LATS: Low Resource Abstractive Text Summarization

Chris van Yperen[a], Flavius Frasincar[a,*], Kamilah El Kanfoudi[b]

[a]*Econometric Institute, Erasmus University Rotterdam, P.O. Box 1738, NL-3000 DR Rotterdam, the Netherlands*
[b]*Tinbergen Institute, Gustav Mahlerplein 117, 1082 MS Amsterdam, the Netherlands*

Abstract

Text summarization is an increasingly crucial focus of Natural Language Processing (NLP), and state-of-the-art models such as PEGASUS have demonstrated remarkable potential to ever more efficient and accurate abstractive summarization. Nonetheless, recent developments of deep learning models that focus on training with large datasets can become at risk of sub-optimal generalization, inefficient training time, and can get stuck at local optima due to high-dimensional non-convex optimization domains. Current research in the field of NLP suggests that leveraging curriculum learning techniques to guide model training (enabling the model to learn from training data with increasing difficulty) could provide a means to achieve enhanced model performance. In this paper we investigate the effectiveness of curriculum learning strategies and data augmentation techniques on PEGASUS to increase performance with low-resource training data from the CNN/DM dataset. 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 and boosts performance of PEGASUS. The Baby-Steps curriculum learning strategy with this sorting method leads to performance improvements of 5.65%, from a combined ROUGE F1-score of 83.28 to 87.99. When this strategy is combined with a data augmentation technique, Easy Data Augmentation, this leads to an improvement to 6.54%. These statistics are relative to a baseline without curriculum learning or data augmentation.

Keywords:
abstractive text summarization, curriculum learning, low-resource summarization, complexity scoring

1. Introduction

The exponential growth of available information, largely textual, distributed through the Internet has postulated a challenge in effective information processing. This phenomena leads to questions about how can we effectively capture, understand, and distribute the increasingly large and diverse textual resources available to users. Therefore, text summarization has emerged an ever-more critical task to address this, and more efficient algorithms can be an important aid (El-Kassas et al., 2021).

*Corresponding author; tel: +31 (0)10 408 1340; fax: +31 (0)10 408 9162
Email addresses: chrisvanyperen@gmail.com (Chris van Yperen), frasincar@ese.eur.nl (Flavius Frasincar), 597783kk@student.eur.nl (Kamilah El Kanfoudi)

Text summarization comes in two forms, extractive and abstractive (Gambhir & Gupta, 2017). The former requires the model to simply extract the most salient words or phrases, and composes a summary from existing text. The latter method requires the model to construct a semantic characterization of the text and generates a representative summary, allowing for newly generated vocabulary (Srivastava et al., 2022). Generally, the latter technique results in more grammatically fluent summaries and is therefore preferable, despite its relative computational complexity (Moratanch & Chitrakala, 2016). Therefore, our work focuses on abstractive summarization.

A vast majority of the literature concerned with Natural Language Processing (NLP) tasks, including text summarization, explores the performance of models and techniques based on large English training corpora of several hundred thousand training examples. Although using large volumes of data is a sensible approach in academic literature, in real-world applications it is often difficult and costly to collect a sufficient 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 real-world applications (Zhang et al., 2020). Bridging the gap would unlock the potential harbored within the range of textual data scattered across organizations with fewer training samples, and provide a means to sustain model performance with fewer computing resources.

Due to the "data-hungry" (Vaswani et al., 2017) nature of neural methods, such as Transformers (which rely on relative dependencies in sequence position), the challenge of a low-resource environment generally sacrifices model performance (Tang et al., 2024). 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. Firstly, 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 follows from the idea that a machine can be more effectively trained by emulating the way that humans learn. This is done by presenting the training data ordered from easy samples to more complex, instead of in randomly sampled batches, using approaches such as One-Pass (Bengio et al., 2009) and Baby-Steps (Spitkovsky et al., 2010). Another approach includes artificially expanding the dataset 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 (Ramirez et al., 2019b) (Aftab & Siddiqui, 2018) (Ramirez et al., 2019a). Here, data augmentation techniques such as Easy Data Augmentation (EDA) (Wei & Zou, 2019) can be applied to generate altered versions of the text-summary pair training data from which the model can extract new information and leads to further performance improvements. Overall, 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. The code is written in Python and made freely available at `https://github.com/CBvanYperen/LATS`.

The remainder of this paper is organised as follows. Section 2 describes the existing literature regarding abstractive text summarization, data augmentation techniques, and curriculum learning strategies. We describe the data we used and collected in order to perform our research in Section 3. Section 4 lays out our methodology, where we describe the model type, pre-training and training methods, as well as the performance evaluation techniques. In Section 5 we present the results and discuss our findings. Last, in Section 6 we conclude our paper and mention subjects for future research.

The major contributions of this paper can be summarized as follows:

- Introduced a novel complexity sorting algorithm for a continuous ex-ante definition of relative text complexity based on four key operations that consistently outperforms baseline sortings;
- Analysed Transformer model optimization within low-resource environments with a range of combinations of curriculum learning strategies, sorting techniques, and with and without EDA;
- Demonstrated the optimal combination for model performance of curriculum learning with Baby-Steps and EDA.

2. Related Work

In this section we outline the current related literature. The section is divided into three subsections which address the literature concerned with abstractive text summarization, curriculum learning, and low-resource text summarization.

2.1. Abstractive Text Summarization

Some of the earliest work on Abstractive Text Summarization (ATS) uses traditional phrase-based machine translation approaches to generate newspaper article headlines (Banko et al., 2000). This became one of the foundational approaches towards the text summarization task, as it is analogous to translation where the target language is Compact English.

Cohn & Lapata (2018) expanded the scope of ATS by building on sentence compression, which previously aimed only on optimising sequences of deletions in texts. The research focused on generating abstracts, considering also operations of substitutions, reorderings, and word insertion through a tree-to-tree sentence compression task. This is done through synchronous tree substitution grammar rules (Eisner, 2003) between tree nodes to model syntactic expressions and abstractively create sentence summaries. This method was later extended to relax the basis on strict structural correspondence between source and target languages. Quasi-synchronous substitution grammar rules, as introduced by Woodsend et al. (2010) allow for flexible correspondence which can handle non-isomorphic structures and do not require a perfect structural alignment between source and target constituents. This method is able to effectively operate at a phrase level, in contrast to the synchronous model. As phrases are shorter than sentences, ATS using the latter technique

tends to read more fluently. This is because important phrases, not whole sentences, are prioritized in the summarization task.

Collobert et al. (2011) shifted the approach in ATS by proposing deep learning algorithms for text summarization, effectively reducing the task-specific nature of the model and reliance on linguistic knowledge. Consequently, researchers have continued to navigate the potential for deep learning as an effective approach for a breadth of NLP problems. The trend towards a deep learning architecture in the literature was further developed by Sutskever et al. (2014) whose work introduced sequence-to-sequence (seq2seq) models with an encoder-decoder architecture based on Recurrent Neural Networks (RNNs) (Rumelhart et al., 1986). This became the dominant framework for ATS (Zhang et al., 2020) 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. One example is the work of Chopra et al. (2016) which reported notable results with an attention-based summarization approach. This was built upon the work of Bahdanau et al. (2015) which employed an attention-based decoder, using a latent soft alignment over the vector encodings to gain additional contextual information for a more efficient network.Likewise, Nallapati et al. (2017) built further upon the attentional RNN encoder-decoder model (Bahdanau et al., 2015). Nallapati et al. (2017) made several improvements to the original model to address specific problems not considered in the former model. The authors implemented the "large vocabulary trick" (Jean et al., 2015) to reduce computational constraints in the softmax layer. Additionally, the authors propose a switching generator-pointer architecture to handle unseen or rare words during training.

