ERASMUS UNIVERSITY ROTTERDAM

ERASMUS SCHOOL OF ECONOMICS

MASTER OF SCIENCE THESIS

BUSINESS ANALYTICS AND QUANTITATIVE MARKETING

LATS: Low-Resource Abstractive Text Summarization

Author
C.B. VAN YPEREN
413732

Supervisor
Dr. F. FRASINCAR

Second assessor
Dr. K. GRUBER

Abstract

In this paper we investigate the effectiveness of curriculum learning strategies and data augmentation techniques on the state-of-the-art abstractive text summarization method PEGASUS to increase performance with low-resource training datasets. Curriculum learning strategies require a method to sort training data according to difficulty, which, to the best of our knowledge, does not exist in current literature for text summarization. Therefore, we introduce a novel text-summary pair complexity scoring algorithm along with two simple baseline difficulty measures. We find that our novel complexity sorting method consistently outperforms the baseline sorting methods. The Baby-Steps curriculum learning strategy with this sorting method leads to performance improvements of 5.7%, and when combined with the data augmentation techniques to 6.5%, compared to a baseline of with curriculum learning or data augmentation and measured in a combined ROUGE F1-score.

Keywords— Abstractive Text Summarization, Curriculum Learning, Low-Resource Summarization, Complexity Scoring

April 27, 2021

ERASMUS UNIVERSITEIT ROTTERDAM
ERASMUS SCHOOL OF ECONOMICS

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The amount of information available to all individuals with an Internet connection is increasing at an incredible rate. Processing even a tiny proportion of the available information is a daunting task for many of us. As a large part of this information is text-based, text summarization (Shi et al., 2020) can help alleviate this task and thereby help to solve many real-world problems.

Text summarization comes in two forms, extractive and abstractive summarization (Gambhir and Gupta, 2017). The former takes words and/or sentences from the original text and pastes those together to generate a summary whereas the latter method can additionally generate new words and change sentences and thus produce a summary using a much broader vocabulary. Generally, the latter technique results in more grammatically fluent summaries and is therefore preferable. Because of this, we focus our work on abstractive summarization.

As the amount of digital information is increasing, the sizes of datasets available for researchers are growing. A vast majority of the prevalent literature concerned with Natural Language Processing tasks, including the text summarization task, explore the performance of various models and techniques based on large English training corpora of several hundred thousand training examples. Although this is a sensible approach in academic literature, in real-world applications it is often difficult and costly to collect a vast amount of example text-summary pairs to train a summarization model. Therefore, there is a mismatch between the datasets used in academic research and those commonly available in potential real-world applications (Zhang et al., 2019). It would be of great value to close this gap between the state-of-the-art results achieved in a research setting and the real-world performance of these models when faced with dataset sizes more common to many organisations worldwide. Closing this gap would help unlocking the massive potential that lies in the huge amount of data dispersed over many organisations in relatively small chunks.

When faced with a low-resource situation where a limited amount of training data is available, and the possibility of gathering more data is excluded, there are two approaches to explore in order to find solutions to this data scarcity problem. First of all, the available data can be more effectively exploited by training a model in a more efficient manner. A possible approach to achieve this is a curriculum learning strategy (Bengio et al., 2009). Curriculum learning is captured in the idea that a machine learning model can be more efficiently trained by presenting the training data ordered from easy samples to more complex ones instead of in random order. In addition, the dataset can be artificially expanded by creating new samples based on the available data such that more training data will become available without the necessity to collect more data. This strategy of data augmentation has been applied in multiple research domains and with various types of data (Wang et al., 2020; Aftab and Siddiqui, 2018; Ramirez et al., 2019). Here, data augmentation techniques can be applied to generate altered versions of the text-summary pair training data from which the model can extract new information which leads to further performance improvements.

This research aims to make the abstractive summarization models more accessible to use cases with smaller amounts of training data by exploring the effectiveness of curriculum learning strategies and data augmentation techniques that are aimed at increasing the training efficiency of these models.

We utilise the pre-trained PEGASUS model introduced by Zhang et al. (2019) and the CNN/DailyMail dataset (Hermann et al., 2015) for further model fine-tuning to optimise the model for the text summarization task. We select this model because of the state-of-the-art performance that it achieved after extensive training and because of the relatively high performance it achieved in the zero-shot setting (a situation where the model is only pre-trained and not fine-tuned) (Zhang et al., 2019). The fine-tuning of the model is considered for dataset sizes of 10, 100, and 1,000 training samples. We compare the performance of the model without any additional techniques with two curriculum learning strategies and three complexity sorting methods per strategy. In order to apply curriculum learning strategies on text-summary pairs, a method to determine the complexity of the specific summarization task is required. To the best of our knowledge, no such method exists in current literature, therefore we introduce a novel complexity scoring algorithm along with two baseline sorting methods which are based on the length of the input text and the percentual text reduction between the original text and the summary.

We found that the One-Pass curriculum learning strategy and the Baby-Steps curriculum learning strategy in combination with our novel complexity scoring method achieves 4.0% and 5.7% better performance measured in a combined ROUGE F1 (Lin, 2004) score when compared to the baseline performance of the model with a 1,000 sample dataset, respectively. Additionally applying Easy Data Augmentation (EDA) (Wei and Zou, 2019) techniques increased the performances to 6.4% for the One-Pass curriculum and 6.5% for the Baby-Steps curriculum. Furthermore, we found that the complexity scoring algorithm consistently outperforms the length and reduction baseline sorting methods for the dataset consisting of 1,000 samples.

For the setting with 10 and 100 sample datasets we found that the curriculum learning strategies in combination with any of our sorting algorithms had a minimal effect due to the limited variation in the ordering possible with such low amount of data points. Hence, we conclude that a minimum sample size somewhere between 100 and 1,000 is required to benefit from the curriculum learning strategy. Applying EDA techniques did result in performance gains for these sample sizes, however only in combination with the Baby-Steps curriculum learning strategy. Finally, an examination of the best model generated summary in these low-resource settings leads to our conclusion that the summaries are promising, but of insufficient quality to be applied in any real-world applications. The code of our implementation is available on https://github.com/CBvanYperen/LATS.

The remainder of this paper is organised as follows. Chapter 2 describes the existing literature concerning abstractive text summarization and curriculum learning strategies. We describe the data we used and collected in order to perform our research in Chapter 3. Chapter 4 lays out our methodology, where we describe the model type, pre-training and training methods, as well as the performance evaluation techniques. In Chapter 5 we discuss the results and draw conclusions based on our findings. Last, in Chapter 6.1 we conclude our paper and mention subjects for future research.

Chapter 2

Related work

In this Chapter we outline the current related literature. The Chapter is divided into three Sections which will address the literature concerned with abstractive text summarization, curriculum learning, and low-resource text summarization. In Section 2.1, a number of notable approaches to abstractive text summarization are explored, each of which resulted in state-of-the-art results at the time of publishing. Section 2.2 describes the existing curriculum learning strategies. Last, Section 2.3 shows other notable research that explored text summarization specifically in a low-resource setting.

2.1 Abstractive text summarization

One of the earlier work on abstractive text summarization was performed by Banko et al. (2000) when he used traditional phrase-based machine translation approaches to generate newspaper article headlines. The machine translation approaches were used because Banko et al. considered the problem of summarization as analogous to translation where the target language in this case was still English, the same as the original language, however a much more compact form of English. Therefore the original text was translated to this "Compact English" language and thus effectively making abstractive summarizations of the text which could be used as headlines. This way of looking at the text summarization problem is still applicable in more recent research as some of the most common text summarization frameworks were originally applied in the machine translation domain.

Cohn and Lapata (2008) approached the text summarization problem through sentence compression. At the time, sentence compression was already applied to text summarization, but the compression of the sentences was limited to word deletion. Therefore, the compression approaches up until then were all of extractive nature. Cohn and Lapata rewrote the text into a summary by additionally using word substitution, reordering, and insertion, thus making the method an abstractive one. In their work, the authors use weighted tree-transformation rules where the appropriate weights are trained on a novel dataset created by having annotators compress sentences.

Another methodology to create abstractive text summaries was proposed by Woodsend et al. (2010). This was a solution that contained a quasi-synchronous grammar (Smith and Eisner, 2006) approach to capture rendering preferences (e.g., paraphrases and compressions) to ensure a grammatically correct and fluent summary output. In contrast to earlier work, where the compression and extraction problem was considered merely at a sentence level, Woodsend et al. allow their methods to operate at a phrase level. Since phrases are shorter than sentences, the text compression comes more naturally as only important phrases are selected instead of whole sentences which often would also include ample unimportant information which was difficult to filter out when compressing those sentences.

The approaches described thus far all rely on linguistic knowledge to design effective solutions. Collobert et al. (2011) greatly decreased this dependency on the researcher's linguistic knowledge when they proposed deep learning as an alternative approach for several NLP problems. Ever since, researchers have considered deep learning as a viable approach to text summarization. The tendency towards a deep learning architecture was amplified by Sutskever et al. (2014) who's work introduced sequence-to-sequence (seq2seq) models with an encoder-decoder architecture based on RNNs (Rumelhart et al., 1986). This became the dominant framework for abstractive text summarization (Zhang et al., 2019) and is also applied plentifully in other NLP tasks such as machine translation.

Various researchers have achieved state-of-the art results by building further upon seq2seq models with an encoder-decoder architecture. The work of Rush et al. (2015), a collaboration between Harvard University and the Facebook AI Research team, reported notable results with an Attention-Based Summarization approach. This approach was developed with the encoder-decoder architecture as its foundation and used the attention-based encoder of Bahdanau et al. (2015), whose work introduced an extension to the encoder-decoder model, which uses a latent soft alignment over the input text to gain additional contextual information for the summary.

The work of Nallapati et al. (2016) also built further upon the attentional RNN encoder-decoder model proposed by Bahdanau et al. (2015). The authors implement several improvements to the original model which solve certain abstractive text summarization-specific problems that were not addressed in the model of Bahdanau et al., because it was developed for machine translation. They applied the large vocabulary 'trick' (Jean et al., 2015) which reduces the computational bottleneck in the softmax layer by focusing on the most frequent words in the text. In addition, a more elaborate way of embedding words was presented that included additional features such as parts-of-speech tags, named-entity tags, and Term Frequency (TF) and Inverse Document Frequency (IDF) statistics in order to better capture the keywords in a document. Last, the authors present a novel switching generator-pointer architecture to improve the handling of words that are unseen or rare in the training data text but could be essential for summarising a new text. At the time it was common that, if a model would not be able to generate a word required for the summary, it would output an '[UNK]' token as a placeholder, which resulted in ineligible summaries. The generator-pointer architecture allows the model to switch between generating output from its training data vocabulary, as long as it was able to generate output based on its training data vocabulary, and pointing to a word in the input text to use in the summary whenever the model encountered a word not sufficiently represented in the training data vocabulary. The model uses an activation function to determine whether it should generate or copy a word, and an attention layer is used to determine which word it should copy if required. Both of these are trained during the training process and executed at every step of the decoding process, i.e., after each word added to the summary the model determines whether the next word should be generated or copied.

Further work on eliminating issues not yet addressed by abstractive text summarization models was performed by See et al. (2017). They proposed a hybrid pointer-generator framework, which is a combination of extractive and abstractive text summarization as it has the ability to copy words from the original text through the pointer mechanism and can also generate new words not yet seen in the original text. The advantage of copying words from the original text improves accuracy and handling of out-of-vocabulary words, that is, words that are present in the input text but have not yet been seen by the model in any of the training data. Furthermore, a commonly occurring problem in text summarization is the repetition of words in the summary. See et al. propose a new variant of the coverage vector (Tu et al., 2016) to keep better track of the text parts that have already been summarised in order to prevent repetitions in the summary. These novel methods showed an improvement compared to the work of Nallapati et al. (2016) for the CNN/DailyMail dataset (Hermann et al., 2015).

