broccoli 1 tsk salt 1 krm nymald svartpeppar 1 msk olivolja 1 dl sweet chilisås 4 msk japansk soja 4 msk rostade jordnötter Tillbehör: 50 g bröd/pers 1. Ansa och skär broccolin i buketter. Lägg dem i en ångkokare. 2. Krydda kycklingfiléerna med salt och peppar. Stek filéerna i olja i en stekpanna under lock 6–8 min på varje sida till 68 grader innertemperatur. 3. Ångkoka broccolibuketterna 3–5 min. Blanda sweet chilisås och soja. 4. Lägg kycklingfiléer och broccolibuketter på tallrikar. Ringla över såsen. Strö över nötterna och servera med bröd. Näring/portion: Ca 540 kcal, 15 g fett, 52 g protein, 45 g kolhydrater.*

Example 4

Summary: *Martin Hellström, litteraturforskare och barnboksrecensent i DN, är en av tre stipendiater som får 85 000 var ur stiftelsen Martha Sandwall-Bergströms stipendiefond. De andra stipendiaterna är Katarina Kieri och Kelly Hübben.*

Article: *1 Vad ska du använda pengarna till? – Jag har sju hungriga barn, så de går nog åt. Nej, inte mina biologiska barn, alltså! De är sju barn som forskar tillsammans med mig. Vi läser Maria Gripes böcker och har litteratursamtal som jag spelar in. Meningen är att förstå barnlitteraturen bättre genom barnens egna perspektiv. Vi har träffats regelbundet under två år. Det går åt mycket fika! 2 Varför blev det just Maria Gripes böcker? – Barnen var 9–11 år när vi började, och det är bra att läsa Gripe så: hennes tidiga böcker handlar om yngre barn, sedan blir de allt äldre, och hennes sista böcker, Skuggserien, är rena ungdomsböcker. Det är bra, barnen växer i takt med böckerna. Just nu läser vi "Hugo och Josefin". 3 Vad betyder stipendiet för dig? – Barnen är med som uttolkare, inte som forskningsobjekt. Jag vill inte sitta på mitt rum och tänka själv, litteraturen är ju barnens. Deras röster är viktigast. Jag ser det som ett erkännande av barnens betydelse.*

Example 5

Summary: *5 frågor till entreprenören och cykelkungen Salvatore Grimaldi, 62 år, som framträder vid ett seminarium i Kungliga Ingenjörsvetenskapsakademiens (IVA) konferenscentrum i dag om behovet av mångfald i arbetslivet.*

Article: *Vad tänker du ta upp i ditt tal? - Jag tänker beskriva hur invandringen har förändrat Sverige till det bättre. Mina föräldrar hittade jobb på Asea genom ett rekryteringskontor i Taranto i södra Italien. Man kan faktiskt säga att de blev headhuntade. Svensk industri ropade efter utbildad arbetskraft till sina verkstäder. Det italienska tillskottet betydde mycket. Den söta svenska limpan finns snart inte längre. Nu äter vi gott, italienskt bröd. - Italienarnas styrka är att de kan design, kultur, mat, och kan ändra sig snabbt. Svenskarna kan organisera, fullfölja sina projekt och visa bra resultat. Vi kompletterar varandra! Är svenska företag bra eller dåliga på mångfaldsfrågor? - Sverige och svenskarna har utvecklats i och med globaliseringen, men det gäller att man kan ta till sig det bästa. Mångfald är berikande. Finns det också exempel där mångfald inte fungerar så bra? - Jag tror att människor är sunda i grunden. Ser man något som inte är bra så bryr man sig inte om att anamma det. Vart skulle du kasta blickarna i dag om du letar efter arbetskraft? - Till Östeuropa, men då får man skynda sig, för de absorberas snabbt. Mjukvaruindustrin kan hitta begåvningar som tänker på ett annat sätt i Indien. Var ser du för utmaningar för svensk industri i framtiden? - Kostnaderna för att få duktigt yrkesfolk är ett problem, liksom kunnandet. Se på miljöfrågorna, där Al Gore tar så stort utrymme i dag. Vi var pionjärer inom kärnkraften, men det kunnandet håller på att försvinna. Jag tycker att vi ska utveckla kärnkraften som en grön teknik. I framtiden kan jag mycket väl tänka mig att man kan återanvända utbränt kärnbränsle. - Cyklar borde också kunna bli en framtidsbransch när våra samhällen blir tätare bebodda.*

Example 6

Summary: *Maria Andersson, Hammarby Sjöstad Petra Fritzson, Söder Anna Arodén, Upplands Väsby Tobias Andersson, Hägersten Lena Molander, Hässelby.*