PEGASUS, introduced by Zhang et al. (2020), showed state-of-the-art text summarization results measured by ROUGE-F1 scores across all twelve considered data sets and set a precedent for the future of ATS models. The PEGASUS model is a seq2seq encoder-decoder based on Transformers (Vaswani et al., 2017). It leverages a novel pre-training and fine-tuning paradigm, known as Gap Sentences Generation (GSG) and a denoising auto-encoder, to achieve remarkable results in generating coherent and informative summaries. One of the key strengths of PEGASUS is its ability to generate abstractive summaries while incorporating important information from the source text. The model utilizes a novel GSG technique, where it predicts missing sentences from the input document, ensuring that important content is not missed during the summarization process. This model signified a paradigm shift in the field as successive models have built upon this base architecture to enhance performance. Wang et al. (2022) introduced an additional component which allocated salience estimators across sentences in input documents to guide ATS through more accurate self-attention. He et al. (2023) introduces the Fourier Transformer, which incorporates spectral filters to achieve more effective computation of the Transformer hidden state vectors. Zhao et al. (2023) incorporates the coordination target in the PEGASUS_{2B} model through an additional step appended to the original model, called Sequence Likelihood Calibration. Zhao et al. (2025) builds a on the Transformer archiecture by using a bidirectional decoder to better represent the characterstics of natural language. These recent developments of ATS are derivatives of the prevailing standard of the Transformer architecture. For this reason, we take

the base model of PEGASUS and aim to boost its performance using CL and EDA.

2.2. Curriculum Learning

The concept of Curriculum Learning (CL) was first introduced by Elman (1993), denoted in their work as incremental learning. Elman (1993) demonstrated improved neural network performance by gradually adding more complex observations, whereas a model that was trained with all (simple and complex) observations in random order showed very poor performance. The work of Bengio et al. (2009) has been accredited to formalizing our understanding of CL in the context of machine learning. As described by Bengio et al. (2009), CL acts essentially as an implementation of a continuation method, providing a means to explore the solution landscape of non-convex optimization problems systematically and can help overcome challenges associated with local optima and non-convexity.

As the complexity of the machine learning tasks have increased over the recent years, CL has received more attention due to its potential for aiding models to achieve improved convergence (MacAvaney et al., 2020), increased robustness (Wang et al., 2021a), and mitigating catastrophic forgetting (Kirkpatrick et al., 2017) (Lesort, 2020). Recent works have also made steps towards creating a generalized benchmark for CL evaluations across domains and baseline models (Zhou et al., 2024). However, Zaremba & Sutskever (2014) found that a straightforward application of CL to RNNs with LSTM units did not consistently improve network performance. The research underlines that it is imperative to form a CL strategy suited for the task. For neural networks performing addition and memorization tasks, it is beneficial for the network to distribute the information across its hidden state or memory cells, using a distributed representation. However, as the difficulty of the examples increases, the network may need to restructure its memory patterns to accommodate additional information. This process of memory pattern restructuring can be challenging to implement, which could explain why the naive CL strategy can show poor performance. The research introduces a combined strategy that addresses this issue by reducing the need for memory pattern restructuring.

Wang et al. (2021a) defined the landscape of CL methods by its approach toward two seminal components of CL architectures: Difficulty Measurer and Training Scheduler, separating Predefined CL (where both the Difficulty Measurer and Training Scheduler are fixed a priori) and Automatic CL (at least one of the components is driven in some manner by the algorithm). Wang et al. (2021a) describes the Training Schedulers (the order in which the model processes the data) in Predefined CL as separable into discrete and continuous variations. Discrete training schedulers separate the data into buckets, and after a fixed number of epochs introduce new data into the running training subset. Continuous Training Schedulers are functions that relate the training epoch to some scalar which indicates the proportion of instances (starting with 'easy') available at each stage of training (Hacohen & Weinshall, 2019). Automatic CL methods have been applied in different forms, each of which is characterized by the manner through which automation is applied in the CL process (Wang et al., 2020). One notable example is Self-Paced Learning (Kumar et al., 2010), in which the model concurrently selects easy samples and learns parameters for the latent variable model. The objective function involves optimizing a regularization function and a criterion for parameter

learning. Gradual Machine Learning is a similar method, but does not assume I.I.D or good coverage data and subverts this by iteratively inferring and labeling unlabeled instances based on their evidential certainty in a factor graph (Wang et al., 2021b). Other Automatic CL methods include those which incorporate an auxiliary model for the Difficulty Measurer (Tsvetkov et al., 2016) (Hacohen & Weinshall, 2019). Furthermore, Reinforcement Learning methods (Fang et al., 2019) (Milani et al., 2024) involve an even more dynamic system through a 'teacher' model (Wang et al., 2021a) that uses a reward algorithm to dynamically set both the Difficulty criterion and Training Scheduler, surveyed extensively by Narvekar et al. (2020). One example of the dynamic Teacher-Student architecture explores the problem of enabling an unknown Deep Reinforcement Learning (DRL) student to master a skill across various environments proposed by Portelas et al. (2020). The authors introduce an innovative teacher algorithm that addresses the challenge of creating a learning curriculum through sequential parameter sampling to generate diverse environments. The components of the CL approach are often also task-dependant. For example, Kwon et al. (2024) propose a curriculum construction method that relies on class-based frequency measures for a personality detection task. This way, the method allows to first identify what words are informative for personality classes. The priority of these words within their class is then used to construct a difficulty measure.

As CL methods have become more widely adopted and increasingly complex, Wang et al. (2021a) highlights the importance of a more robust theoretical understanding of CL, revealing relationships between its efficacy with varying noise in data distributions and model ablations. Furthermore, they aim to understand the manners through which CL interacts with stochastic gradient descent and how its possible conflicts can be regulated (Soviany et al., 2022). Lastly, the selection of an optimal CL strategy for real-life applications is still unclear, as the methods trade-off between effectiveness, cost, flexibility, and robustness (Soviany et al., 2022).

For this problem, we utilize a CL algorithm that enhances performance with fewer epochs and low computational cost. We construct a simple and effective CL architecture though a pre-defined Difficulty Measurer and a discrete Training Scheduler. Ranaldi et al. (2023) similarly defined an a-priori complexity metric using sentence length, word rarity, and readability as components. However, we believe our proposed metric is better defined for the summarization task as it relates directly to the transformation from text to summary, rather than text attributes themselves. We use Baby-Steps (Spitkovsky et al., 2010) and One-Pass (Bengio et al., 2009) for the curriculum framework. Both methods are intuitive meta-heuristics that guide the learning task and support increased convergence.