A recently introduced model is the PEGASUS model and was proposed by Zhang et al. (2019). The model showed state-of-the-art text summarization results measured by ROUGE F1 scores across all twelve considered datasets. The PEGASUS model is a seq2seq encoder-decoder based on Transformers (Vaswani et al., 2017) instead of RNN's which were used by Nallapati et al. (2016). An important contribution to their state-of-the-art results was their novel pre-training method. They pre-trained the model by removing one or more sentences from the input text and letting the model generate the missing sentences based on the remaining text. As this pre-training method does not require training texts to have parallel summaries, it was possible to pre-train the model on a massive dataset consisting of ∼1.85 billion texts. The vast majority of the data (∼1.5B) is part of the HugeNews dataset consisting of articles from news and news-like Web sites.

2.2 Curriculum learning

The concept of curriculum learning was introduced by Elman (1993) when he showed that a neural network could be trained better at the task of learning a grammatical structure by first training the model on simple sentences and gradually adding more complex sentences. The final model showed high performance levels even on complex sentences, whereas a model that was trained with all (simple and complex) sentences in random order showed very poor performance. A generalization of this idea is that a network can learn a task better when first presented with easier training data for the task followed by training data gradually increasing in complexity.

The concept of curriculum learning has gained more attention in recent years because, i.a., the complexity of the problems that are considered has increased (Bengio et al., 2009). Bengio et al. (2009) shows that a curriculum learning strategy can improve vision and language tasks. Zaremba and Sutskever (2014) show that when considering recurrent neural networks with LTSM units in a sequence-to-sequence model, as introduced in Sutskever et al. (2014), the naive curriculum strategy described in the work of Bengio et al. (2009), where training buckets gradually increase in complexity, did not consistently improve the network performance. The authors proposed a new curriculum learning strategy that did consistently outperform both the baseline of no curriculum learning and the naive curriculum learning strategy. This newly introduced strategy consists of a combination of the naive strategy and randomly selected samples, such that each bucket has at least some examples of all complexity levels while the majority of the samples, and thus the average complexity of the bucket, still gradually increases in complexity. The advantage that this combined strategy gives compared to the naive strategy, according to the authors, is that since the combined strategy provides at least some complex examples from the beginning, the network does not specialise too much in the simple task. The authors provide the following example to illustrate this: If a network is trained to add two numbers, there is certain memorisation that needs to take place in order to remember those numbers. For example, if the training data initially consists of 5-digit numbers, the network might allocate its available memory capacity to five slots, one for each number. When later presented with a 9-digit number, the memory allocation needs to change to be able to store all 9-digits. The combined strategy does not require such a memory reallocation and thus significantly outperforms the naive strategy proposed by Bengio et al. (2009).

Although curriculum learning is gathering more attention in recent academic literature, the adoption of the concept is still slow (Graves et al., 2017). Graves et al. (2017) state that a reason for this could be the sensitivity of the effectiveness of curriculum learning to the mode of progression through the task. That is, how does one determine which example is more complex than the others and in which order should one present the training examples to achieve an optimal result. Therefore Graves et al. propose a stochastic policy to determine which training example should be presented next. The authors found that their method of using a stochastic syllabus led to significant improvements in the efficiency of a curriculum learning strategy.

Recent work by Platanios et al. (2019) introduced a curriculum learning strategy for neural machine translation based on the difficulty of a training sample and the competence of the model. Thus, a training sample is only presented to the model if the model's competency at that moment is sufficient. The competence of a model is measured as a root based function of the proportion of the training data it has been trained on. The difficulty score of a sentence is based on the sentence length and the rarity of the words in the sentence. The results of this curriculum learning approach showed a decrease in training time of 70% compared to no curriculum learning, while increasing the accuracy of the model. This curriculum learning strategy can be applied to both RNN and transformer-based models.

2.3 Low-resource text summarization

As the amount of available training data is growing for many tasks within natural language processing, the performance that models achieve has been increasing rapidly. These systems all rely on the vast amount of labelled training data to achieve these levels of performance which makes them very good at a very specific task, but usually the performance on other tasks than what they have been trained for is mediocre at best. Radford et al. (2019) trained a transformer based model on a new dataset called WebText containing millions of Web pages and measured the results that it achieved without any further task specific training on several NLP problems including text summarization. Here, the model, called GPT-2, achieved a ROUGE-2 F1 score of 8.27 on the CNN/DailyMail dataset in the zero-shot[1] setting. The model is trained on a large dataset called WebText, consisting of a large amount of scraped Web pages. As many Web pages and articles include a “TL;DR:” (Too Long; Didn't Read) section after which the author writes a summary of the preceding text and the model is a language model that predicts the next word in a sequence, it is able to understand that if the input is a text ending with “TL;DR:”, then the words that will follow are a summary, thus achieving abstractive summary generation without any task-specific training.

Another result for low-resource text summarization was presented by Khandelwal et al. (2019) who achieved a ROUGE-2 F1 score of 13.1 with a pre-trained decoder-only network. The decoder-only setting, instead of decoder-encoder setting, forces the same transformer to both encode the source and generate the summary. This is made possible by appending the summary (output) to the article (input) such that text summarization can be treated as a

[1]Zero-shot refers to the situation where the model has been pre-trained on a pre-training task, usually a different task than the training task, but no further training involving text-summary pairs have taken place and thus the weights and biases are not adjusted to match the specific task. Since all the pre-training is done through unsupervised tasks this model is an unsupervised model.

language modelling task. Having a decoder-only setting allows all of the parameters, including the attention layer parameters to be pre-trained before the fine-tuning step and the number of training samples for the fine-tuning step to be reduced considerably. The authors showed this by achieving the aforementioned ROUGE-2 F1 score of 13.1 with only 3000 training samples from the CNN/DailyMail dataset whereas the decoder-encoder framework pre-trained and trained on the same data achieved a ROUGE-2 F1 score of 2.3.

Lastly, the aforementioned PEGASUS model by Zhang et al. (2019) achieved a ROUGE-2 F1 score of 13.28 in a zero-shot setting and was able to increase this score to 19.35 with merely 1000 training samples of the CNN/DailyMail dataset. To the best of our knowledge, these results make the PEGASUS model the current state-of-the-art performing model in low-resource abstractive text summarization.

Chapter 3

Data

In this Chapter we describe the datasets used in our research. As the PEGASUS model that we are applying in this research requires both pre-training and training (fine-tuning) we need two datasets, both with their own requirements. We will build our research upon an already pre-trained model introduced in Zhang et al. (2019), therefore we will not directly be using the pre-training dataset. However, as it is a substantial part of our research model we will address the dataset used for pre-training in Section 3.1. For the training of our model we require a dataset that consists of texts and parallel summary texts such that the generated summary can be compared to the actual human-written summary. This dataset is described in Section 3.2.

3.1 Pre-training corpus

In our research we utilize the pre-trained model provided by Zhang et al. (2019). This model was pre-trained on a combination of the Colossal Clean Crawled Corpus (C4), introduced in Raffel et al. (2019), and HugeNews (Zhang et al., 2019) datasets. The pre-training corpus consists of ∼1.85 billion texts stochastically sampled from the two corpora weighted by the individual dataset sizes. The pre-training corpus does not include any summaries as the pre-training task is unsupervised and therefore does not require text-summary pairs. The pre-training task is described in further detail in Section 4.2.

The first dataset, the Colossal Clean Crawled Corpus (C4), is composed of English texts from 350 million Web pages. The dataset is based on the Common Crawl Web archive which is publicly available and contains scraped HTML files from which the markup has been removed. The HTML files have been scraped from a large variety of domains and as a result contain a diverse set of Web page content. The majority of these files do not consist of natural language, therefore Raffel et al. cleaned the dataset by following a number of heuristics which resulted in the C4 dataset of cleaned natural English language. These heuristics include removing all sentences that do not end in a terminal punctuation mark or contain less than four words, and discarding pages with less than 5 sentences, containing inappropriate words, containing programming code, or containing placeholder text such as "lorem ipsum".

For their work on the PEGASUS framework, Zhang et al. (2019) created a new English text corpus which comprises of 1.5 billion news and news-like articles between 2013 and 2019. This dataset is called HugeNews and is the second dataset that is used in the pre-training corpus. The articles were collected with a Web crawler which scraped news-like articles from white-listed website domains. These domains range from high quality news websites to high-school newspapers and blogs.

3.2 Training corpus

For the model fine-tuning, a dataset is required that contains texts with a parallel summary. For this research we chose to utilize the CNN/DailyMail (CNN/DM) dataset introduced by Hermann et al. (2015). The CNN/DM dataset contains 287,000 documents, all of which are articles from either the CNN (∼93,000) or DailyMail (∼220,000) news websites. The reason these articles are very suitable for the abstractive text summarization task is that both news websites provide multi-sentence human generated summarizing sentences for each article. This provides us with a gold standard on which the model can be trained and evaluated. Table 3.1 shows summary statistics for the CNN articles, DM articles, and the combined dataset. The average text length, average summary length, and reduction percentage are similar in both datasets which shows that the datasets are well suitable to be combined.

Table 3.1: This table shows the summary statistics concerning the text-summary pair datasets. The first column shows the number of samples in the dataset, followed by the average, minimum, and maximum text and summary length. These lengths are measured by the number of words that a text contains.

	Text-summary pairs	Average length text	Average length summary	Average reduction	Range length article	Range length summary	Range reduction
Full datasets							
CNN Dataset	92,541	757	46	79%	[20,2529]	[7,108]	[59%,99%]
DM Dataset	219,506	787	55	91%	[9,2865]	[7,197]	[66%,99%]
CNN/DM Dataset	312,047	778	52	87%	[9,2865]	[7,197]	[59%,99%]
CNN/DM Dataset samples							
CNN/DM Dataset	1,000	763	46	92%	[32,2072]	[11,76]	[64%,99%]
CNN/DM Dataset	100	772	46	92%	[180,2049]	[21,67]	[66%,98%]
CNN/DM Dataset	10	923	51	91%	[214,2049]	[40,58]	[77%,97%]

Additionally, Table 3.1 shows the summary statistics of the dataset subsets which have been used in our research. These were obtained by taking random samples of size 1,000, 100, and 10, from the combined CNN/DM dataset. The data shows that the average values are similar to the full CNN/DM dataset with some metrics leaning more towards either the CNN or DM dataset for the dataset consisting of 1,000 and 100 samples. The ranges of the lengths and reduction is showing a clear shift away from the extreme values resulting in a narrower range for the dataset consisting of 1,000 samples and to an ever further extend for the datasets consisting of 100 and 10 samples. The latter dataset additionally shows a deviation from the average length summary which is significantly longer than the average length in the full dataset. The average length of the summary, however, is closer to the average of the complete dataset compared to the dataset samples of size 100 and 1,000.

Chapter 4

Methodology

In this research a sequence-to-sequence framework with a transformer-based encoder-decoder architecture is used. The model is pre-trained and then fine-tuned with a variation of curriculum learning strategies and data augmentation techniques. The following Sections delve into the various components of this framework. Section 4.1.1 describes the sequence-to-sequence model. Next, Section 4.2 describes the strategies applied to pre-train the model. Thirdly, Section 4.3 describes the various curriculum learning strategies and data augmentation steps applied during the model's training phase. Lastly, Section 4.4 describes the evaluation methods used to determine the performance of the models.

4.1 The Model

The model we use in this research is the PEGASUS model, as introduced in Zhang et al. (2019). PEGASUS is a sequence-to-sequence model with a standard transformer encoder-decoder architecture. Before we describe further details of the PEGASUS model in Section 4.1.3, we first delve further into sequence-to-sequence models in Section 4.1.1 and transformer architectures in Section 4.1.2.

4.1.1 Sequence-to-sequence model

Sequence-to-sequence models based on a neural network were introduced by Sutskever et al. (2014) for machine translation and are generally applied to problems where both the input and output are sequences, although not necessarily of equal length. This is the case for, e.g., text translation, speech recognition, and text summarization. In the case of text summarization, the summary (output) clearly must have a different length that the original text (input). It is called a sequence-to-sequence model as it maps the original text, which is a sequence of words, to a summary, which is another sequence of words.

