H.ABBAS

ATILIM UNIVERSITY 2021

ABSTRACTIVE TEXT SUMMARIZATION USING DEEP LEARNING

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ATILIM UNIVERSITY

HANAN WAHHAB ABBAS ABBAS

A MASTER OF SCIENCE THESIS
IN
THE DEPARTMENT OF COMPUTER ENGINEERING

SEPTEMBER 2021

ABSTRACTIVE TEXT SUMMARIZATION USING DEEP LEARNING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ATILIM UNIVERSITY

BY

HANAN WAHHAB ABBAS ABBAS

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2021

Approval of the Graduate School of Natural and Applied Sciences, Atilim University.

Prof. Dr. Ender KESKİNKILIÇ
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of **Master of Science in Computer Engineering Department, Atilim University**.

Assoc. Prof. Dr. Gökhan ŞENGÜL
Head of Department

This is to certify that we have read the thesis **"ABSTRACTIVE TEXT SUMMARIZATION USING DEEP LEARNING"** submitted by **Hanan Wahhab Abbas Abbas** and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Beytullah YILDIZ
Supervisor

Examining Committee Members:

Assoc. Prof. Dr. Nergiz ÇAĞILTAY
Software Engineering, Atilim University

Asst. Prof. Dr. Beytullah YILDIZ
Software Engineering, Atilim University

Asst. Prof. Dr. Gürhan GÜNDÜZ
Computer Engineering,
Mugla Sıtkı Kocman University

Date: September 16, 2021

I declare and guarantee that all data, knowledge and information in this document has been obtained, processed and presented in accordance with academic rules and ethical conduct. Based on these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Hanan Wahhab Abbas Abbas

ABSTRACT

ABSTRACTIVE TEXT SUMMARIZATION USING DEEP LEARNING

Abbas, Hanan Wahhab Abbas

M.S., Department of Computer Engineering

Supervisor : Asst. Prof. Dr. Beytullah YILDIZ

September 2021, 50 pages

The ability to produce summaries automatically helps to improve knowledge dissemination and retention, as well as efficiency in a variety of fields. There are basically two approaches to summarizing, abstractive and extractive. The abstractive approach is considered more successful as it is the process of creating a brief summary of the source text to capture the main ideas. In this approach, summaries created from the source text may contain new phrases and sentences not included in the original text. The use of attention-based Recurrent Neural Networks encoder-decoder models has been popular for a variety of language-related tasks, including summarization and machine translation. Recently, in the field of machine translation, the Transformer model has proven to be superior to the Recurrent Neural Networks-based model. In this thesis, we propose an improved encoder-decoder Transformer model for text summarization. As a baseline model, we used Long Short-Term Memory with attention, a Recurrent Neural Networks model, for the abstractive text summarization task. Evaluation of this study is performed automatically using the ROUGE score. Experimental results show that the Transformer model provides a better summary and a higher ROUGE score.

Keywords: Abstractive text summarization, Transforme, Attention, Recurrent neural

network, Sequence-to-sequence

ÖZ

SOYUTLAYICI METIN ÖZETLEMESI DERIN ÖĞRENME KULLANARAK

Abbas, Hanan Wahhab Abbas

Yüksek Lisans, Bilgisayar Mühendisliği

Tez Yöneticisi : Dr. Öğr. Üyesi Beytullah YILDIZ

Eylül 2021 , 50 sayfa

Özetleri otomatik olarak üretme yeteneği, çeşitli alanlarda verimliliğin yanı sıra bilginin yayılmasını ve elde tutulmasını iyileştirmeye yardımcı olabilir. Özetleme, soyutlamacı ve çıkarıcı olmak üzere temelde iki yaklaşım vardır. Ana fikirleri yakalamak için kaynak metnin kısa bir özetini oluşturma süreci olduğu için soyutlayıcı yaklaşım daha başarılı kabul edilir. Bu yaklaşımda, kaynak metinden oluşturulan özetler, orijinal metinde yer almayan yeni ifadeler ve cümleler içerebilir. Dikkate dayalı Tekrarlayan Sinir Ağları kodlayıcı-kod çözücü modellerinin kullanımı, özetleme ve makine çevirisi dahil olmak üzere dille ilgili çeşitli görevler için popüler olmuştur. Son zamanlarda, makine çevirisi alanında, Transformer modelinin Tekrarlayan Sinir Ağları tabanlı modelden üstün olduğu kanıtlanmıştır. Bu tezde, metin özetleme için geliştirilmiş bir kodlayıcı-kod çözücü Transformer modeli öneriyoruz. Temel model olarak, soyutlayıcı metin özetleme görevi için bir Tekrarlayan Sinir Ağları modelini olan Dikkatli Uzun Kısa Süreli Bellek kullandık. Bu çalışmanın değerlendirilmesi, ROUGE puanı kullanılarak otomatik olarak yapılmıştır. Deneysel sonuçlar, Transformer modelinin daha iyi bir özet ve daha yüksek bir ROUGE puanı sağladığını göstermektedir.