2.3. Low Resource Text Summarization

The difficulty in constructing datasets of sizes large enough to engineer and run ATS algorithms is particularly well documented for languages that are resource-poor (Kurniawan & Louvan, 2018). The ATS task requires a vast amount of labelled training data which makes the pre-trained models very good at a very specific task. Yet, without a similarly robust data set for downstream training, the performance tends to suffer. Thus, a multitude of methods have been explored to synthetically expand the working dataset. For

example, multiple studies have demonstrated performance improvements in NLP models through dataset expansion with iterative back translation (Hoang et al., 2018) (Zhang et al., 2018), noising (Kim et al., 2019) and resource-intensive autoencoder-based or GPT techniques (Kesgin & Amasyali, 2024). Wei & Zou (2019) introduced Easy Data Augmentation (EDA), which outlines a method based on defined operations (synonym replacement, random insertion, random swap, and random deletion). Although marginal on large data sizes, EDA is the most significant improvements in low-resource environments for both RNN's and Convolutional Neural Networks (CNNs).

There have been a multitude of models built upon the paradigms in ATS. However, optimizing in low-resource and zero-shot environments progresses much more slowly and few of the newest models have addressed this problem domain. Radford et al. (2019) GPT-2 model trained on WebText (consisting of a large amount of scraped webpages) achieved a ROUGE-2 F1 score of 8.27 on the CNN/DailyMail dataset in the zero-shot setting. Other research has utilized Transformer-based architectures for specific low-resource use cases (Su et al., 2025). Khandelwal et al. (2019) achieved a ROUGE-2 F1 score of 13.1 with a pre-trained decoder-only network with only 3000 samples. 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. In addition, many recent developments in ATS with promising performance on the CNN/Daily Mail dataset such as SimCLS (Liu & Liu, 2021) and SEASON (Wang et al., 2022) do not address the zero or few-shot predicton capability at all.

The aforementioned PEGASUS model (Zhang et al., 2020) 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. Developments in the PEGASUS model (Zhao et al., 2023) continue to surpass the ROUGE performance of alternate state-of-the-art ATS models such as BRIO (Liu et al., 2022) , thus making PEGASUS one of the best-performing models in low-resource abstractive text summarization. Moreover, PEGASUS is specifically architecturally suited for ATS due to its use of GSG, as apposed to models like BART that are general-purpose language models (Lewis et al., 2020).

In addition, PEGASUS is based on a Transformer encoder-decoder architecture, which makes it more versatile than decoder-only models like GPT-2, as monolithic design can be less adaptable to various tasks that require different processing for input and output sequences (Vaswani et al., 2017) (Radford et al., 2019). While capable of performing tasks like text generation and completion, a decoder-only model is less generalizable for tasks that require a transformation from one sequence to another, as it does not explicitly separate the encoding and decoding processes. Furthermore, GPT-2 is an example of a large language model, which requires extensive training. For these reasons, we built our model based on PEGASUS.

3. Data

In this section we describe the data sets 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 data sets, both

with their own requirements.

3.1. Pre-Training Corpus

In our research we utilize the pre-trained PEGASUS model (Zhang et al., 2020). In this study, the model is not pre-trained once again but has only been fine-tuned for the downstream summarization tasks. This model was pre-trained on a combination of the Colossal Clean Crawled Corpus (C4) (Raffel et al., 2020) and HugeNews (Zhang et al., 2020) data sets. The pre-training corpus consists of 1.85 billion texts stochastically sampled from the two corpora weighted by the individual data set 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.

3.2. Training Corpus

For the model fine-tuning, a data set is required that contains texts with a parallel summary. For this research we chose to utilize the CNN/DailyMail (CNN/DM) data set introduced by Hermann et al. (2015). The CNN/DM data set contains over 300,000 documents, all of which are articles from either the CNN (∼ 93,000) or DailyMail (∼220,000) news websites. Both news websites provide multi-sentence human generated summarizing sentences for each article, making this corpus particularly appropriate for ATS. This provides us with a gold standard on which the model can be trained and evaluated. Table 1 shows summary statistics for the CNN articles, DM articles, and the combined data set.

Table 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%]

4. Methodology

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 describes the pre-training method. Next, Section 4.2.1 describes the various CL strategies and

data augmentation steps applied during the model's training phase. Lastly, Section 4.2.2 describes the evaluation methods used to determine the performance of the models.

The workflow of the LATS architecture is depicted in Figure 1. The top row presents the application of the self-supervised pre-training objective of PEGASUS described proposed by Zhang et al. (2020). The downstream training task consists of applying EDA (Wei & Zou, 2019) to a corpus sub-sample of the CNN/DM dataset. Each text-summary pair is given a complexity score through our novel scoring system, sorted, and presented to PEGASUS according to a Training Scheduler (Baby-Steps (Spitkovsky et al., 2010) or One-Pass (Bengio et al., 2009)).

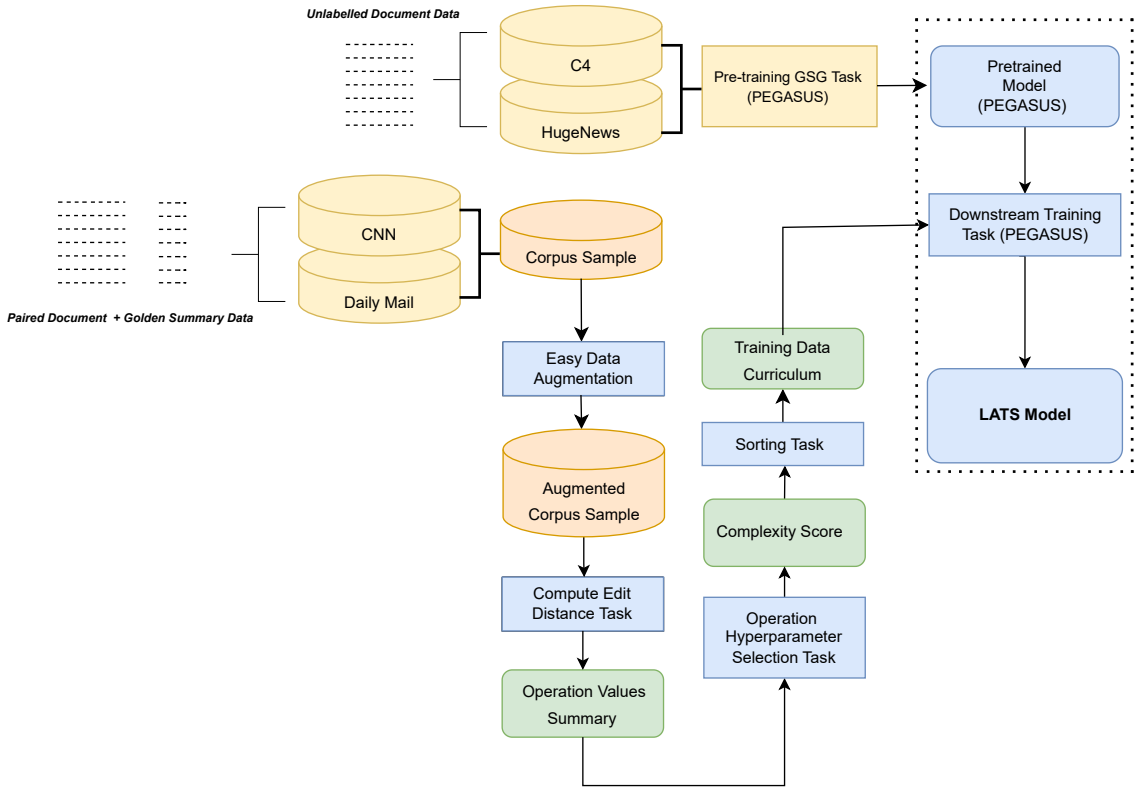

Figure 1: The complete workflow of the proposed LATS model.