Typically, a sequence-to-sequence model consists of two sections: an encoder and a decoder. The encoder turns the model input into a vector representation, which is then used as input for the decoder. The decoder turns this vector representation into an output sequence. In the context of text summarization this would be equivalent to the encoder taking the original text as input and transforming it into a vector representation. The decoder takes this vector representation as input and decodes it to an output summary. In the following section we will delve further into the details of the encoding and decoding processes.

4.1.2 Transformers

Before Vaswani et al. (2017) introduced transformers, most sequence-to-sequence models in NLP were based on recurrent neural networks (RNN) (Rumelhart et al., 1986; Werbos, 1990) which was considered the state-of-the-art approach in sequence-to-sequence modeling (Vaswani et al., 2017). These RNN's used Long Short-Term Memory (LTSM) (Hochreiter and Schmidhuber, 1997) or Gated Recurrent Unit (GRU) mechanisms (Cho et al., 2014) which were developed in order to deal with the vanishing gradient problem (Hochreiter et al., 2001) that occurs within RNN when dealing with long sequences. Although these models dealt with the vanishing gradient successfully to a certain extent, there were still some drawbacks (Vaswani et al., 2017). The most fundamental and impactful drawback being the constraint of sequential sequence processing. That is, a sequence-to-sequence model based on RNNs required the input sequence to be processed (encoded and decoded) one element at a time because the processing of the current element depends on the preceding elements in the sequence. Thus, the preceding elements had to be processed first. Although there have been successful efforts to increase the computational efficiency of these networks, due to the nature of modern computing systems a process executed sequentially will not achieve the efficiency levels of a parallel process (Xhafa et al., 2015).

To deal with the previously identified shortcoming of RNNs, Vaswani et al. (2017) introduced transformers which can process the input sequences in parallel, thereby making them significantly faster. Transformers are able to process sequences in parallel while still managing to take into account the relevant information from the entire sequence through the use of a combination of feedforward neural networks (FNN) and an attention mechanism (Bahdanau et al., 2015), specifically self-attention (Cheng et al., 2016). Self-attention is a concept which allows the transformer to consider all other words in the sequence and their relation to the word it is currently processing. By implementing a self-attention layer, the transformer "knows" that for the 8^{th} word in the sequence it is important to take into account the 2^{nd} and 5^{th} word in the sequence without having to process any of those words first.

Figure 4.1 shows the model architecture of a transformer. Transformers consist of an encoder (left) and a decoder (right) stack, both of which are composed of a stack of N identical layers as is indicated by the "Nx" in the figure. Each encoder layer consists of a multi-head attention sub-layer and a feed-forward sub-layer. The decoder layer also consists

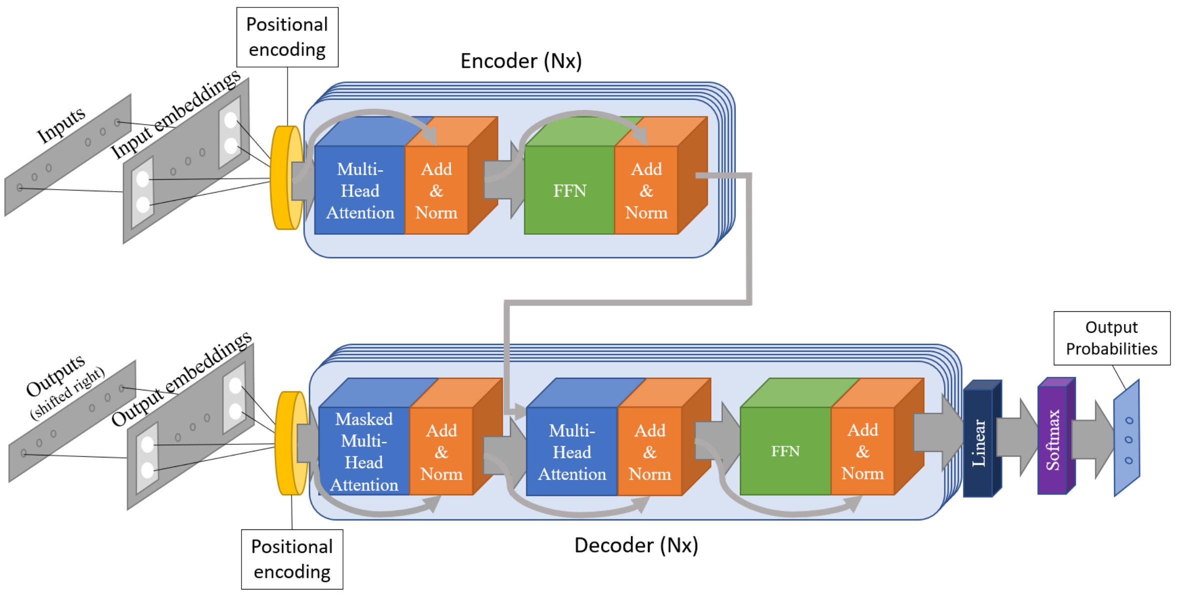

Figure 4.1: The transformer model architecture.

of these two sub-layers and an additional multi-head attention sub-layer to take into account the output of the decoder. The input of the encoder and decoder is first embedded and positional encoding is added to the embedding.

The embedding of the input and output sequences converts the text, a string, to a low-dimensional vector representation which preserves the contextual similarity of words (Press and Wolf, 2017) such that the algorithm can process the corpus of text. Subsequently, the embedded vectors are supplemented with a positional encoding (Gehring et al., 2017) which contains information about the position of each element in the sequence. In this research we will follow Vaswani et al. (2017) and Zhang et al. (2019), and use sinusoidal positional encoding. Vaswani et al. (2017) hypothesised that this type of positional encoding would allow the model capture the relative positions between words well and allow the model to extrapolate for sequences that are longer than those that encountered during the training process.

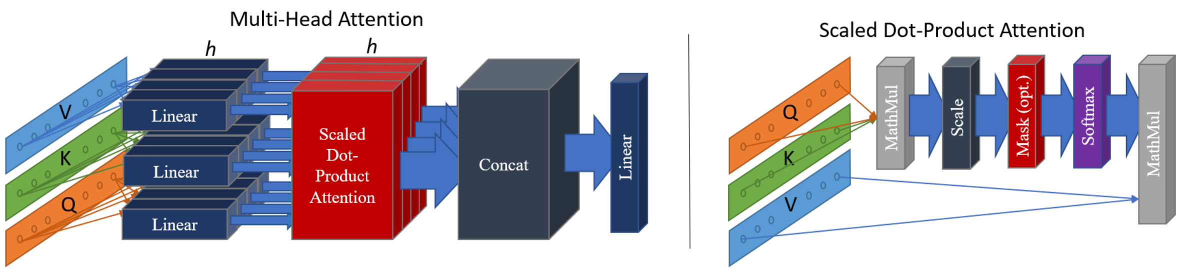

Figure 4.2: The attention mechanism.

The embedded words with positional encoding are then processed by the self-attention layer. Figure 4.2 shows a representation of the scaled dot-product attention mechanism and the multi-head attention mechanism, which are several parallelized scaled dot-product attention mechanisms such that multiple input vectors can be processed simultaneously and thus increase computational efficiency. The scaled dot-product attention mechanism is almost identical to the dot-product attention mechanism and can be captured with the following equation;

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{4.1}$$

where Q is a matrix of queries, K is a matrix of keys, V is a matrix of values, and d_k is the dimension of the model. The difference between the scaled dot-product attention mechanism and the dot-product attention mechanism is the scaling factor $\frac{1}{\sqrt{d_k}}$, which improves the performance of the dot-product attention model for a large dimension (large values of d_k) according to Vaswani et al. (2017). A possible explanation is that the gradients of softmax function become very small as the dimension and thus the dot-products grow exceedingly large (Vaswani et al., 2017). The scaling product counteracts this growth of the dot-products and thus the gradients do not become as small which in turn improves the performance of the attention mechanism.

The Multi-Head Attention, displayed on the right in Figure 4.2, shows h layers (also referred to as "heads") of the scaled dot-product attention. That means that the process that occurs within the scaled dot-product attention as described above is performed h times in parallel where each one is initialised randomly. This creates multiple representation subspaces which improves the models ability to incorporate the information about the position of the word in a sentence, because each attention head can focus on a separate relationship within the sentence. To illustrate this point further, consider the sentence "The chicken crossed the road because it wanted to get to the other side.", while processing the word "it" in this sentence there could be one attention head which focuses on the relation between "it" and "the chicken", while another head captures the relation between "it" and "wanted". As this creates h separate output values, whereas the next layer of the model expects only one input, the h output matrices are concatenated and multiplied with weight matrix W^O which is trained jointly with the model. This results in a single matrix containing information from all h attention heads. Equation 4.2 describes this process.

$$\begin{gathered} MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O \\ \text{where } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \end{gathered} \tag{4.2}$$

The output of the multi-head attention blocks is passed through a fully connected feed-forward network (Sainath et al., 2015). This output contains information about the position of each word in the sentence and the relation with respect to the other words in the sentence. Since this information is now all captured in a single matrix, the recurrence of an RNN is no longer required to capture this information. Using a self-attention layer has several advantages, first of all they can be parallelized which makes them significantly faster than RNNs. Furthermore, the distance between the words in a sequence becomes much smaller as the distance is logarithmic in an FNN and linear in an RNN, which makes it easier to learn long-term dependencies in an FNN (Vaswani et al., 2017). The feed-forward network consists of two linear transformations and a ReLU function (Nair and Hinton, 2010) as is shown in equation 4.3.

$$FNN(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{4.3}$$

The last steps in the transformer, shown on the right top in Figure 4.1, consists of a learned linear transformation and softmax function to convert the output of the decoder feed-forward network to the probabilities which can be used to predict the next word in the sequence.

4.1.3 PEGASUS

PEGASUS is a transformer based sequence-to-sequence model. There are several parameters that can be set in this type of model which we will describe in this section. Two versions of the PEGASUS model have been introduced in Zhang et al. (2019), however, we will use only the larger model in this research.

As indicated by “Nx” in Figure 4.1, the amount of decoder and encoder layers need to be specified, PEGASUS consists 12 of these layers. This is only half of the amount of layers in the BERT$_{LARGE}$ model (Devlin et al., 2019), which is another transformer model widely applied in Natural Language Processing. Furthermore, the amount of attention heads in the Multi-Head attention layer is set to 16, BERT$_{LARGE}$ has 12. The dimensionality of the hidden layers is set to 1024, that is, the size of the output of each encoder and decoder layer is 1024, this is the same as in BERT$_{LARGE}$. Last, the size of the feed-forward layer is set to 4096, which is also the same as in BERT$_{LARGE}$. The total amount of parameters to be fine-tuned is 568M.

For optimisation, both pre-training and fine-tuning used Adafactor (Shazeer and Stern, 2018) with square root learning rate decay and a dropout rate of 0.1.

4.2 Pre-training

The essence of training a neural network lies in the ability to find the optimal weights and biases for each node in the network. This is traditionally done through training on a large training dataset with, e.g., back-propagation (Rumelhart et al., 1986), where at the beginning of the training, the weights and biases are initialized randomly. Pre-training gives the network a head-start by training it on a task that is not the same as the main task, but it is similar enough such that the weights and biases for this task can be used as good starting values for the main task. Therefore, the subsequent training-phase requires less training data in order to find the optimal weights and biases for the main task as it starts off closer to the optimal values. Determining a good pre-training task is more of an art than a science and is where the work of Zhang et al. (2019) provided a significant contribution by developing a novel pre-training method, called gap sentences generation.