Anahtar Kelimeler: Soyutlayıcı metin özetleme, Dönüştürücü, Dikkat, Tekrarlayan

sinir ağı, Diziden diziye

To my family

ACKNOWLEDGMENTS

I would like to thank my professors for providing me with invaluable feedback and pointers to help move the research forward.

I want to express my gratitude to my parents, brothers, and sisters for believing in me and assisting me throughout my graduate studies.

Finally, a special thank to my dear brother Husham for constantly motivating me to achieve my goals.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

NLP	:	Natural Language Processing
LSA	:	Latent Semantic Analysis
TF	:	TensorFlow
RNN	:	Recurrent neural network
GloVe	:	Global vectors for word representation
OOV	:	Operation Objective Voice
Seq2seq	:	Sequence to Sequence
GPU	:	Graphics Processing Units
LSTM	:	Long Short-Term Memory
BLEU	:	Bilingual evaluation understudy
ROUGE	:	Recall-Oriented Understudy for Gisting Evaluation
METEOR	:	Metric for Evaluation of Translation with Explicit ORdering
TPU	:	Tensor Processing Unit
HMNet	:	Hierarchical Meeting summarization Network
DUC	:	Document Understanding Conferences

CHAPTER 1

INTRODUCTION

1.1 Overview

Without doubt, the amount of knowledge available on the internet and in other forms of written text has never been higher. Consequently, many people struggle with how to obtain valuable and meaningful knowledge from the vast amounts of data available. It is important that the information and data is easily searchable, and also presented in such a manner that the user can quickly determine whether or not the information is relevant. Automated text summarization is one technique that could theoretically assist in achieving this goal. Text summarization extracts only the most important information from a document or set of documents, enabling the reader to absorb the text's main message more quickly. Abstracts in research journals, headlines in journal articles, and legal briefs are all examples of common uses. Humans typically summarize by reading text and paraphrasing paragraphs and sentences to condense the details. This is a time-consuming task that could be automated; literature in this field dates all the way back to the 1950s [1], [2]. Automatic text summarization has two main categories, extractive and abstractive. Extractive summarization methods create summaries by taking a subset of the sentences in the original input document. These summaries include the most relevant sentences. In contrast, abstractive techniques produce sentences not seen in the source texts. Abstractive methods can produce summaries that are more efficient and difficult compared to extractive methods [3]. Additionally, abstractive methods can provide insight into our understanding of language. When abstractive summarization is used, the algorithm must ensure that the generated sentences are intelligible and grammatically correct, which requires the algorithm to extract language knowledge during training. For summarization, there is

the difficult task of determining which detail to include and which to exclude. Traditionally, approaches to abstractive summarization have relied heavily on heuristics based on the linguistic and syntactic experience of the language [4]. Encoder-decoder models, usually composed of two Recurrent Neural Networks (RNN), have recently demonstrated their ability to interpret and generate sequential data such as text. One encoder reads the input and encodes it into a latent vector, while the decoder reads the data represented in the vector and generates a new sequence of text [5]. Due to the success of encoder-decoder models for text generation tasks, a lot of research has been conducted on various configurations of this architecture [6]. For tasks involving language generation, such as machine translation and abstractive text summarization, one method has been to combine encoder-decoder models with an attention mechanism [3], [5]. The attention mechanism was introduced to aid in comprehension and to facilitate the generation of longer text sequences. These models, which incorporate both attention and the encoder-decoder architecture, are commonly referred to as sequence2sequence (seq2seq) models and have contributed significantly to the increased interest in abstractive summarization [7], [8], [9]. Recently, there has been an attempt to improve the RNN-based encoder-decoder model in the area of machine translation by focusing more on the attention mechanism. Vaswani et al. [10] developed the Transformer model, which eliminated the need for the RNN component by integrating the attention mechanism and feed-forward layers. They were able to achieve state-of-the-art results by doing so, significantly improving translation efficiency, especially for longer text sequences. Additionally, the Transformer model was found to train more quickly than RNN-based seq2seq models, in part because it allows for more parallel operations during training. The Transformer model was initially developed and tested for machine translation tasks, where it outperformed state-of-the-art solutions by reducing training time and improving translation efficiency [10]. Given the similarities between machine translation and automatic summarization, it was reasonable to expect the Transformer model to perform well on the abstractive summarization task as well.