4.1. 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. Determining a good pre-training task is more of an art than a science and is where the work of (Zhang et al., 2020) provided a significant contribution by developing a novel pre-training method (GSG).

GSG 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. (2020) showed that this task is similar enough to the main task, ATS, and 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. (2020) 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. This leads 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 with each approach on the pre-training dataset and fine-tuning the model on four different datasets. 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 builds further upon the model pre-trained with the principal gap-sentences approach with independently scored sentences while allowing multiple identical n-grams.

PEGASUS consists of two model sizes. In this research, we use the $\text{PEGASUS}_{\text{LARGE}}$ which consists of 568M parameters. We use the pre-trained model as calibrated by Zhang et al. (2020). For fine-tuning, we adjust the batch size and number of steps to accommodate our hardware constraints. The hyperparamer values are presented in Table 2.

4.2. Model Training

The following section outlines the model training approaches aimed at optimising the model's performance with the smallest amount of required training samples. Section 4.2.1 describes the CL concept and how it is applied in our research where we introduce a novel text-summary pair difficulty scoring system. Lastly, Section 4.2.2 describes the data augmentation technique applied in the study.

4.2.1. Curriculum Learning

Recent literature in the field of CL has dispersed over a variety of domains, each with differing levels of complexity. Furthermore, selecting the correct model Difficulty Measurer and Training Scheduler (Wang et al., 2020) must be carefully decided based on the task at hand. For our research pertaining to low-resource environments, we intend to focus on scalable and effective techniques with an emphasis on economical and

Pre-Training		Fine-Tuning	
Hyperparameter	Value	Hyperparameter	Value
Encoder- Decoder Layers	16	Encoder- Decoder Layers	16
Hidden Size	1024	Hidden Size	1024
Feed Forward Layer Size	4096	Feed Forward Layer	4096
Self-Attention Heads	16	Self-Attention Heads	16
Batch Size	8192	Batch Size	2
Learning Rate	1.00E-01	Learning Rate	5.00E-04
Pre-training Steps	500,000	Fine-Tuning Steps	50,000

Table 2: Comparison of pre-training and fine-tuning hyperparameters. The pre-trainig hyperperameters are that of the model of Zhang et al. (2020) which were decided by grid search. The fine-tuning parameters are taken from those of Zhang et al. (2020), with adjustments made to the batch size and step count.

efficacious use of data. For this reason, we consider two methods to dictate the sequence of training data subsets at a given epoch.

Firstly, a CL strategy that we 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 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 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 overfitting 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.

Spitkovsky et al. (2010) proposed another CL strategy, Baby-Steps, 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 CL 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 One-Pass.

For the same discussed motivations when selecting the CL scheduler we focus on a heuristics approach for measuring difficulty. Although utilizing sentence length is a common approach (Spitkovsky et al., 2010), it comes short of capturing in its entirety the various means of measuring syntactic complexity. We propose a novel difficulty measure based on the rewriting operations required to get from the original text to the

summary. To do this we take inspiration from the work of Cohn and Lapata (Cohn & Lapata, 2018). This research used Synchronous Tree Substitution Grammer (STSG) rules to build abstractive summaries through operations including substitution, reordering, insertion, and deletion for a sentence compression task. We build upon this to create a complexity score for the text-summary training pair for our proposed LATS model through a weighted sum of 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. Furthermore, the simplicity of this method is advantageous in that it requires no initialization or pre-training itself. 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 1-3.

$$C(s) = w_{wd} * \sum_{i} wd_i + w_{wr} * \sum_{j} wr_j + w_{ws} * \sum_{k} ws_k + w_{wa} * \sum_{l} wa_l \tag{1}$$

$$w_{wd}, w_{wr}, w_{ws}, w_{wa} \in [0, 1] \tag{2}$$

$$w_{wd} + w_{wr} + w_{ws} + w_{wa} = 1 \tag{3}$$

Algorithms 1, 2, 3, and 4 outline the methods used to derive values for wd, wa, wr, and ws, respectively. These functions are called after having cleaned both text and summary samples (removed stopwords, stripped for punctuation, and lowercasing). We define a deletion as the presence of a word in the text but not in the summary, and an addition as the inverse operation.

The weights of these text operations are optimized through a random search approach. We iterate though 10 randomly generated possible weight combinations within the constraints given by Equations 2 and 3. Using these weight combinations, 1,000 training samples 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 is evaluated on the validation set (10% of training sample) and the best checkpoint is used as the initialization point of the next bucket.

Algorithm 1 Deletion

t_i = Corpus Text Sample i
s_i = Corpus Summary i
procedure DeletionValue(t_i, s_i)
    $DelNum \leftarrow 0$
    $DelWords \leftarrow []$
    for $word$ in t_i **do**
        $count_t \leftarrow 0$
        $count_s \leftarrow 0$
        $count_t \leftarrow |\{y \mid y \in t_i \wedge y = word\}|$
        $count_s \leftarrow |\{y \mid y \in s_i \wedge y = word\}|$
        $count_d \leftarrow count_t - \min(count_t, count_s)$
        $DelNum \leftarrow DelNum + count_d$
        for d in range $count_d$ **do**
            $DelWords \cup \{word\}$
        end for
    end for
    $DelResult \leftarrow [DelNum, DelWords]$
    return $DelResult$
end procedure

Algorithm 2 Addition

t_i = Corpus Text Sample i
s_i = Corpus Summary i
procedure AdditionValue(t_i, s_i)
    $AddNum \leftarrow 0$
    $AddWords \leftarrow []$
    for $word$ in t_i **do**
        $count_t \leftarrow 0$
        $count_s \leftarrow 0$
        $count_t \leftarrow |\{y \mid y \in t_i \wedge y = word\}|$
        $count_s \leftarrow |\{y \mid y \in s_i \wedge y = word\}|$
        $count_a \leftarrow count_s - \min(count_s, count_t)$
        $AddNum \leftarrow AddNum + count_a$
        for a in range $count_a$ **do**
            $AddWords \cup \{word\}$
        end for
    end for
    $AddResult \leftarrow [AddNum, AddWords]$
    return $AddResult$
end procedure