The gap sentences generation method consists of removing one or more sentences from a text and then using the remaining text to fill in those gaps in the text. That is, the remainder of the text is the input data for the model which then generates the missing sentences as output. The generated sentence can then be compared with the original sentence and thereby the model can be trained. Zhang et al. (2019) showed that this task is similar enough to the main task, abstractive text summarization, that this pre-training method leads to high quality generated summary outputs. The question that remains is which sentence should be deleted from the original text to achieve optimal results. Zhang et al. (2019) introduced and compared several approaches. These are the following; (1) masking a random selection of m sentences, (2) masking the first m sentences, and (3) masking the m most important sentences, where the importance is determined through the ROUGE-1 F1 (Lin, 2004) score between each sentence and the remainder of the document. The latter method, called the Principal approach, can be further subdivided into four variations. First of all, the sentences can be scored independently or sequentially (Nallapati et al., 2017). Additionally, when calculating the ROUGE-1 F1 score, the n-grams can be considered as a set in order to prevent identical n-grams to be counted multiple times or simply as a bag where duplicates are allowed. Thus leading to the four options: (1) independent scoring + set of n-grams, (2) independent scoring + bag of n-grams, (3) sequential scoring + set of n-grams, and (4) sequential scoring + bag of n-grams. The methods were compared by pre-training the pre-training dataset with each approach and fine-tuning the model on four different datasets. All training and pre-training was done with a batch size of 256 and a learning rate of $5e-4$, pre-training was done for 500,000 steps and fine-tuning for 50,000 steps. The performance was measured by a non-weighted average of the ROUGE-1, ROUGE-2, and ROUGE-L F1 scores. The results showed that the principal approach with independently scored sentences and a bag of n-grams outperformed or performed very similar to the other approaches across all 4 datasets. Therefore, our research will build further upon the model pre-trained with the principal gap-sentences approach with independently scores sentences while allowing multiple identical n-grams.

4.3 Model training

The following section outlines the model training approaches that are applied after the pre-training methods and are aimed at optimising the model's performance with the smallest amount of required training samples. Section 4.3.1 describes the curriculum learning concept and how it is applied in our research where we introduce a novel text-summary pair difficulty scoring system. Last, Section 4.3.2 describes the data augmentation techniques that are applied in our research.

4.3.1 Curriculum learning

The concept of curriculum learning is that algorithms, just like humans, learn better when initially presented with easy examples followed by other examples which gradually increase in complexity. This idea was introduced by Elman (1993) and the concept has been successfully applied in various machine learning and NLP applications (Pentina et al., 2015; Spitkovsky et al., 2010). Although the general concept behind each curriculum learning approach is the same, many variations exist and have been applied (Cirik et al., 2016; Bengio et al., 2009; Spitkovsky et al., 2010).

To the best of our knowledge, curriculum learning strategies have not been applied to the abstractive text summarization problem. Therefore, it is not be possible to directly apply previously developed methods. However, there are curriculum learning strategies which have been applied to other natural language processing tasks which we can base our strategy on.

Algorithm 1: One-Pass Curriculum

```
Result: Trained model based on one-pass curriculum
M = Model
D = Dataset
$d_i$ = sample i from the dataset
Function C($d_i$):
    return complexity score i

D' = sort(D,C)
$\{D^1, D^2, ..., D^k\} = D'$ where $C(d_a) < C(d_b)$, $d_a \in D^i$, $d_b \in D^j, \forall i < j$ $D^{train} = \emptyset$
for $s = 1, ..., k$ do
    $D^{train} = D^s$
    while model accuracy improved during the last p epochs do
        $TrainForOneEpoch(M, D^{train})$
    end
end
```

Firstly, a curriculum learning strategy that we will apply is the One-Pass curriculum learning strategy as proposed by Bengio et al. (2009). In this strategy the training data D is sorted from easy samples to complex samples by a curriculum C which is then divided into k buckets. The strategy, as proposed by Bengio et al. (2009) would train the model on the first bucket, containing the simplest training samples, then continue to the next bucket once it trained on the first bucket for a fixed number of epochs. In this paper we will follow the modified version with early stopping (Cirik et al., 2016) which means that the training from a bucket will stop once the accuracy of the model has not improved during the last p epochs, therefore, allowing it to move to the next bucket of training samples faster in order to avoid over-fitting on any particular sample bucket. The model accuracy is computed on a sample held-out from each bucket consisting of 10% of the samples in the bucket. The training of the model is finalised once all buckets have been used to train the model. The one-pass curriculum learning strategy pseudo-code is shown in Algorithm 1.

Spitkovsky et al. (2010) proposed another curriculum learning strategy, Baby-Steps curriculum, which is identical to the one-pass curriculum until the moment where the accuracy of the model does not improve for p subsequent epochs. Whereas the samples in the current bucket are discarded in the one-pass curriculum strategy, in the baby-steps curriculum the current bucket's training samples are merged with the next training bucket, thereby, increasing the average complexity of the training samples through expansion of the total sample pool instead of replacement of the sample pool as in the one-pass curriculum strategy. Algorithm 2 shows the pseudo-code of the baby steps curriculum strategy.

Algorithm 2: Baby-Steps Curriculum

```
Result: Trained model based on baby steps curriculum
M = Model
D = Dataset
$d_i$ = sample i from the dataset
Function C($d_i$):
    return complexity score i

D' = sort(D,C)
$\{D^1, D^2, ..., D^k\} = D'$ where $C(d_a) < C(d_b)$, $d_a \in D^i$, $d_b \in D^j, \forall i < j$
$D^{train} = \emptyset$
for $s = 1, ..., k$ do
    $D^{train} = D^{train} \cup D^s$
    while model accuracy improved during the last $p$ epochs do
        $TrainForOneEpoch(M, D^{train})$
    end
end
```

All of the methods above have in common that the training samples need to be sorted by difficulty before the curriculum learning strategy can be implemented. To the best of our knowledge, a difficulty measure for abstractive text summarization training data is not described in existing literature. Therefore, we propose a novel difficulty measure for abstractive text summarization based on the rewriting operations required to get from the original text to the summary. The available operations are (1) word deletion, (2) word addition, (3) word substitution, and (4) word reordering (Cohn and Lapata, 2008). In order to sort the sentences it would be ideal to capture the difficulty of a text-summary pair in a single number. We propose to use a weighted sum of the text operations. Naturally, some operations will be much more common than others and some operations are harder for a model to learn. A weighted average will therefore give a fairer representation of the difficulty of a text-summary pair than an unweighted average.

We hypothesise that the magnitude of the weights, from lowest weight to highest weight, of the text operations are expected to be in the following order:

1. Word deletion;
2. Word reordering;
3. Word substitution;
4. Word addition.

Each of these text operations could be further subdivided into easier and more complex versions of their respective operation. E.g., we expect that adding the word "the" to a sentence will be much easier for the model to learn than adding the word "snowstorm", simply because the former word is much more common than the latter. However, as there is certainly value in the simplicity of a ranking system we choose to not delve further into these possible sublevels of text operations and consider all versions of a text operation as equivalent. Let us formalise the method described above by defining the complexity (C) of a training sample (s) as the weighted sum of the text operations word deletion (wd), word reordering (wr), word substitution (ws), and word addition (wa) as described in Equations 4.4 - 4.6.

$$C(s) = w_{wd} * \sum_i wd_i + w_{wr} * \sum_j wr_j + w_{ws} * \sum_k ws_k + w_{wa} * \sum_l wa_l \tag{4.4}$$

$$w_{wd}, w_{wr}, w_{ws}, w_{wa} \in [0, 1] \tag{4.5}$$

$$w_{wd} + w_{wr} + w_{ws} + w_{wa} = 1 \tag{4.6}$$

The weights of these text operations will be optimized through a random search approach. We will iterate though 10 randomly generated possible weight combinations within the limitations given by Equations 4.5 and 4.6. Using these weight combinations, 10 sets of 1.000 training samples each are generated and divided into 5 buckets (200 samples per bucket) after sorting them according to the complexity scores using their respective hyperparameters. Each bucket is trained up to 5 epochs using the one-pass curriculum learning strategy without early stopping and a checkpoint is created after each epoch. Then the performance of each model checkpoint will be evaluated on the valuation set and the best checkpoint will be used as the initialization point of the next bucket. The performance is measured based on a combined ROUGE F1 score as defined in Section 4.4. In order to incorporate our hypothesis stated in Section 4.3.1, we add an 11th weight combination into the comparison which is in line with our hypothesis and has the following weights;

- $w_{wd} = 0.1$
- $w_{wr} = 0.2$
- $w_{ws} = 0.3$
- $w_{wa} = 0.4$.

Finally, two baseline measures will be included in our comparison. Firstly, the length of the input text, measured in number of words, will be used as a proxy for complexity. Second of all, the reduction percentage between the input text and human-written summary will be considered as a complexity proxy. E.g., if a text has 1000 words and the corresponding summary consists of 50 words, then the reduction percentage is 95%. The input data will be sorted from low to high based on these values and then the curriculum learning strategies will be applied. The performance of these methods will then be compared to our novel complexity scoring method in order to assess its effectiveness to improve performance.

4.3.2 Data Augmentation

Another technique to improve model performance with small training datasets is Data Augmentation (Shorten and Khoshgoftaar, 2019). Data augmentation is a technique to artificially expand the pool of training samples by altering the existing training samples in some way without diverting from the original meaning too much. Common applications in image based training samples include rotating, cropping and mirroring the image. In our research we will apply the data augmentation techniques for NLP proposed by Wei and Zou (2019). Their work presents four easy-to-implement data augmentation techniques that create significant performance improvements for five classification tasks, especially with small datasets. Therefore we expect that applying similar techniques to our text and summary data could result in performance improvements in the abstractive summarization task. The four techniques proposed by Wei and Zou (2019) are (1) Synonym Replacement (SR), (2) Random Insertion (RI), (3) Random Swap (RS), and (4) Random Deletion (RD). Synonym replacement The details of the four techniques are outlined in Table 4.1.

Operation	Description	Example Sentence
None		His very rough summary does not do justice to the original text and its intellectual sophistication.
SR	Choose n words from the sentence at random (excluding stop words). Replace those words with a randomly selected synonym.	His very **unpolished** summary does not do justice to the original text and its intellectual sophistication.
RI	Insert a synonym of a random word in the sentence (excluding stop words) at a random position in the sentence. Perform this n times.	His very rough summary does not do **elegance** justice to the original text and its intellectual **sophistication**.
RS	Swap the position of two random words in the sentence. Perform this n times.	His **do** rough summary does not **very** justice to **its** original text and **the** intellectual sophistication.
RD	Remove each word in the sentence with probability p.	His rough summary does not do to the original text and its sophistication.

Table 4.1: This table shows the four operations proposed by Wei and Zou (2019) and applied in our research. The operations described are Synonym Replacement (SR), Random Insertion (RI), Random Swap (RS), and Random Deletion (RD). For further clarification an example sentence is included for each operation.

4.4 Evaluation methods

A metric that is widely applied as an evaluation method for text summarization models is the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) package introduced by (Lin, 2004). The ROUGE package consists of a precision metric, a recall metric, and an F1 score which combines the precision and recall scores by calculating their harmonic mean. Several variations of the ROUGE score exist, the key difference between each variation is the manner in which the overlap between the generated output and the gold standard output is measured. In this paper we will be applying the ROUGE-N and ROUGE-L score, as these are commonly used in existing literature to evaluate abstractive summaries. The ROUGE-N and ROUGE-L scores are further described in Sections 4.4.1 and 4.4.2, respectively.