Being able to produce well-written summaries that accurately return relevant information has the potential to boost productivity in a wide variety of fields, including those that rely heavily on written information, such as medicine or law. A frequently used strategy has been to use seq2seq models with attention for this purpose. Recently, many models, including Transformer, outperformed RNN-based seq2seq models for machine translation and text summarization, primarily because it can perform more operations in parallel and is better at handling long sequences. Comparing the Transformer model to the RNN-based seq2seq model on the abstractive summarization task will advance our understanding of how removing recurrence affects summary quality in neural networks. The Transformer model does not have a recurrence, which is the main difference compared with RNNbased seq2seq models. Since Transformer is a newer model than recurrent models, there are still questions to be answered, such as how removing recurrence affects learning and success in various contexts. As the evaluation process, we use ROUGE metric because the majority of prior work evaluating seq2seq models on summarization tasks relying on this automated metrics [11], [8], [3], [4], [6], [12]. The aim of this study is to compare the Transformer model's output on the abstractive text summarization task to an RNNbased seq2seq baseline model using the ROUGE evaluation method. This knowledge will help with model selection for text summarization tasks, and also provide insights into the readability of the output from encoder-decoder summarization models. Our contribution to this study is as follows:

1. Proposing an improved Transformer model for abstractive text summarization.
2. Experimenting on two different datasets having different document lengths. One has small and the other has very large document size.
3. Using subword tokenization strategy that helps us solve full length problem by breaking unknown words into "subword units".
4. Proving that Transformer based model outperforms the LSTM based model.

This thesis consists of five chapters. We provide an overview and describe the research's central problem in the introduction chapter. Following the introduction, the background and related work chapter discusses text summarization, evaluation methodologies, and some previous models used for abstractive text summarization tasks. Following that, in the methodology chapter, we examine our dataset and provide critical information about it, as well as explain our approach, model architecture, and input-output relationships. In the experiments and results chapter, we give our results and sample predictions. Finally, we present a conclusion and discuss potential future work in the conclusion and future work chapter

CHAPTER 2

RELATED WORK

2.1 Models Used for Abstractive Text Summarization

This section discusses recent abstractive text summarization models. It begins with Artificial intelligent Facebook researchers [3] who proposed a neural-attention model for paragraph summarization. They combined this new model with the generation algorithm to make the system produce summaries with high accuracy. Subsequently, in another paper [4], they improved their model with a conditional Recurrent Neural Networks architecture. They conducted experiments to optimize the decoder's performance using a convolutional attention-based technique.

Another study [13] examined abstractive summarization of Chinese language text by building a dataset called LCSTS using a microblogging website (Large-scale Chinese Short Text Summarization). This dataset contains a big corpus of brief Chinese writings annotated with concise descriptions. This corpus was then employed to develop an RNN-based model for summary generation. The researchers recommend that future study focus on the use of RNNs for summary generation and on difficulties involving rare words.

Nallapati et al. [8] proposed an abstractive text summarization model that incorporates attention and is based on neural encoder-decoders. As input, word/phrase embeddings are used. They make use of long short-term memory (LSTMs) and bidirectional RNNs. Their work builds on the structure established in [24], but also presents unique models that solve fundamental challenges in abstractive summarization.

Zhang [14] introduced a novel evaluation metric aims to solve the factual inconsistencies of abstractive summarization tasks and achieve a high ROUGE score. The factual score calculation system contains three different types of modules: fact scorer, fact encoder, and fact extractor. The OpenIE(open information extraction) methods are extracted the facts from the reference summary and candidate summary and converted them into embeddings using the fact encoder.

Gu and Lu [15] proposed the COPYNET model for sequence learning problems, which included a copying function in a neural encoder-decoder model. Replication is required for non-parsing elements of the input sequence, such as nouns and numerical details. This mechanism locates an appropriate segment from the input sequence and put it into the output sequence.

Chen et al. [16] proposed a work that was distinct from previous research in that, additionally to utilizing the attention method to maintain focus, it included a distraction mechanism that enabled traversing different sections of the document in order to gain a better understanding of the overall context and produce a more accurate summary. The researchers carried out encoding and decoding activities using a bidirectional gated recurrent architecture.

See et al. [17] created a new architecture that improves the seq2seq attention model's effectiveness in two novel ways. They began by using a hybrid pointer generator, which is capable of copying data from the source document using the pointing technique while also generating new phrases using a hybrid generator. To prevent needless repetition, the second approach they used the covering technique in keeping track of what had previously been summarized.