Algorithm 3 Reorder

t_d = Corpus Text Sample i [Removed Deletions]
s_a = Corpus Summary i [Removed Additions]
procedure ReorderValue(t_i, s_i)
    $ThreeGram_s \leftarrow []$
    $ThreeGram_t \leftarrow []$
    $Reorder \leftarrow 0$
    for $i \in [0, length(s_i) - 2)$ **do**
        $ThreeGram_s := ThreeGram_s \cup$ Sublist of s_c from index$[i, i + 2]$
    end for
    for $i \in [0, length(t_i) - 2)$ **do**
        $ThreeGram_t := ThreeGram_t \cup$ Sublist of t_d from index$[i, i + 2]$
    end for
    for $threegram$ in $ThreeGram_s$ **do**
        $count_t \leftarrow 0$
        $count_s \leftarrow 0$
        $count_t \leftarrow |\{y \mid y \in ThreeGram_t \wedge y = threegram\}|$
        $count_s \leftarrow |\{y \mid y \in ThreeGram_s \wedge y = threegram\}|$
        $count_r \leftarrow count_s - \min(count_s, count_t)$
        $ReoNum \leftarrow ReoNum + count_r$
    end for
    return $ReoNum$
end procedure

Algorithm 4 Substitution

```
1: procedure SubstitutionValue(DelResult, AddResult)
2:     LemDeletions ← []
3:     LemAdditions ← []
4:     for word in DelResult[1] do
5:         LemDeletions := lemmatize(word)
6:     end for
7:     for word in AddResult[1] do
8:         LemAdditions := lemmatize(word)
9:     end for
10:    SubNum ← |LemDeletions ∩ LemAdditions|
11:    DelNum ← DelResult[0] − SubNum
12:    AddNum ← AddResult[0] − SubNum
13:    return SubNum, DelNum, AddNum
14: end procedure
```

The procedure defined in Algorithm 1 produces a count of word deletions and a list of deleted words. It calculates the number of deletions for each word by considering the excess occurrences in the text sample over the summary. The result is a list containing the total number of deletions and a list of words that were deleted along with their frequency. The algorithm is designed to identify and quantify the deletions made from the summary compared to the original text.

The procedure defined in Algorithm 2 produces a count of word additions and a list of added words. The algorithm computes the difference between the occurrences of each word in the summary and the sample text. It calculates the number of additions for each word by considering the excess occurrences in the summary. The algorithm is designed to identify and quantify the additions made in the summary compared to the original text.

The reorder procedure outlined in Algorithm 3 uses as input the text sample and summary less deleted and added words, respectively (so that both samples contain the same words). This procedure utilizes sublists of sets of 3 consecutive words (three-gram), and counts a reorder when a three-gram is in the text but not in the summary in that order. The output, $ReoNum$, represents the number of reordered three-grams in the summary compared to the text.

Substitution is defined as a word that is deleted from the text, but added to the summary in a alternate form. For example, if the text contains the word "having" and the summary contains the word "had", this is considered a substitution. The algorithm 4 first lemmatizes the words in the deleted and added sets to normalize them. Then, it calculates the number of substitutions by finding the intersection of lemmatized deletions and lemmatized additions. $DelNum, AddNum, SubNum$, and $ReoNum$ are used to instantiate wd, wa, ws, and wr respectively. Please note that the used values for $DelNum$ and $AddNum$ are those computed in Algorithm 4, and not the values from Algorithms 1 and 2.

The performance is measured based on a combined ROUGE F1 score as defined in Section 4.3. In order to

incorporate our hypothesis stated in Section 4.2.1, we add an 11th weight combination into the comparison which is in line with our hypothesis and has the following weights;

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

Lastly, two baseline measures will be included in our final results. Firstly, the length of the input text, measured in number of words, will be used as a proxy for complexity. Secondly, 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.

4.2.2. Data Augmentation

Another technique to improve model performance with small training datasets is data augmentation (Ramirez et al., 2019b) (Aftab & Siddiqui, 2018). Data augmentation is a technique to artificially expand the pool of training samples by altering the existing training samples in some way with minimal diversion from original meaning. Common applications in image based training samples include rotating, cropping, and mirroring the image. In our research we apply EDA for NLP proposed by (Wei & Zou, 2019), which 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 ATS task. The four techniques proposed by (Wei & Zou, 2019) are (1) Synonym Replacement (SR), (2) Random Insertion (RI), (3) Random Swap (RS), and (4) Random Deletion (RD). The details of the techniques are outlined in Table 3.

Table 3: 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.

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.

4.3. Evaluation methods

A measure 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 score, a recall score, 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 use mainly the F1 score of ROUGE-N and ROUGE-L measures, as these are commonly used in existing literature to evaluate abstractive summaries. The F1 score of these metrics will be denoted as $ROUGE - N_{\mathrm{F1}}$ and $ROUGE - L_{\mathrm{F1}}$, respectively. The ROUGE-N and ROUGE-L scores are further described in Sections 4.3.1 and 4.3.2.

4.3.1. 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, recall measures how many of the possible N-grams the model is able to generate and precision measures how many of the generated N-grams are in the reference summary. The $ROUGE - N_{\mathrm{F1}}$ measure combines these measure into a single value through a harmonic mean. The three measures are defined in Equations 4, 5 and 6.

$$\mathrm{F1}_{N-gram} = \frac{2}{\frac{1}{recall_{N-gram}} + \frac{1}{precision_{N-gram}}}, \tag{4}$$

where

$$recall_{N-gram} = \frac{\text{number of overlapping N-grams}}{\text{total N-grams in the reference summary}}, \tag{5}$$

and

$$precision_{N-gram} = \frac{\text{number of overlapping N-grams}}{\text{total N-grams in the model summary}}. \tag{6}$$

4.3.2. 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). 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 $ROUGE-L_{\text{F1}}$ score is the harmonic mean between these two measures. The three ROUGE-L measures are defined in Equations 7, 8 and 9.

$$\text{F1}_{LCS} = \frac{2}{\frac{1}{recall_{LCS}} + \frac{1}{precision_{LCS}}}, \tag{7}$$

where

$$recall_{LCS} = \frac{LCS(sentence_{reference}, sentence_{model})}{m}, \tag{8}$$

and

$$precision_{LCS} = \frac{LCS(sentence_{reference}, sentence_{model})}{n}. \tag{9}$$

4.3.3. Combined ROUGE

In this research we follow the measures used by (Nallapati et al., 2017) and (Zhang et al., 2020), which are the $ROUGE-1_{\text{F1}}$ (ROUGE-N with $N=1$), $ROUGE-2_{\text{F1}}$ (ROUGE-N with $N=2$) and $ROUGE-L_{\text{F1}}$ scores. However, it is possible that three separate scores can provide inconsistent conclusions between models. Therefore we combine these measures into a combined ROUGE score (see Equation 10) through a weighted average of these three scores, as was introduced in the code of Zhang et al. (2020) available at `https://github.com/google-research/pegasus`. The weights are 1 for the $ROUGE-1_{\text{F1}}$ and $ROUGE-L_{\text{F1}}$ scores, and 2 for the $ROUGE-2_{\text{F1}}$ score. The weights are chosen in favour of the $ROUGE-2_{\text{F1}}$ 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 $ROUGE-2_{\text{F1}}$ 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 $ROUGE-1_{\text{F1}}$ score is still an important measure to include in the combined ROUGE score. Furthermore, the $ROUGE-L_{\text{F1}}$ score is beneficial to include in the combined $ROUGE_{\text{F1}}$ score as it captures the value of the longer correct sequences that a model can generate.

$$\text{Combined ROUGE F1 score} = ROUGE-1_{F1} + 2 * ROUGE-2_{F1} + ROUGE-L_{F1} \quad (10)$$

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).