4.4.1 ROUGE-N

The ROUGE-N method considers the overlap of N-grams (a sequence of n words) between the generated summary and the actual summary. The N-gram recall is the amount of overlapping N-grams divided by the total amount of N-grams in the reference summary, i.e., the human written summary. The precision measure is calculated by dividing the number of overlapping N-grams by the total amount of N-grams in the model generated summary. Thus, the recall measure measures how many of the possible N-grams the model is able to generate and the precision measure measures how many of the generated N-grams are in the reference summary. The F1 measure combines these measure into a single value through a harmonic mean. The three measures are defined in Equations 4.7, 4.8 and 4.9.

$$\mathrm{F1}_{N-gram} = \frac{2}{\frac{1}{recall_{N-gram}} + \frac{1}{precision_{N-gram}}}, \tag{4.7}$$

where

$$recall_{N-gram} = \frac{\text{number of overlapping N-grams}}{\text{total N-grams in the reference summary}}, \tag{4.8}$$

and

$$precision_{N-gram} = \frac{\text{number of overlapping N-grams}}{\text{total N-grams in the model summary}}. \tag{4.9}$$

4.4.2 ROUGE-L

The ROUGE-L compares the generated and actual summary, similarly to ROUGE-N, but does this based on the longest common sub-sequence (LCS) (Cormen et al., 1989). The LCS is the longest sequence of words that is present in both the reference summary and the model generated summary. The recall is the LCS divided by the total number of words in the reference summary (m) and precision is the LCS divided by the total number of words in the model generated summary (n). The F1 score is the harmonic mean between these two measures. The three ROUGE-L measures are defined in Equations 4.10, 4.11 and 4.12.

$$\text{F1}_{LCS} = \frac{2}{\frac{1}{recall_{LCS}} + \frac{1}{precision_{LCS}}}, \tag{4.10}$$

where

$$recall_{LCS} = \frac{LCS(sentence_{reference}, sentence_{model})}{m}, \tag{4.11}$$

and

$$precision_{LCS} = \frac{LCS(sentence_{reference}, sentence_{model})}{n}. \tag{4.12}$$

4.4.3 Combined ROUGE

In this research we follow the measures used by Nallapati et al. (2016) and Zhang et al. (2019), which are the ROUGE-1 (ROUGE-N with $N = 1$), ROUGE-2 (ROUGE-N with $N = 2$) and ROUGE-L F1-scores. Although we believe that using all three of these measured gives a more complete evaluation of a summary than using merely one of these scores, it presents us with a challenge when comparing performance of various models. Namely, three separate scores can provide inconsistent conclusions when comparing 2 or more models, since different measures may result in multiple best performing models. Therefore we combine these measures into a combined ROUGE score (see Equation 4.13) through a weighted average of these three scores, as was introduced in the code of the implementation of the work of Zhang et al. (2019) available at `https://github.com/google-research/pegasus`. The weights are 1 for the ROUGE-N, and ROUGE-L F1 scores and 2 for the ROUGE-2 F1 score, which is in line with the work of Zhang et al. (2019). The weights are chosen in favour of the ROUGE-2 F1 score as we believe this score strikes the best balance when evaluating summary quality between determining whether a model generates the correct words and whether it places them in the correct order. The ROUGE-2 F1 score does not assign any value to a model that chooses all the correct words but places them in a completely wrong order, however, we would certainly prefer such a summary to a summary that has none of the correct words. Therefore, the ROUGE-1 F1 score is still an important measure to include in the combined ROUGE score. Furthermore, the ROUGE-L F1 score is beneficial to include in the combined ROUGE F1 score as it captures the value of the longer correct sequences that a model can generate. For example, a model with merely a ROUGE-1 F1 and ROUGE-2 F1 score would not be able to differentiate between a model that can generate 7 words in the correct (1-2-3-4-5-6-7-9-8-10) order compared to a model that can correctly order at most a 4 word sequence correctly (1-2-3-4-6-5-7-8-9-10).

$$\text{Combined ROUGE F1 score} = R_1 + 2 * R_2 + R_L \tag{4.13}$$

As all the ROUGE scores are proportions, the range of the ROUGE scores is $[0, 1]$. For improved readability we scale all the reported ROUGE scores in this paper by 100 (e.g., a ROUGE score of 0.20567 will be 20.567).

Chapter 5

Results

The following section outlines the results of our research. First of all, Section 5.1 outlines the results of the optimal parameters for our novel summary-text complexity ranking system. Then, Section 5.2 describes the achieved combined ROUGE F1 scores through the various strategies and techniques applied to the datasets.

5.1 Complexity scoring

Our novel complexity scoring system required us to set a number of hyperparameters, namely the weight of each text operation. In order to find the optimal values for these weights we applied a random search approach as previously described. The resulting scores are shown in Table 5.1.

Table 5.1 shows that the best overall results were achieved by combination number 4 (in bold) which was 1.7% above the overall average. The weights that resulted in this highest score where not particularly close to those anticipated, especially the weight for word additions. This indicates that the word addition has a smaller impact on the training sample complexity than we hypothesised. This is likely explained by the fact that the word additions task was far less common that the word deletion and word reordering task, as shown in Table 5.2. Thus, although the task is likely still more complex for the model to learn, it is not so important to learn it because even if the model performs poorly in that task the generated summary can still be of high quality. With the same logic one might expect that the word deletion task should receive the highest weight, as it is by far the most common task that the model is required to perform. However, when comparing the results in Table 5.1 it is clear that this is not the case as there are several combinations with high weights for the word deletion task (2, 7, and 8), which all show average or below average results. Furthermore, when considering the low average occurrence and low standard deviation of the word substitution task one can conclude that the effect of the word substitution weight is very small. This leads us to believe that the quality of the complexity scoring system is determined by the relative distribution of the weights between the remaining three tasks. Our results indicate the the optimal solution is a weight combination where similar weights are attached to the word deletion and word addition task and a weight approximately four times as high to the word replacement task. One might argue that the word addition task also has a low occurrence in this dataset, and thus the only relevant weights are those for word deletion and word reordering. However, when considering combination 5, where the weight of the word reordering task is also four times as large as that of the word deletion task, we see that the performance is merely average and significantly worse than that of combination four. If the word addition weight would have a very minimal effect on the score, just like the word substitution weight, then combination 4 and 5 should be equivalent as the ratio of the only two relevant weights under this assumption is almost the same (1:4). However, since this is not the case, these results indicate that the word addition task weight does make a significant impact.

The astute reader might note that, when rounding off the weights of each text operation to two decimals, we are left with 1,061,106 possible weight permutations. This is a well-known discrete mathematics problem commonly solved by a "stars-and-bars" approach (see equation 5.1). In order to determine how many ways there are in which one can assign a value between 0 and 1 to four weights such that they sum to 1 let us first consider a single possible solution as shown in Equation 5.1. In this solution we assign a weight of 0.97 to w_{wd}, 0.01 to w_{wr}, 0.01 to w_{ws}, and 0.01 to w_{wa}. We can represent this solution in "stars" and "bars", where we split up the numbers into their smallest components, which we chose to be 0.01, and represent them by stars and use the bars to show where the split is made between the weights. That is, every star to the left of the first bar represents a value of 0.01 assigned to the first weight, every star in between the first and second bar represents a value of 0.01 assigned to the second weight, etc.. Using this representation it becomes clear that the number of possible ways that we can assign the weights is the same as the number of ways we can place 3 bars among those 100 stars. This is equal to the number of possible permutations of length 3 (3 bars) out of a set of 103 elements (100 stars + 3 bars), namely $1,061,106$ (see equation 5.2).

Figure 5.1 shows that the Baby-steps and One-Pass curriculum learning strategies follow a similar pattern for the majority of the combinations, indicating that the effect of the hyperparameter choice is comparable for both strategies. Although we do not expect both curriculum learning strategies to have the same optimal hyperparameters, we will continue our research with the weights of combination four for both curriculum learning strategies as it gave the highest overall combined ROUGE F1 score.

Although the random search approach has shown to be efficient at finding a solution that is reasonably close to the optimal solution (Bergstra and Bengio, 2012), we would agree that a more thorough examination of the optimal weight combination could benefit the performance of the complexity scoring strategy as we could potentially find a better weight combination. However, we are unfortunately limited in our time and resources and chose to concentrate

Table 5.1: The results of the curriculum learning complexity scoring strategy hyperparameter optimisation. Through a random search approach weight combination 1-10 have been generated and weight combination 11 has been added based on our hypothesis. The weights indicate the relative importance given to a text operation. These operations are word deletion (w_{wd}), word reordering (w_{wr}), word substitution (w_{ws}), and word addition (w_{wa}). After sorting a dataset of 1,000 samples using these hyperparameters and separating the samples into 5 buckets based on their complexity score, the PEGASUS model fine-tuned using the One-pass and Baby-steps curriculum learning strategy. This resulted in a Rouge-1, Rouge-2, and Rouge-L F1 score for all weight combinations. All ROUGE scores have been divided by their mean value across the 11 weight combinations, as this gives easier insight into relative performance. The scores are aggregated into one score such that comparisons can be made by adding up the Rouge-1 score, 2 times the Rouge-2 score and the Rouge-L F1 score. The last row shows the overall score, which consists of the average of the combined scores of the One-pass curriculum learning strategy and Baby-steps curriculum learning strategy.

Combination	1	2	3	**4**	5	6	7	8	9	10	11
w_{wd}	0.02	0.52	0.25	**0.11**	0.02	0.19	0.64	0.33	0.08	0.30	0.10
w_{wr}	0.01	0.06	0.23	**0.41**	0.08	0.53	0.21	0.19	0.23	0.45	0.20
w_{ws}	0.43	0.20	0.41	**0.37**	0.81	0.02	0.01	0.14	0.57	0.20	0.30
w_{wa}	0.54	0.21	0.11	**0.11**	0.09	0.26	0.13	0.34	0.12	0.06	0.40
One-Pass Curriculum											
Rouge-1	1,003	1,007	1,005	**0,994**	1,008	0,997	1,001	0,998	0,993	1,003	0,992
Rouge-2	1,009	1,014	1,018	**0,987**	1,004	0,999	1,004	0,999	0,983	1,004	0,980
Rouge-L	0,996	0,995	0,996	**1,124**	0,984	0,985	0,991	0,987	0,976	0,990	0,975
Score	1,003	1,006	1,007	**1,028**	1,000	0,994	0,999	0,995	0,985	1,000	0,983
Baby-Steps Curriculum											
Rouge-1	0,994	0,999	0,996	**1,006**	1,006	1,006	0,986	1,017	1,000	1,001	1,003
Rouge-2	1,000	0,989	0,995	**1,004**	1,003	1,004	0,985	1,008	0,998	1,017	1,008
Rouge-L	0,995	0,996	1,000	**1,007**	0,999	1,007	0,991	1,002	1,002	1,005	1,006
Score	0,996	0,995	0,997	**1,005**	1,003	1,006	0,987	1,010	1,000	1,007	1,005
Overall score	1,000	1,000	1,002	**1,017**	1,001	1,000	0,993	1,002	0,992	1,003	0,994

our efforts for this paper on other sections of our research. Nonetheless, we would highly encourage further research into the optimal weight combination by either applying a different strategy or by increasing the number of random combinations considered.

$$\begin{aligned} w_{wd} + w_{wr} + w_{ws} + w_{wa} &= 1 \\ 0.97 + 0.01 + 0.01 + 0.01 &= 1 \\ \underbrace{\star\star\star ... \star\star\star}_{97} \mid \star \mid \star \mid \star &= 1, \text{ with } \star = 0.01 \end{aligned} \tag{5.1}$$

$$\text{Number of weight sets} = P(103, 3) = \frac{103!}{(103-3)!} = 103 * 102 * 101 = 1,061,106 \tag{5.2}$$

Table 5.2: The average occurrence, with the respective standard deviation in brackets, of each task in the 1000 samples used for the complexity scoring strategy hyperparameter optimization.

	word deletion	word reordering	word substitution	word additions
Average occurence	287.2 (156.54)	18.31 (5.82)	0.4 (0.62)	4.4 (2.99)

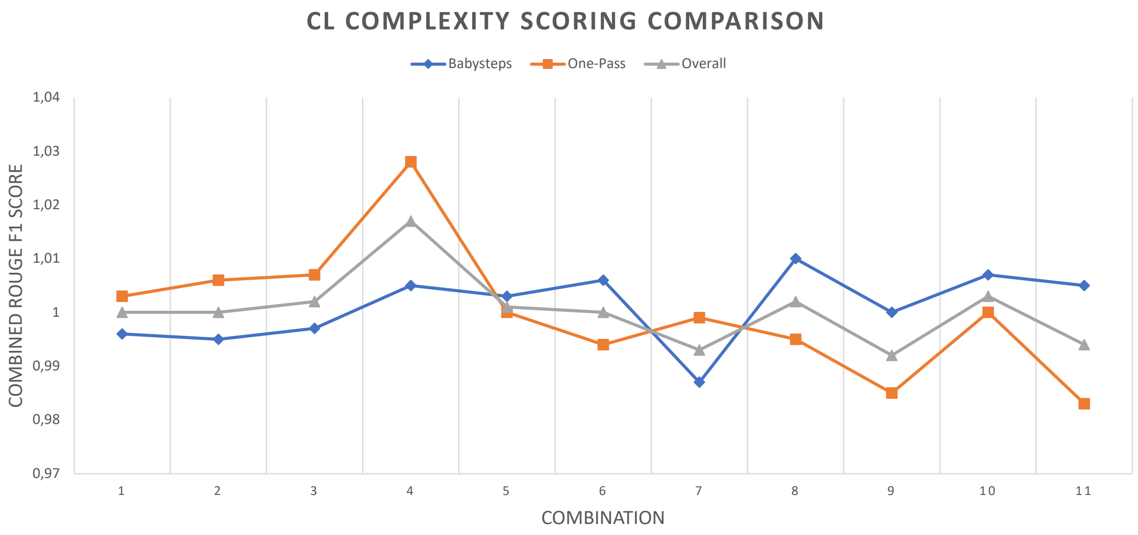

Figure 5.1: This figure shows the relative combined ROUGE F1 scores for the complexity scoring hyperparameter comparison for all the weight combinations as described in Table 5.1.

$$D(s) = w_{wd} * \sum wd + w_{wr} * \sum wr + w_{ws} * \sum ws + w_{wa} * \sum wa \tag{5.3}$$

$$w_{wd}, w_{wr}, w_{ws}, w_{wa} \in [0, 1] \tag{5.4}$$

$$w_{wd} + w_{wr} + w_{ws} + w_{wa} = 1 \tag{5.5}$$

5.2 Summarization results

Table 5.3 shows the achieved ROUGE F1 scores for the various training strategies described in the methodology section of this paper. The first row shows the results with no adaptation to the original PEGASUS framework and is the baseline against which we compare our other strategies.