Chen and Bansal [18] developed an abstractive text summarization task-specific model called a hybrid extractive-abstractive architecture with reinforcement learning based on policy. The new features in the model allow it to be aware of the sentence's word hierarchy. On all versions of the CNN/Daily Mail news dataset, this model achieved the novel state of the art with significant improvements in both testing and training speed.

Duan et al. [19] proposed a novel attention technique for abstractive summarization

called the constructive mechanism. It is distinct from the more conventional methodology of paying attention. By introducing two constructive attention mechanisms, the new attention mechanism increases the model's attentional ability. The first type of attention is focused on the most significant parts of the sentence, while the second type is directed toward the less significant terms in the sentence. The softmin and softmax features are used to train both attentions in opposite directions. As a result, attention is focused more effectively on the relevant portions of the input, introduces the most reliable performance.

Zaki, Khalil, and Abbas [20] proposed many deep learning building blocks for abstractive text summarization in the Amharic language, including seq2seq models using LSTM with attention, a planned sampling strategy, and a pointer genera-tor network (African languages). Due to the lack of a publicly available dataset for African languages, the researcher created one from scratch. They conducted the training using the Google Colab framework and evaluated the results using the BLEU and ROUGE methods.

Masum et al. [21] suggested a strategy for abstractive summarization using the Amazon fine food review dataset. To begin, the dataset is cleaned and modified to make it suitable for the deep learning model/algorithm. This method begins with text splitting, contractions, stop word removal, and lemmatization and progresses to vocabulary size counting, word embedding, and special tokens such as UNK and EOS. Seq2seq is constructed utilizing a bidirectional LSTM encoder and a conventional LSTM decoder. The study's limitation, as highlighted by the researchers, is that only minimal input text can be adequately described. When massive amounts of text are used as data, the model becomes inconclusive.

Zhu et al. [22] proposed HMNet, an abstractive document summarization technique based on deep learning methodologies for autonomously generated meeting transcripts (Hierarchical Meeting summarization Network). To generate abstractive summaries of meeting transcripts, the HMNet model employs a sequence-to-sequence transformer structure. The researchers claim that the Hierarchical Meeting summarizing differs from traditional document summary. Due to the large number of participants at a meeting, the character of the transcripts varies significantly due to the

variety of grammar styles, opinions, and member positions. Finally, the model was tested on ICSI and AMI meeting corpora using three ROUGE technique variations: Rouge 1, Rouge 2, and Rouge SU4.

Saito et al. [23] suggested a methodology for abstractive document summarizing called Length controllable Prototype-guided Abstractive Summarization that limits the length of the output summary (LPAS). This model's purpose is to combine the use of abstractive and extractive systems. The model's construction includes a prototype extractor and an encoder-decoder. The prototype extractor extracts significant words, feeds them to the encoder, and the decoder creates the summary. Finally, the model was evaluated using ROUGE scores on the Newsroom and CNN/Daily mail datasets.

2.1.1 Literature Summary Results

Table 2.1 summarizes some previous results obtained by applying the ROUGE-F1 score to various datasets using encoder-decoder models for abstractive text summarization.

Table 2.1 Summary of the related work experimental metrics.

Model Name	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**	**Datasets**
RNN-seq2seq [13]	10.0	7.0	1.0	LCST
words-lvt2k-temp-att [8]	35.5	13.3	32.62	CNN/Daily Mail
COPYNET model[15]	34.4	21.6	31.3	LCST
rnn-ext+abs +RL+rerank[18]	40.9	17.8	38.54	CNN/Daily Mail
Transformer [19]	37.9	18.69	35.22	Gigaword
Transformer [19]	31.4	10.89	27.18	DUC2004
Scheduled sampling [20]	20.51	8.59	14.76	Amharic
Bidirectional LSTM [21]	33.0	0	33.0	Amazon food reviews
LC LenEmb LPAS [23]	35.60	17.04	33.12	CNN/Daily Mail
LC LenEmb LPAS [23]	33.48	23.01	30.98	NEWSROOM
Distraction M3 [16]	27.1	8.2	18.7	CNN/Daily Mail
Distraction model [16]	35.2	22.6	32.5	LCST

CHAPTER 3

BACKGROUND

We discuss the theoretical foundations of abstractive text summarization in this chapter, which introduces to the reader the fundamental theoretical principles and detailed explanations of the models utilized in this study, as well as the most often used datasets for abstractive and extractive text summarization.