5. Results

The following section outlines the results of our research. Firstly, 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 $ROUGE_{\text{F1}}$ scores through the various strategies and techniques applied to the datasets.

5.1. Complexity Scoring

Table 4 shows that the best overall results were achieved by combination number 4 (in bold) which was 1.7% above the overall average. This is likely explained by the fact that the word additions task was far less common than the word deletion and word reordering tasks, as shown in Table 5. Thus, although the task is likely still more complex for the model to learn, it is not so important to learn because even if the model performs poorly in that task the generated summary can still be of high quality. Comparing the results in Table 4, the word deletion task is not assigned the highest weight, despite being the most common task, which contradicts our hypothesis that frequency is an important parameter for weight selection. It is possible that deletion is regarded by the model as a less complex task than reordering or substitution. Therefore, it is not informative for the CL algorithm in forming an effective complexity ranking. Combinations 2, 7, and 8, which have high weights for word deletion, show average or below-average results. The word substitution task has the lowest occurrence and standard deviation. However, it is assigned a relatively high weight in the well-scoring models (combinations 1, 3 and 4). The quality of the complexity scoring depends on the relative distribution of weights among the tasks. Optimal results are achieved when similar weights are assigned to word deletion and word addition, and a weight approximately four times higher is given to word replacement. The word addition task's weight also has a significant impact, as shown by the difference between combinations 4 and 5.

The astute reader might note that, when rounding off the weights of each text operation to two decimals, we are left with 126,851 possible weight combinations. This is a well-known discrete mathematics problem commonly solved by a "stars-and-bars" approach (see Equation 11). 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 11. 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. The bars can be placed in 103 positions (100 stars + 3 bars), considering they may also be placed directly after each other, and of these we select 3 positions. The order in which these positions are selected does not matter, as the arrangement of variable weights is automatically encoded in the bar positions regardless of selection order. This is equal to the number of possible combinations of length 3 (3 bars) out of a set of 103 elements, namely $126,851$ (see Equation 12).

Figure 2 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 continue our research with the weights of combination four for both CL strategies as it gave the highest overall combined $ROUGE_{\text{F1}}$ score.

$$\begin{aligned} w_{wd} + w_{wr} + w_{ws} + w_{wa} &= 1 \\ 0.97 + 0.01 + 0.01 + 0.01 &= 1 \\ \underbrace{\star\star\star\ldots\star\star\star}_{97} | \star | \star | \star &= 1, \text{ with } \star = 0.01 \end{aligned} \tag{11}$$

$$\text{Number of weight sets} = C(103, 3) = \frac{103!}{(103-3)! * 3!} = \frac{103 * 102 * 101}{6} = 126,851 \tag{12}$$

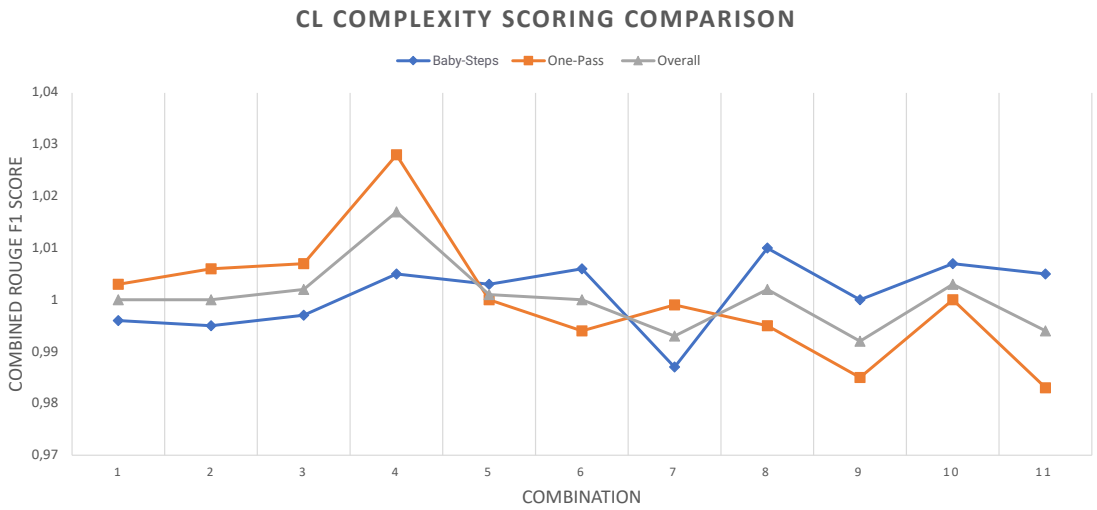

Figure 2: This figure shows the relative combined ROUGE F1 scores for the complexity scoring hyperparameter comparison for all the weight combinations as described in Table 4.

Table 4: 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 $ROUGE-1_{\text{F1}}$, $ROUGE-2_{\text{F1}}$, and $ROUGE-L_{\text{F1}}$ 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 $ROUGE-1_{\text{F1}}$, 2 times the $ROUGE-2_{\text{F1}}$ score, and the $ROUGE-L_{\text{F1}}$ 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	0.95	0.996	**1.124**	0.984	0.85	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

Table 5: The average occurrence, with the respective standard deviation in brackets, of each task in the 1000 samples used for the complexity scoring strategy hyperparameter optimisation.

	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)

5.2. Summarization Results

Table 6 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. The percentage improvements with respect to this baseline are shown in square brackets below the score.

The CL strategies have been executed as described in Section 4.2.1 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 is common practice when using the CNN/DM dataset (Wang et al., 2021b) consisting of 11,490 samples.

Let us first consider the results of the One-Pass CL strategy. In the dataset consisting of 10 samples, the various sorting methods did not result in any major differences with a 0.14% decrease in performance for a CL strategy without sorting and a 0.29% increase for all other strategies compared to the baseline performance of no CL. 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 6 shows that the performance of the CL strategy without any sorting method is slightly worse than the baseline (No CL) performance (-2%). 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 model's solution space and thus have 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. However, applying EDA to the dataset in combination with Baby-Steps boosted the performance by 4.3% in comparison to the no CL strategy using Baby-Steps. The final sample size we considered in our research consists of 1000 samples as shows in Table 6. 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 5.66% compared to the baseline performance of no CL strategy using Baby-Steps. Extending the CL strategy with the EDA techniques increased the performance further to a total performance increase of 6.54% compared to the the no CL baseline.

Table 6: 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. Percentage improvements from the baseline (first row) are shown in square brackets.