In an ideal situation we would be able to compare our results with the state-of-the-art results achieved by the PEGASUS model as introduced in Zhang et al. (2019). In order to fairly compare our results with the results reported there, we would have to use comparable parameters and hyperparameters in our research. To get their low-resource results, Zhang et al. (2019) used 2,000 training steps and a batch size of 256. This means that 512,000 $(2,000 * 256)$ training samples have been available to the model for training, e.g., a dataset of 1,000 examples was trained up to 512 $(512,000/1,000)$ epochs. A batch size is the amount of samples that are being used per iteration on the model parameters. That is, a batch size of 1 means that after each sample the model updates all the model parameters based on the result of that single sample. A batch size of 256 means that the parameter adjustments from 256 samples are combined and then used to update all model parameters only once. The batch size is limited by the available hardware resources such as Graphics Processing Units (GPUs) or Central Processing Units (CPUs) as the intermediate results of all the samples are stored on their memory until an entire batch can be processed. The memory stores all the model parameters, intermediate calculations, model specific optimization variables and then it additionally needs free working space to process the next sample.

With the hardware resources available to us we are able to train the model with a maximum batch size of 2[1], which means that to train on a comparable number of training steps, we would have to train the model for 256,000 $(512,000/2)$ steps. This would take us an estimated 114 hours and 284.40 EUR of cloud computing costs to train each model which is infeasible due to the limited funding and time constraints for this research. Therefore we have opted run the model made available by Zhang et al. (2019) with a much lower number of steps and compare our results with those results, such that comparison between the results is meaningful and relative differences can be shown.

The obvious downside to this approach is that there is no guarantee that any differences in results achieved on this smaller scale will translate into equivalent results in a setting with larger batch-sizes and higher number of training steps. However, as there is no indication that this would not be the case we believe that this small scale comparison is meaningful and a worthwhile contribution to the existing research literature. Nonetheless, we would highly encourage other researchers with greater resource availability to perform further research on this matter.

The curriculum learning strategies have been executed as described in Section 4.3.1 with $p = 3$. Thus, if the validation performance of a model has not improved during the last 3 epochs, the algorithm moved forward to the next training bucket. For the dataset with 10 samples we chose to use 2 buckets, for the dataset with 100 samples we use 5 buckets and for the dataset with 1,000 samples we use 10 buckets. All the results are achieved by testing on the out-of-sample test set as commonly used when using the CNN/DM dataset (Wang et al., 2019) consisting of 11,490 samples.

Let us first consider the results of the One-Pass curriculum learning strategy. In the dataset consisting of 10 samples, the various sorting methods did not result in any major differences with a 0.1% decrease in performance for a curriculum learning strategy without

[1]A larger batch size caused the model to run out of memory and throw an error.

sorting and a 0.3% increase for all other strategies compared to the baseline performance of no curriculum learning. This is not surprising, as with such a small dataset a variation in the sample training order is likely to have limited effects. Furthermore, in this case, the length, reduction, and complexity sorting methods led to the exact same results as all strategies led to the exact same ordering of the samples. Applying EDA to the complexity scoring strategy resulted only in a very small score improvement compared to the complexity strategy without EDA, namely from 78.47 to 78.51. When considering the results for the dataset consisting of 100 samples, Table 5.3 shows that the performance of the CL strategy without any sorting method is slightly worse than the baseline (No CL) performance (-1.1%). We expect this to be due to the limited sample, and consequently, bucket size in this training process. The first buckets determine the starting point for the parameter optimisation process and thus has a large effect on the final performance. We expect that the limited sample representation in the initial buckets, due to their small size, resulted in a too narrow representation of summaries for the model. The remainder of the training sample set seems to not be large enough to correct for this initial misalignment. This is also the case for the length, reduction, and complexity strategies, which all resulted in the same result (-1.5%) as all sorting methods ended up with an identical training bucket distribution. Applying EDA to the dataset in combination with complexity scoring boosted the performance by 2.0% in comparison to the complexity scoring strategy without EDA. The final sample size we considered in our research consists of 1000 samples as shows in Table 5.3. With a dataset of this size we see the hypothesised performance improvements resulting from the Curriculum Learning strategy and EDA. The complexity sorting strategy results in an performance improvement of 4.0% compared to the baseline performance of no curriculum learning strategy. Extending the curriculum learning strategy with the EDA techniques increased the performance further to a total performance increase of 6.4% compared to the the no curriculum learning baseline.

The Baby-Steps strategy seems to improve model performance more than the One-Pass curriculum in most situations. This is likely due to the nature of the algorithm which expands the training buckets instead of replacing them. Thus, if the number of epochs is equal, the Baby-Steps curriculum will see the simpler training samples more often than the One-Pass curriculum. Especially in a low-resource setting, as we are considering, it is not surprising that this has a positive effect on performance. The samples consisting of 10 and 100 samples sizes showed similarly limited performance differences as with the one pass curriculum learning strategy. A major difference between the two strategies is the effect EDA has on the performance. With the one pass curriculum learning strategy the performance improvement between the complexity sorting method without EDA and with EDA for the sample sizes of size 10 and 100 are 0.1% and 2.4%, respectively. When considering the same effects for the baby-steps algorithm, we see performance improvements of 3.1% and 5.8% for these two datasets. This again indicates the increased exposure the model has to the simpler examples

Table 5.3: The results of the abstractive summarization tasks. The CL strategy indicates the curriculum learning strategy we used, either One-Pass (CLOP) or Baby Steps (CLBS). The EDA column indicates whether the data has been augmented with the EDA techniques. The sorting method column shows the data sorting method for the curriculum learning strategy, where complexity refers to our novel complexity scoring algorithm and length and reduction refer to the baseline sorting methods. The score column shows the sum of R_1, $2 * R_2$, and R_L. All results were achieved with a batch size of 2 and learning rate of 5e−4.

CL strategy	**EDA**	**Sorting method**	**10 samples**				**100 samples**				**1.000 samples**			
			R_1	R_2	R_L	Score	R_1	R_2	R_L	Score	R_1	R_2	R_L	Score
No CL	No	None	32.14	12.33	21.44	78.24	32.41	12.58	22.27	79.84	33.21	13.45	23.17	83.28
CLOP	No	None	32.15	12.29	21.40	78.13	32.16	12.32	21.44	78.24	33.25	13.74	23.38	84.11
CLOP	No	Length	32.22	12.39	21.47	78.47	32.05	12.24	21.40	77.93	33.33	13.46	23.11	83.37
CLOP	No	Reduction	32.22	12.39	21.47	78.47	32.05	12.24	21.40	77.93	33.73	13.76	23.30	84.56
CLOP	No	Complexity	32.22	12.39	21.47	78.47	32.02	12.22	21.47	77.93	34.31	14.25	23.78	86.60
CLOP	Yes	Complexity	32.24	12.38	21.51	78.51	32.42	12.67	22.07	79.83	34.74	14.72	24.46	88.63
CLBS	No	None	32.10	12.27	21.50	78.14	32.49	12.60	22.16	79.85	34.65	13.74	23.04	85.17
CLBS	No	Length	32.06	12.22	21.50	78.00	32.39	12.54	22.08	79.55	34.05	13.42	23.43	84.33
CLBS	No	Reduction	32.06	12.22	21.50	78.00	32.39	12.54	22.08	79.55	35.51	14.16	24.08	85.48
CLBS	No	Complexity	32.06	12.22	21.50	78.00	32.13	12.38	21.85	78.74	33.89	14.20	23.93	87.99
CLBS	Yes	Complexity	32.77	12.79	22.06	80.41	33.39	13.39	23.10	83.27	36.56	13.87	24.43	88.73

is strengthened further by the application of EDA techniques.

An interesting aspect to highlight is the effect of the sorting methods. For this examination we consider the CLOP and CLBS strategies without any sorting as our baseline performances. In this comparison we ignore the dataset of 10 samples as the sorting methods resulted in identical order for each method. In the CLOP strategy for 1000 samples, the length (-0.9%) and reduction (+0.5%) sorting methods show little difference in performance when compared to the non-sorted baseline. Thus, we can can conclude that in this setting these sorting methods have very limited effect on the model's performance. The complexity sorting algorithm does show a performance improvement of 3.0% compared to the One-Pass algorithm without any sorting. Thus, our complexity sorting algorithm seems to have a positive effect on the model's performance. Similar results are achieved with the CLBS strategy with 1000 samples, the length and reduction sorting method resulted in very limited performance differences of -0.2% and 0.4%, respectively, compared to the CLBS strategy with no sorting strategy. However, the complexity sorting algorithm does result in a performance improvement of 3.3% compared to the baseline without a sorting strategy.

Combining the individual effects described the in the previous paragraphs allows us to compare the final performance compared to the initial baseline performance of no curriculum learning strategy. The combination of methods that achieves the best performance is the CLBS strategy with a complexity sorting algorithm and with EDA techniques applied, which resulted in a performance gain of 6.5% compared to the baseline of no curriculum learning. The performance gain for the CLOP strategy with the complexity sorting algorithm and EDA applied is very close with 6.4%.

Appendix A shows a few examples of the model generated summaries. We see that the model is certainly able to shorten certain sections while maintaining the meaning a good fluency as is shown for example in the case where the original sentence "the football association will be contacting qpr and chelsea" was summarised by the model as "football association to contact qPr and chelsea". The generated texted is objectively shorter and we, of course subjectively, consider it fluent and equally informative as the original text. Furthermore, the human written summary includes the sentence "qpr unlikely to face disciplinary action over the incident" without any elaboration of what "the incident" is, which in our opinion is an incompleteness in the human written summary. The model generated summary does incorporate this information through the sentence "football association to contact qPr and chelsea after incident which saw Branislav ivanovic struck on the head objects were thrown onto the pitch", which we would consider a much more informative sentence about "the incident". Although these are clearly promising signs in the model generated summaries we simultaneously have to acknowledge that there are still simple mistakes left in the generated summaries. An example of this would be that both the text "objects were thrown onto the pitch by the crowd" and "objects being thrown onto the pitch by the crowd" were in generated summary A in appendix A.

Summary B in the appendix shows another example of a situation where the generated summary arguably outperforms the human written summary in a certain aspect. Here the human written summary includes the sentence " the girl,dressed in pink, tells man off camera she has killed 400 fighters". This information is captured in the generated summary through the sentence "young girl fires a machine gun at isis militants , claims to have killed 400 fighters". According to us, the model generated summary outperforms the human written summary here in terms of conciseness as the detail "dressed in pink" seems redundant and is not captured by the model generated summary.