Article: *– Jo det har absolut hänt. Jag försökte hålla regelbundna tider, äta frukost för att få igång ämnesomsättningen tidigt och cykla till jobbet. Men jag gillar mat och äter mycket, men oftast inte efter middagstid. Jag försöker utesluta kolhydrater och äta mer sallad och fisk. – Ja. Jag tänker mycket på helheten med kost och motion. Att man ska äta långsamma*

kolhydrater och bra fetter, som från avokado och olivolja. Är man uppe sent på kvällarna och dricker mycket vuxendricka och äter fyllekäk då går man inte ned i vikt har jag märkt. Men just nu är jag på bäbisdieten och äter allt jag vill. – Ja någon gång. Det var mest att jag inte åt så mycket socker och ingen snabbmat. Sen tränade jag. Jag gick väl ned något, men jag äter det jag känner för och det som är gott; husmanskost och lite blandat. Men jag tror att vi äter för mycket idag. – Ja det har jag gjort. Jag vägde 92 kilo och nu väger jag 75 kilo. Jag åt mindre mat och onyttigheter, som godis och kaloririka saker. Det tog ett och ett halvt år ungefär. Man har hört om GI, Atkinson, fruktdieten och allt vad det är. Men jag är mer ute efter att ändra ett beteende. – Ja jag försökte under ett helt år och gick ner tio kilo. Det var för tio år sen och jag har inte bantat sedan dess. Jag hittade dieten i en damtidning och åt hela kostcirkeln varje dag med lite godis och lite socker. Kroppen vande sig efter ett tag.
Enkät: Annelie Ivarsson

Example 7

Summary: *Invandrares och svenskars gemensamma värdegrund måste stärkas. Det säger regeringen, mot bakgrund av de senaste veckornas inflammerade debatt kring konstnären Lars Vilks bild av profeten Muhammed som rondellhund.*

Article: *Under 2008 satsar integrations- och jämställdhetsdepartementet på dialog med ett brett spektrum av samhällsaktörer för att stävja framväxten av extremistiska, rasistiska och fundamentalistiska grupper som ifrågasätter samhällets grundläggande normer. Sveriges befolkning måste kunna enas kring demokrati och mänskliga rättigheter som universella normer och basen för ömsesidig respekt i samhället, betonar departementet. Till våren aviserar regeringen en sammanhållen strategi för integrationspolitiken under resten av mandatperioden. Bland annat åtgärder för att stärka utrikes födda personers konkurrenskraft på arbetsmarkanden. Departementet vill i vår också slå ihop de olika diskrimineringsombudsmännen - JämO, HomO, DO och HO - i en gemensam ombudsmannamyndighet, med en samlad diskrimineringslagstiftning att förvalta.*

Example 8

Summary: *Börsen öppnade med sjunkande kurser i linje med de ledande Europabörserna. Nedgången höll sedan i sig medan placerarna väntade på inköpschefsindex från USA (ISM-index). Indexsiffran, som publicerades vid 16-tiden var klart lägre än väntat och sänkte börshumöret rejält.*

Article: *Sax-index hade vid stängning backat med 1,5 procent till 158,2 medan OMX-index hade fallit med 1,7 procent till 522,3. Omsättningen uppgick till 7,6 miljarder kronor. De flesta storbolagen sjönk. Ericsson backade med 2,9 procent till 8:35 kronor, Nokia tappade 1,2 procent till 129:50 kronor, Telia Sonera föll med 1,8 procent till 32:60 kronor och Astra Zeneca föll med 2,2 procent till 317:50 kronor. Inom finanssektorn föll Nordea med 2,3 procent till 37:70 kronor medan Föreningssparbanken backade med lika mycket till 108 kronor. Assa Abloy föll med 3,2 procent till 75 kronor och Securitas tappade 3,1 procent till 79:50 kronor. På plussidan fanns ABB som klättrade 0,8 procent till 26:70 kronor. ABB sade på tisdagen att det väntar sig ett domstolsbeslut rörande dess asbestuppgörelse snart. Micronic Laser Systems, som fanns bland måndagens vinnare, fortsatte upp med 19,3 procent till 47 kronor. Bolaget presenterade en ny order på tisdagen. Stockholmsbörsens ägare OM steg med 8,8 procent till 62 kronor efter måndagens besked om ett nytt åtgärdsprogram. Operatören Song Networks steg med 1,3 procent till 38 kronor. Bolaget har tecknat ett treårigt avtal med Riksskatteverket värt 50 miljoner kronor.*