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
			[0.03%]	[-0.32%]	[-0.19%]	[-0.14%]	[-0.77%]	[-2.07%]	[-3.73%]	[-2.00%]	[0.12%]	[2.16%]	[0.91%]	[1.00%]
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
			[0.25%]	[0.49%]	[0.14%]	[0.29%]	[-1.11%]	[-2.70%]	[-3.91%]	[-2.39%]	[0.36%]	[0.07%]	[-0.26%]	[0.11%]
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
			[0.25%]	[0.49%]	[0.14%]	[0.29%]	[-1.11%]	[-2.70%]	[-3.91%]	[-2.39%]	[1.57%]	[2.30%]	[0.56%]	[1.54%]
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
			[0.25%]	[0.49%]	[0.14%]	[0.29%]	[-1.20%]	[-2.86%]	[-3.59%]	[-2.39%]	[3.31%]	[5.95%]	[2.63%]	[3.99%]
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
			[0.31%]	[0.41%]	[0.33%]	[0.35%]	[0.03%]	[0.72%]	[-0.90%]	[-0.01%]	[4.61%]	[9.44%]	[5.57%]	[6.42%]
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
			[-0.12%]	[-0.49%]	[0.28%]	[-0.13%]	[0.25%]	[0.16%]	[-0.49%]	[0.01%]	[4.34%]	[2.16%]	[-0.56%]	[2.25%]
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
			[-0.25%]	[-0.89%]	[0.28%]	[-0.31%]	[-0.06%]	[-0.32%]	[-0.85%]	[-0.36%]	[2.53%]	[-0.22%]	[0.73%]	[1.26%]
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
			[-0.25%]	[-0.89%]	[0.28%]	[-0.31%]	[-0.86%]	[-0.32%]	[-0.85%]	[-0.36%]	[6.93%]	[5.28%]	[3.93%]	[2.64%]
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
			[-0.25%]	[-0.89%]	[0.28%]	[-0.31%]	[0.06%]	[-0.32%]	[-1.89%]	[-1.38%]	[1.74%]	[5.58%]	[3.28%]	[5.66%]
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
			[1.96%]	[3.73%]	[2.89%]	[2.77%]	[3.02%]	[6.44%]	[3.73%]	[4.30%]	[10.09%]	[3.12%]	[5.44%]	[6.54%]

The Baby-Steps strategy improves 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, 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 0.06% and 2.38%, respectively. When considering the same effects for the Baby-Steps algorithm, we see performance improvements of 3.08% and 5.68% for these two datasets. This again indicates the increased exposure the model has to the simpler examples 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 (-0.99%) and reduction (+0.54%) 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 2.99% compared to the One-Pass algorithm without any sorting. Thus, our complexity sorting algorithm demonstrates 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 -0.99% and 0.39%, respectively, compared to the CLBS strategy with no sorting strategy. However, the complexity sorting algorithm does result in a performance improvement of 3.41% compared to the baseline without a sorting strategy.

Combining the individual effects described in the previous paragraphs allows us to compare the final performance to the initial baseline performance of no CL strategy. The combination of methods that achieves the best performance is the CLBS strategy with a complexity sorting algorithm and with EDA data augmentation techniques applied, which resulted in a performance gain of 6.54% compared to the baseline of no curriculum learning, from a combined ROUGE F1-score of 83.28 to 88.73. The performance gain for the CLOP strategy with the complexity sorting algorithm and EDA applied is very close with 6.42%.

6. Concluding Remarks

In this final section 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. 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 in low-resource environments by combining these models with selected methods to optimize efficiency. This was done by applying CL strategies in combination with data augmentation. To do so, we introduced a novel text-summary pair complexity scoring algorithm and have shown that our extensions help improve results.

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 simple and efficient applications of CL 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

As for future research, 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 limitation of this work is the focus on a granular analysis, as the data-augmentation and complexity scoring both happen at the word-level. Although our model showed promising results in improving ATS in low-resource environments, future research may focus on making summaries more informative by accounting for summary sentiment. By employing sentiment mining solutions in addition to the proposed architecture, such as those in the work of Srinivasarao & Sharaff (2024) and Srinivasarao & Sharaff (2023) it could be possible to assess and penalize semantic mismatch and generate even more faithful and coherent results. Furthermore, it may be interesting to explore more sophisticated data augmentation techniques by augmenting at the phrase level while maintaining class-labels to come up with a richer and more diverse set of synthetic data.

Furthermore, 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.

Bibliography

Aftab, U., & Siddiqui, G. F. (2018). Big data augmentation with data warehouse: A survey. In *3rd IEEE International Conference on Big Data (ICBDA 2018)* (pp. 2785–2794). IEEE.

Bahdanau, D., Cho, K., & 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., & Witbrock, M. J. (2000). Headline generation based on statistical translation. In *38th Annual Meeting of the Association for Computational Linguistics (ACL 2000)* (pp. 318–325). ACL.

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. In *26th Annual International Conference on Machine Learning (ICML 2009)* (pp. 41–48). ACM.

Chopra, S., Auli, M., & Rush, A. M. (2016). Abstractive sentence summarization with attentive recurrent neural networks. In *15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)* (pp. 93–98).

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

Cohn, T., & Lapata, M. (2018). Sentence compression beyond word deletion. In *22nd International Conference on Computational Linguistics (COLING 2008)* (pp. 137–144). ACL.

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

Eisner, J. (2003). Learning non-isomorphic tree mappings for machine translation. In *41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)* (pp. 205–208). ACL.

El-Kassas, W. S., Salama, C. R., Rafea, A. A., & Mohamed, H. K. (2021). Automatic text summarization: A comprehensive survey. *Expert Systems with Applications*, *165*, 113679.

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

Fang, M., Zhou, T., Du, Y., Han, L., & Zhang, Z. (2019). Curriculum-guided hindsight experience replay. In *32nd Annual Conference on Neural Information Processing Systems (NIPS 2019)* (pp. 12602–12613). Curran Associates.

Gambhir, M., & Gupta, V. (2017). Recent automatic text summarization techniques: A survey. *Artificial Intelligence Review*, *47*, 1–66.

Hacohen, G., & Weinshall, D. (2019). On the power of curriculum learning in training deep networks. In *36th International Conference on Machine Learning (ICML 2019)* (pp. 2535–2544). PMLR.

He, Z., Yang, M., Feng, M., Yin, J., Wang, X., Leng, J., & Lin, Z. (2023). Fourier transformer: Fast long range modeling by removing sequence redundancy with FFT operator. In *Findings of the Association for Computational Linguistics: ACL 2023* (pp. 8954–8966). ACL.

Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., & Blunsom, P. (2015). Teaching machines to read and comprehend. In *29th Annual Conference on Neural Information Processing Systems (NIPS 2015)* (pp. 1693–1701). MIT Press.

Hoang, V. C. D., Koehn, P., Haffari, G., & Cohn, T. (2018). Iterative back-translation for neural machine translation. In *2nd Workshop on Neural Machine Translation and Generation (WNMT 2018)* (pp. 18–24). ACL.

Jean, S., Cho, K., Memisevic, R., & 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 (IJCNLP 2015)* (pp. 1–10). ACL.

Kesgin, H. T., & Amasyali, M. F. (2024). Advancing NLP models with strategic text augmentation: A comprehensive study of augmentation methods and curriculum strategies. *Natural Language Processing Journal*, *7*, 100071.

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

Kim, H.-Y., Roh, Y.-H., & Kim, Y.-K. (2019). Data augmentation by data noising for open-vocabulary slots in spoken language understanding. In *17th Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2019)* (pp. 97–102). ACL.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A. et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, *114*, 3521–3526.

Kumar, M., Packer, B., & Koller, D. (2010). Self-paced learning for latent variable models. In *24th Annual Conference on Neural Information Processing Systems (NIPS 2010)* (pp. 1189–1197). Curran Associates.