In our opinion the best generated summary is generated summary C. We consider this to be an almost complete and fluent summary: "gold ring found at finns beach club in bali has been shared 23 thousand times on Facebook The message on the ring is: 'darling joe , happy 70th birthday 2009 ,love jenny' continues the search for joe and jenny". The only clear flaw we notice is of grammatical nature as the second sentence implies that "The message" continues a search for Joe and Jenny as opposed to the woman who wrote the Facebook post. However, besides this one error we consider it to contain all the relevant information and no redundant details.

The first two summaries highlight some situation in which the model performs well, however we must also acknowledge that that the model does still make errors on deciding which information to include in the summary as is shown in summary C. In this summary the model included the text "His owner , duane Smith , saystank is a real good one", which is not a piece of information that according to us belongs in a well written summary and which is also not present in the corresponding human written summary. Furthermore, generated summary D shows that the model can sometimes fail to deliver an understandable and informative summary as the generated summary “NEW : : ’s attorney says defendant ’s plea is not guilty to murder” is missing almost all relevant information which is present in the human written summary.

Chapter 6

Concluding Remarks

In this final chapter we give some concluding remarks about our research. In Section 6.1 we present the conclusions that we draw based on the findings described in this paper. Additionally, multiple topics for further research are outlined in Section 6.2.

6.1 Conclusion

In this research we investigated whether the state-of-the-art summarization models could be improved by combining these models with methods that are aimed at improving the training efficiency. We did this by applying curriculum learning strategies in combination with data augmentation techniques and in order to do so we introduced a novel text-summary pair complexity scoring algorithm.

Due to resource limitation we were only able to investigate the effects on the state-of-the-art models in a setting with a lower number of training steps than the amount with which the model achieved the state-of-the-art results. In this setting we found that our methods were able to improve the model's performance and thus, under the assumption that these improvements would hold up when increasing the number of training steps, allow the model to achieve new state-of-the-art performance.

We believe that our novel complexity scoring system for text-summary pairs is an important step which opens up extensive possibilities for further research into curriculum learning strategies within text summarization, as well as other applications where a ranking system is useful. We found that the optimal operation weight assignment gave a much higher weight to word reordering and word substitution operations, compared to the word deletion and word addition operations. This implies that the proficiency of a model in the former two tasks are much more indicative of the summary quality of the model, measured in ROUGE F1 scores, than the latter two tasks.

We examined the performance of this complexity scoring system by comparing it with baseline performances of the state-of-the-art PEGASUS model and with baseline scoring systems based on the length and reduction measurements of the test-summary pairs. We found that our complexity scoring system outperformed the baseline sorting methods up to 5.7% with 1,000 samples without EDA techniques. Applying the EDA techniques in combination with the complexity sorting algorithms increasing this improvement up to 6.5%.

6.2 Further research

The first and foremost recommendation we would provide for further research is to apply the methodology presented in this paper and increase the length of the training process such that it can be directly compared to the low resource results as presented in Zhang et al. (2019). In order or achieve this the methods need to be replicated with parameters and hyperparameters comparable to those reported by Zhang et al.. This will show whether the performance improvements shown in this paper result in similar improvements when the models are trained longer and thus whether it results in new state-of-the-art performances.

In addition, the optimisation of the weights in the complexity scoring system should be investigated further. A more elaborate exploration of the optimal weights using the random search technique applied in this paper could reveal better weight combinations that lead to higher performing curriculum learning strategies. Additionally, other methods could be applied to find these weight combinations such as grid search, Bayesian optimisation, or meta-heuristic algorithms for optimisation such as Genetic algorithms. A second aspect of the weight optimisation process which could be improved upon in further research is the evaluation of the weight combinations, due to resource limitations the performance of each weight combination was evaluated on a smaller training set than the actual task. Although, this is a common approach it may result in a weight combination that is not actually the combination leading to the best performing curriculum learning strategy once it is applied to the final task. Therefore, an evaluation of the weight combination in a setting which is more closely related to the actual task may provide more insightful results.

Additionally, we would recommend further research into the optimal representation of the quality of a summary such that easy comparison is possible, e.g., by having the quality represented by a single number. In this paper we used a combination of the ROUGE-1, ROUGE-2, and ROUGE-L F1 scores, with more importance assigned to the ROUGE-2 F1 score. To the best of our knowledge, no research has investigated whether this combination of these F1 scores is the best representation of the quality of a summary. We believe that this would be a valuable contribution to the text summarization literature. Firstly, an investigation could be done into the relevant ROUGE F1 scores, or other measures, that would need to be included in this representation. E.g., besides the ROUGE-N and ROUGE-L measures, Lin (2004) additionally introduced the ROUGE-W and ROUGE-S measures, which are rarely considered in summarization evaluation. ROUGE-W measures a weighted longest common sub-sequence, which allows us to differentiate between consecutive and non-consecutive common sub-sequences. ROUGE-S measures the Skip-bigrams, which are pairs of words that can have arbitrary gaps between them. Secondly, the relative importance of each score could be researched such that they can be combined into a single value.

Lastly, we would consider further research concerned with low resource text summarization, where well performaning models are combined with techniques aimed at improving training efficiency, to be a valuable addition to the existing literature. The methods presented in this paper could be applied to other datasets, both English datasets and other languages, in order to analyse whether the results achieved with the datasets applied in our research lead to similar results with other data. Furthermore, there are other techniques available to improve performance with low resource datasets, including active learning or other curriculum learning strategies such as "Less is More" and "Leapfrog" (Spitkovsky et al., 2010). All of these research efforts would contribute to making high quality text summarization more accessible in real world application without requiring enormous training datasets.

Bibliography

Aftab, U. and Siddiqui, G. F. (2018). Big data augmentation with data warehouse: A survey. In *International Conference on Big Data (IEEE 2018)*, pages 2775–2784. IEEE.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, (ICLR 2015)*.

Banko, M., Mittal, V. O., and Witbrock, M. J. (2000). Headline generation based on statistical translation. In *38th Annual Meeting of the Association for Computational Linguistics (ACL 2000)*, pages 318–325. ACL.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *26th Annual International Conference on Machine Learning (ICML 2009)*, volume 382 of *ACM International Conference Proceeding Series*, pages 41–48. ACM.

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

Cheng, J., Dong, L., and Lapata, M. (2016). Long short-term memory-networks for machine reading. *CoRR*, abs/1601.06733.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1724–1734. ACL.

Cirik, V., Hovy, E. H., and Morency, L. (2016). Visualizing and understanding curriculum learning for long short-term memory networks. *arXiv preprint*, arXiv:1611.06204.

Cohn, T. and Lapata, M. (2008). Sentence compression beyond word deletion. In *22nd International Conference on Computational Linguistics (COLING 2008)*, pages 137–144. ACL.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Cormen, T. H., Leiserson, C. E., and Rivest, R. L. (1989). *Introduction to Algorithms*. The MIT Press and McGraw-Hill Book Company.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In *2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2019)*, pages 4171–4186. ACL.

Elman, J. L. (1993). Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99.

Gambhir, M. and Gupta, V. (2017). Recent automatic text summarization techniques: a survey. *Artificial Intellingence Review*, 47(1):1–66.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). In *34th International Conference on Machine Learning (ICML 2017)*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252. PMLR.

Graves, A., Bellemare, M. G., Menick, J., Munos, R., and Kavukcuoglu, K. (2017). Automated curriculum learning for neural networks. In *34th International Conference on Machine Learning (ICML 2017)*, volume 70 of *Proceedings of Machine Learning Research*, pages 1311–1320. PMLR.

Hermann, K. M., Kociský, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. (2015). Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems (NIPS 2015)*, pages 1693–1701.

Hochreiter, S., Bengio, Y., Frasconi, P., and Schmidhuber, J. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press.

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

Jean, S., Cho, K., Memisevic, R., and Bengio, Y. (2015). On using very large target vocabulary for neural machine translation. In *53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015) and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (IJCNLP 2015)*, pages 1–10. ACL.

Khandelwal, U., Clark, K., Jurafsky, D., and Kaiser, L. (2019). Sample efficient text summarization using a single pre-trained transformer. *arXiv preprint*, arXiv:1905.08836.

Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81. ACL.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *27th International Conference on Machine Learning (ICML 2010)*, pages 807–814. Omnipress.

Nallapati, R., Zhai, F., and Zhou, B. (2017). Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *31st Conference on Artificial Intelligence, (AAAI 2017)*, pages 3075–3081. AAAI Press.

Nallapati, R., Zhou, B., dos Santos, C. N., Gülçehre, Ç., and Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond. In *20th SIGNLL Conference on Computational Natural Language Learning (CoNLL 2016)*, pages 280–290. ACL.

Pentina, A., Sharmanska, V., and Lampert, C. H. (2015). Curriculum learning of multiple tasks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2015)*, pages 5492–5500. IEEE Computer Society.

Platanios, E. A., Stretcu, O., Neubig, G., Póczos, B., and Mitchell, T. M. (2019). Competence-based curriculum learning for neural machine translation. In *2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019)*, pages 1162–1172. ACL.

Press, O. and Wolf, L. (2017). Using the output embedding to improve language models. In *15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pages 157–163. ACL.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint*, arXiv:1910.10683.

Ramirez, J. M., Montalvo, A. R., and Calvo, J. R. (2019). A survey of the effects of data augmentation for automatic speech recognition systems. In *Pattern Recognition, Image Analysis, Computer Vision, and Applications (CIARP 2019)*, volume 11896 of *Lecture Notes in Computer Science*, pages 669–678. Springer.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Rush, A. M., Chopra, S., and Weston, J. (2015). A neural attention model for abstractive sentence summarization. In *2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 379–389. ACL.

Sainath, T. N., Vinyals, O., Senior, A. W., and Sak, H. (2015). Convolutional, long short-term memory, fully connected deep neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2015)*, pages 4580–4584. IEEE.

See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. In *55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 1073–1083. ACL.

Shazeer, N. and Stern, M. (2018). Adafactor: Adaptive learning rates with sublinear memory cost. In *35th International Conference on Machine Learning (ICML 2018)*, volume 80 of *Machine Learning Research*, pages 4603–4611. PMLR.

Shi, T., Keneshloo, Y., Ramakrishnan, N., and Reddy, C. K. (2020). Neural abstractive text summarization with sequence-to-sequence models: A survey. *arXiv preprint*, arXiv:1812.02303v3.

Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6:60.

Smith, D. A. and Eisner, J. (2006). Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Workshop on Statistical Machine Translation (WMT 2006)*, pages 23–30. ACL.

Spitkovsky, V. I., Alshawi, H., and Jurafsky, D. (2010). From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *17th Annual Conference of the North American Chapter of the Association of Computational Linguistics: Human Language Technologies (NAACL-HLT 2010)*, pages 751–759. ACL.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *27th Annual Conference on Neural Information Processing Systems (NIPS 2014)*, pages 3104–3112. Curran Associates, Inc.

Tu, Z., Lu, Z., Liu, Y., Liu, X., and Li, H. (2016). Modeling coverage for neural machine translation. In *54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 76–85. ACL.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *31st Annual Conference on Neural Information Processing Systems (NIPS 2017)*, pages 5998–6008. Curran Associates, Inc.

Wang, Q., Liu, P., Zhu, Z., Yin, H., Zhang, Q., and Zhang, L. (2019). A text abstraction summary model based on bert word embedding and reinforcement learning. *Applied Sciences*, 9(21):4701.

Wang, X., Wang, K., and Lian, S. (2020). A survey on face data augmentation for the training of deep neural networks. *NCA*, 32(19):15503–15531.

Wei, J. and Zou, K. (2019). EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388. ACL.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Woodsend, K., Feng, Y., and Lapata, M. (2010). Title generation with quasi-synchronous grammar. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*, pages 513–523. ACL.

Xhafa, F., Naranjo, V., and Caballé, S. (2015). Processing and analytics of big data streams with Yahoo!S4. In *29th International Conference on Advanced Information Networking and Applications (AINA 2015)*, pages 263–270. IEEE.

Zaremba, W. and Sutskever, I. (2014). Learning to execute. *arXiv preprint*, arXiv:1410.4615.

Zhang, J., Zhao, Y., Saleh, M., and Liu, P. J. (2019). PEGASUS: pre-training with extracted gap-sentences for abstractive summarization. *arXiv preprint*, arXiv:1912.08777.

Appendix A

Summarization results

Table A.1 shows a sample of the results that our best model was able to achieve in a low-resource setting of 1000 training samples. This is the model trained through a baby-steps curriculum learning approach where data augmentation has been applied as described in chapter 4. The table shows four examples of text-summary pairs used in the evaluation of our models performance. For each example the table contains the original text as it has been prepared as input for our model, a human written summary, and a summary generated by the model.

Table A.1: This table shows a sample of the results. The table contains the input, human written summary, and the model generated summary. The model used for these summaries was the best performing model, thus the model that was trained with a baby steps curriculum learning approach based on a complexity-based sorting method and a dataset consisting of 1,000 samples which have been augmented with data augmentation techniques.

Description	**Text**
Input text A (1/2)	the football association will be contacting qpr and chelsea after an incident which saw branislav ivanovic struck on the head by a cigarette lighter thrown from a section of home supporters at loftus road. qpr are unlikely to face disciplinary action over the incident which happened as chelsea players celebrated the winning goal in sunday's 1-0 victory. it is understood however that the fa will seek observations from each club and will ensure that qpr and the police are taking the necessary steps to identify the culprit via cctv and for them to face a banning order. jubilant chelsea celebrations were marred by objects being thrown onto the pitch by the crowd . during the chelsea celebrations at loftus road, the blues were targeted by objects thrown from the crowd . qpr are reviewing cctv footage and have promised to ban those involved. they have also appealed to witnesses to come forward. a club spokesperson said: 'whilst we understand that passions run high during derby fixtures, this is not the kind of behaviour the club condones in any way, shape or form.

Input text A (2/2)	'we take a zero tolerance approach to this sort of behaviour and, if the perpetrators are identified, they will be banned indefinitely from the stadium and, where appropriate, we will liaise with the metropolitan police. chelsea duo nemanja matic (left) and branislav ivanovic react after being struck with objects . chelsea midfielder nemanja matic hands over a coin to referee andre marriner during the match on sunday . the loftus road pitch was covered in litter and objects following the west london derby contest on sunday . 'the club are awaiting communication from the fa following the referee's report and will work closely with them in ensuring something like this does not happen again. 'we would also appeal to fans with any information regarding yesterday's incident to contact the club, as we feel it's important to identify the perpetrators as soon as possible, as we believe they are not a true representation of the qpr fan-base.'
Human written summary A	qpr unlikely to face disciplinary action over the incident . queens park rangers to review cctv and promise to ban anyone involved . cesc fabregas scored a late winner for chelsea at loftus road .
Generated summary A	football association to contact qPr and chelsea after incident which saw Branislav ivanovic struck on the head objects were thrown onto the pitch by the crowd during chelsea ' victory was marred by objects being thrown onto the pitch by the crowd
Input text B (1/2)	this is the shocking moment a young kurdish girl fires a machine gun at isis, before boasting she has killed 400 of the islamic militants - as she is egged on with cries of 'kill, kill'. the little girl, who appears to be around six or seven, is shown sitting on a ridge in her bright pink jumper and trainers, just behind the large machine gun. she chats away to a man behind the camera, telling him she is 'shooting at daesh' in kurdish - before firing several rounds into the desert. shocking: the little girl claims she is firing the machine gun at 'daesh', the arabic name for isis . kill: she tells the man she has killed 400 of the extremists so far, as he encourages her to kill more . target: however, it is unclear exactly what the girl is shooting at in the minute-long video . the man asks her how many isis fighters she has killed so far, and she holds up four fingers while boasting '400'. as she fires the gun, and, according to vocativ.com, he says: 'kill, kill.'however, whether she has actually hit anything is unclear. the video - entitled young ypg girl shoots pk machine gun - was posted in january, but has only recently emerged on social networks. ypg is the acronym for the people's protection unit, which has been battling isis since the extremists rise to power last year. but while isis are known for using young recruits to carry out executions, fight and even become suicide bombers, the kurds have so far kept their children out of the battle. propaganda: isis are known to use children in videos like this one, where they are being trained as fighters . sickening: in more horrific videos, isis have claimed to be using children as executioners . feared: the kurdish are famous for their female fighters, who have been key to driving isis out . rumours: isis fighters believe if they are killed by a woman they won't get their virgins in the afterlife .

Input text B (2/2)	one of the most horrifying videos released by the islamists showed a young boy seeming to execute a so-called spy. they have also released a video of dozens of 'child soldiers' from kazakhstan, apparently being trained to fight against isis enemies. instead, they are famous for their female recruits: it has proved to be one of their strengths against the militants, as they appear to believe if they are killed by a woman they will not be rewarded with 72 virgins in the afterlife. as a result isis fighters have been reluctant to enter battles with the women's protection units, or ypj, reportedly leading to descent within the ranks of the terrorists and hastening their retreat from battles in places like kobane.
Human written summary B	video of little girl firing several rounds 'towards isis' has gone viral . the girl, dressed in pink, tells man off camera she has killed 400 fighters . the man encourages her by saying 'kill, kill' as she shoots machine gun . kurdish known for their female fighters - but isis known for using children .
Generated summary B	young girl fires a machine gun at isis militants , claims to have killed 400 fighters , in video posted to social networks Video was posted to social networks in January , but has only recently emerged
Input text C	the ocean is known for claiming countless of treasured items, particularly rings and sentimental jewellery which remain lost forever. however, an australian woman is on a campaign to reunite a lost ring with its owner after discovering the buried treasure deep in the ocean. 'so here's a super long shot,' queensland resident roxy walsh wrote on facebook. 'found this gold ring snorkelling at finns beach club in bali today (april 7).' roxy walsh shared a photo of a gold ring she found on facebook in the hope of finding its owner . roxy walsh (pictured in bali) found the ring while snorkelling at finns beach club in bali on april 7 . the ring has what appears to be a family crest and has been engraved with the heartfelt message: 'darling joe, happy 70th birthday 2009, love jenny'. the facebook message has already been shared an astonishing 23 thousand times, taking the search for the special ring global. 'sometimes hear of these things finding their way home so worth a shot,' ms walsh wrote. travel blog the bali bible shared the post to more than 171 thousand followers. 'it would be great if all of you amazing people out there could (share the picture of the ring) and hopefully reunite this ring with its owners, joe or jenny asap..! nice find, roxy! respect reunite .' the search for joe and jenny continues. travel blog the bali bible shared the post to more than 171 thousand followers .
Human written summary C	queensland woman roxy walsh found an inscribed gold ring in bali . the sentimental jewellery piece was found in the ocean while snorkelling . ms walsh has launched a campaign to return the ring to the people who own it, hoped to be "joe" or "jenny" according to inscription . facebook post has already been shared by more than 23 thousand people .
Generated summary C	gold ring found at finns beach club in bali has been shared 23 thousand times on Facebook The message on the ring is: 'darling joe , happy 70th birthday 2009 , love jenny' continues the search for joe and jenny

Input text D	the pups came from as far away as new jersey and tennessee in hopes of landing a new nickname: 'beautiful bulldog.' they were all defeated by a native who likes eating snow and watching turtles. a 2-year-old dog from des moines named tank won the 36th annual beautiful bulldog contest sunday at drake university. scroll down for video . winner: tank, a 2-year-old bulldog from iowa won drake university's 36th annual 'beautiful bulldog' contest sunday . a real beauty: tank, who enjoys eating snow and watching turtles, will now serve as mascot of this year's drake relays . tank received top honors as well as a crown and cape. he will appear before more than 16,000 fans – or, royal subjects – at the university's drake relays to be honored as mascot of the event, which will be held from thursday through saturday, according to the contest's website. the tongue-in-cheek beauty pageant, which featured 50 dogs, is the kickoff event for the drake relays track meet. 'he's funny,' said tank's owner, duane smith. 'he's a real good one.' pageant organizers narrowed a pool of more than 100 hopeful pups by a lottery held last month. owner: here, tank enjoys a rub from his owner, duane smith, after winning the 36th annual contest . judges weren't looking for beauty though. they wanted to see the slobber, drool and bulging, bloodshot eyes synonymous with english bulldogs. they got all that and more from tank — who now willingly shares his house with a pomeranian and some turtles after smith found him on craigslist a year ago. should tank be unable to fulfill his duties as the drake relays mascot, fellow des moines pup steve will step in. steve was second even though he was initially so shy about the makeshift catwalk set up on drake's basketball court that his owner had to pick him up and plunk him down, much to the delight of the few thousand spectators on hand. pageant: the tongue-in-cheek beauty pageant, which featured 50 dogs, is the kickoff event for the drake relays track meet . there also was a 'best dressed' winner in linus the lovebug — who had to be dragged around in a wagon because of arthritic legs — and the congeniality award went to a dog named princess mabel. if there was an award for driving the furthest to enter the contest it would've gone to ronnie sussman and her dog bex, who drove 17 hours from union, new jersey. sussman and bex will go home empty-handed, but she said the trip was more than worth it. 'this is just like a bucket list item of life for me,' sussman said.
Human written summary D	an iowa bulldog named tank took home the crown sunday at drake university's annual 'beautiful bulldog contest' tank beat out 49 other dogs in the 36th annual contest . tank will now serve as the mascot for the drake relays .
Generated summary D	tank , a 2-year-old dog from des moines , Iowa , wins drake university ' beautiful bulldog' contest tank gets the crown and cape His owner , duane Smith , says tank is a real good one

Input text E	a retired marine has pleaded not guilty to murdering his clothing-designer girlfriend, whose skeletal remains were discovered in a jungle in panama two years after she disappeared. brian karl brimager, 37, entered his plea on friday after he was indicted by a federal grand jury in san diego, in connection with the death of yvonne lee baldelli. he has been in custody since june 2013 on charges including obstruction of justice and falsifying records related to the investigation. the 42-year-old woman from laguna niguel, california, was last seen in september 2011 when she arrived in panama with brimager. her family reported her missing the following january. brian karl brimager, 37, was indicted by a federal grand jury in san diego, in connection with the death of yvonne lee baldelli . the 42-year-old woman from laguna niguel, california, was last seen in september 2011 when she arrived in panama with brimager. her family reported her missing the following january . the indictment alleges that brimager murdered baldelli, dismembered her body and disposed of her body parts in a remote jungle area. brimager then engaged in an elaborate scheme to cover up the crime, including destroying evidence and sending a series of emails purportedly from baldelli in order to make it appear to her friends and family that she was still alive, according to court papers. the indictment also alleges that brimager attempted to conceal his crime by disposing of a bloody mattress involved in badelli's murder in the ocean. brimager conducted two internet searches on baldelli's computer, one for 'washing mattress' and a second for 'washing mattress blood stain,' according to documents. a man who was cutting bushes on the island province of bocas del toro found a bag containing baldelli's remains in august 2013. baldelli's family has the clothing designer was frequently out of touch so they did not immediately suspect anything was wrong. panama police say brimager left panama for costa rica and the united states about 10 days after last being seen with baldelli. in 2012 fbi agents and panamanian forensic specialists found traces of blood on the walls and floor of the hostel el sapo in the bocas del toro archipelago, which is the popular tourist spot where baldelli was last seen. brimager (right) has been in custody since june 2013 on charges including obstruction of justice and falsifying records related to the investigation . the indictment alleges that brimager murdered baldelli (pictured), dismembered her body and disposed of her body parts in a remote jungle area . panama police say brimager left panama for costa rica and the united states about 10 days after last being seen with baldelli . sorry we are not currently accepting comments on this article.
Human written summary E	brian karl brimager, 37, was indicted by a grand jury in san diego . is accused of murdering clothing designer yvonne lee baldelli in 2011 . allegedly dismembered her body and disposed of it in a military backpack . then engaged in an elaborate scheme to cover up the crime . sent emails from her account to make people think she was still alive . he has been in custody since june 2013 on charges including obstruction of justice and falsifying records related to the investigation .

Generated summary E	NEW : : 's attorney says defendant 's plea is not guilty to murder