Kurniawan, K., & Louvan, S. (2018). Indosum: A new benchmark dataset for indonesian text summarization. In *9th International Conference on Asian Language Processing (IALP 2018)* (pp. 215–220). IEEE.

Kwon, N., Yoo, Y., & Lee, B. (2024). Novel curriculum learning strategy using class based TF-IDF for enhancing personality detection in text. *IEEE Access*, *12*, 87873–87882.

Lesort, T. (2020). Continual learning: Tackling catastrophic forgetting in deep neural networks with replay processes. *arXiv preprint arXiv:2007.00487*, .

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2020). BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)* (pp. 7871–7880). ACL.

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *2004 Workshop on Text Summarization Branches Out (WAS 2004)* (pp. 74–81). ACL.

Liu, Y., & Liu, P. (2021). SimCLS: A simple framework for contrastive learning of abstractive summarization. In *In 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2021)* (pp. 1065–1072). ACL.

Liu, Y., Liu, P., Radev, D. R., & Neubig, G. (2022). BRIO: bringing order to abstractive summarization. In *60th Annual Meeting of the Association for Computational Linguistics (ACL 2022)* (pp. 2890–2903). ACL.

MacAvaney, S., Nardini, F. M., Perego, R., Tonellotto, N., Goharian, N., & Frieder, O. (2020). Training curricula for open domain answer re-ranking. In *43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2020)* (pp. 529–538). ACM.

Milani, S., Topin, N., Veloso, M., & Fang, F. (2024). Explainable reinforcement learning: A survey and comparative review. *ACM Computing Survey*, *56*, 168:1–168:36.

Moratanch, N., & Chitrakala, S. (2016). A survey on abstractive text summarization. In *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT 2016)* (pp. 1–7). IEEE.

Nallapati, R., Zhai, F., & Zhou, B. (2017). SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. In *31st AAAI Conference on Artificial Intelligence (AAAI 2017)* 1.

Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M. E., & Stone, P. (2020). Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, *21*, 7382–7431.

Portelas, R., Colas, C., Hofmann, K., & Oudeyer, P.-Y. (2020). Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments. In *4th Conference on Robot Learning (CoRL 2020)* (pp. 835–853). PMLR.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I. et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, *1*, 9.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, *21*, 5485–5551.

Ramirez, J. M., Montalvo, A., & Calvo, J. R. (2019a). A survey of the effects of data augmentation for automatic speech recognition systems. In *24th Iberoamerican Congress on Pattern Recognition (CIARP 2019)* (pp. 669–678). Springer.

Ramirez, J. M., Montalvo, A. R., & Calvo, J. R. (2019b). A survey of the effects of data augmentation for automatic speech recognition systems. In *24th Iberoamerican Congress: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications (CIARP 2019)* (pp. 669–678). Springer.

Ranaldi, L., Pucci, G., & Zanzotto, F. M. (2023). Modeling easiness for training transformers with curriculum learning. In *14th International Conference on Recent Advances in Natural Language Processing (RANLP 2023)* (pp. 937–948). INCOMA.

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

Soviany, P., Ionescu, R. T., Rota, P., & Sebe, N. (2022). Curriculum learning: A survey. *International Journal of Computer Vision*, *130*, 1526–1565.

Spitkovsky, V. I., Alshawi, H., & Jurafsky, D. (2010). From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *2022 Conference of the North American Chapter of the Association of Computational Linguistics Human Language Technologies (NAACL-HLT 2022)* (pp. 751–759). ACL.

Srinivasarao, U., & Sharaff, A. (2023). Machine intelligence based hybrid classifier for spam detection and sentiment analysis of SMS messages. *Multimedia Tools and Applications*, *82*, 31069–31099.

Srinivasarao, U., & Sharaff, A. (2024). Sentiment analysis from email pattern using feature selection algorithm. *Expert Systems*, *41*.

Srivastava, R., Singh, P., Rana, K., & Kumar, V. (2022). A topic modeled unsupervised approach to single document extractive text summarization. *Knowledge-Based Systems*, *246*, 108636.

Su, Z., Zhang, Z., Xu, G., Liu, J., Han, X., Zhang, T., & Dong, Y. (2025). Multilingual encoder knows more than you realize: shared weights pretraining for extremely low-resource languages. *arXiv preprint arXiv:2502.10852*, .

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *27th Annual Conference on Neural Information Processing Systems (NIPS 2014)* (pp. 3014–3112). Curran Associates.

Tang, Y., Wang, Y., Guo, J., Tu, Z., Han, K., Hu, H., & Tao, D. (2024). A survey on transformer compression. *arXiv preprint arXiv:2402.05964*, .

Tsvetkov, Y., Faruqui, M., Ling, W., MacWhinney, B., & Dyer, C. (2016). Learning the curriculum with bayesian optimization for task-specific word representation learning. In *54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)* (pp. 130–139). ACL.

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

Wang, F., Song, K., Zhang, H., Jin, L., Cho, S., Yao, W., Wang, X., Chen, M., & Yu, D. (2022). Salience allocation as guidance for abstractive summarization. In *2022 Conference on Empirical Methods in Natural Language Processing (EMNLP 2022)* (pp. 6094–6106). ACL.

Wang, X., Chen, Y., & Zhu, W. (2021a). A survey on curriculum learning. *Transactions on Pattern Analysis and Machine Intelligence*, *44*, 4555–4576.

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

Wang, Y., Chen, Q., Shen, J., Hou, B., Ahmed, M., & Li, Z. (2021b). Aspect-level sentiment analysis based on gradual machine learning. *Knowledge-Based Systems*, *212*, 106509.

Wei, J., & 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 (EMNLP 2019) and the 9th International Joint Conference on Natural Language Processing (IJCNLP 2019)* (pp. 6382–6388). ACL.

Woodsend, K., Feng, Y., & Lapata, M. (2010). Generation with quasi-synchronous grammar. In *Conference on empirical methods in natural language processing (EMNLP 2010)* (pp. 513–523). ACL.

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

Zhang, J., Zhao, Y., Saleh, M., & Liu, P. (2020). PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In *37th International Conference on Machine Learning (ICML 2020)* (pp. 11328–11339). PMLR.

Zhang, Z., Liu, S., Li, M., Zhou, M., & Chen, E. (2018). Joint training for neural machine translation models with monolingual data. In *32nd AAAI Conference on Artificial Intelligence (AAAI 2018)* (pp. 555–562). AAAI.

Zhao, J., Sun, X., & Feng, C. (2025). Introducing bidirectional attention for autoregressive models in abstractive summarization. *Information Sciences*, *689*, 121497.

Zhao, Y., Khalman, M., Joshi, R., Narayan, S., Saleh, M., & Liu, P. J. (2023). Calibrating sequence likelihood improves conditional language generation. In *11th International Conference on Learning Representations (ICLR 2023)*. OpenReview.net.

Zhou, Y., Pan, Z., Wang, X., Chen, H., Li, H., Huang, Y., Xiong, Z., Xiong, F., Xu, P., Liu, S., & Zhu, W. (2024). Curbench: curriculum learning benchmark. In *41st International Conference on Machine Learning, (ICML 2024)* (pp. 62088–62107). PMLR.