3.1 Text Summarization

Summarization is the process of condensing a piece of material while retaining critical explanatory sections and the content's core. Due to the fact that manually summarizing literature is a time-consuming and frequently difficult activity, automating it is critical and serves as a substantial motivator for academic research.

Text summarizing is advantageous for a variety of natural language processing applications, including categorization, question answering, legal text summaries, news summary, and headline generation. Additionally, as part of the pre-processing phase, these systems may generate summaries, which aids in document length reduction.

The growth in the volume of text data available from a number of sources has occurred during the big data era. This vast amount of content contains an unlimited amount of information and knowledge that must be summarized effectively to be useful. With the proliferation of documents, extensive research in the field of natural language processing (NLP) for automatic text summarization is necessary. Automatic text summarizing is the process of automatically generating a brief and fluent sum-

mary without human intervention while retaining the original text's content.

It's quite difficult since, as humans, we often read a piece of content completely to ensure total comprehension and then write a description or summary emphasizing the key points. Automatic text summarization is a complex and time-consuming task since computers lack human intelligence and language abilities.

Numerous machine learning techniques have been presented for this subject. The majority of these solutions treat the subject as a classification problem, with the result indicating whether or not to include a sentence in the summary. Additionally, topic knowledge, Latent Semantic Analysis (LSA), Sequence to Sequence models, Reinforcement Learning, and Adversarial processes were utilized [24].

3.2 Methods and Technologies

Automatic summarization can be approached in two distinct ways: extraction and abstraction.

3.2.1 Extractive Methods

Extractive summarization utilizes a scoring mechanism to extract sentences from the text and integrate them into a coherent summary. This technique involves identifying key sections of text , cutting and sewing them together to create a simpler form. As a result, they rely primarily on the extraction of sentences from the original text. Currently, the majority of summary research has been on extractive summarization, which is simpler and produces naturally grammatical summaries without requiring extensive linguistic analysis. Additionally, extractive summaries extract the most significant lines from an input source, which can be a single document or a collection of documents.

A typical extractive summarization system flow is like follows:

1. Constructs an intermediate representation of the incoming text in order to find its most significant material. Typically, it works by calculating Tensorflow (TF)

metrics for each text contained in a matrix.

2. Evaluates the sentences in light of the representation, awarding each sentence a weight indicating its likelihood of being included in the summary.

3. Generates a summary based on the most pertinent sentences. Several researches have employed latent semantic analysis (LSA) to classify semantically significant utterances.

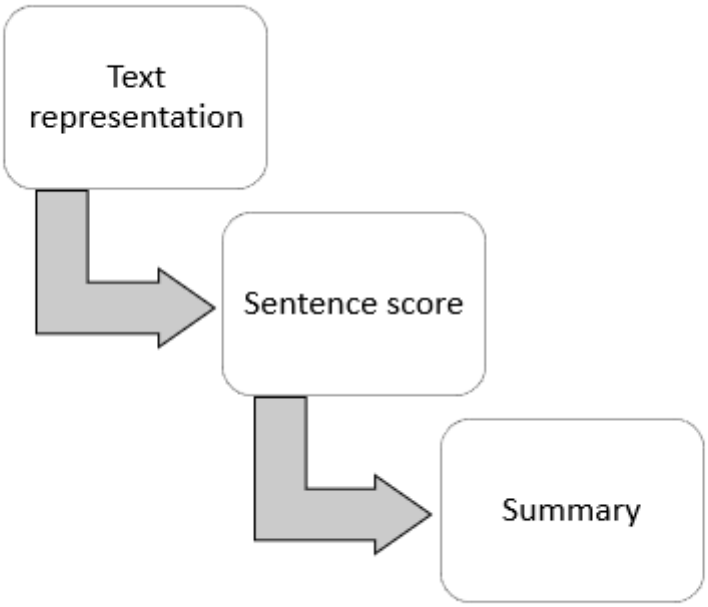

Figure 3.1 Extractive summarization systems

3.2.2 Abstractive Methods

Abstractive summarization methods construct summaries by understanding the text using advanced natural language algorithms to generate a new, shorter text that delivers the most important information from the original text, requiring sentence rephrasing and including information from the full text. Abstractive summarization has been considered a more complex problem than extractive summarization due to the complexity of extracting relevant information from a document and automatically producing coherent text. This fact has also resulted in less research on abstractive summa-

rization than on extractive summarization. Abstractive summarization earliest techniques relied on statistical methods inspired by machine translation and typically concentrated on summarizing one sentence at a time. Many of these early methods relied heavily on heuristics and dictionaries to perform word substitution [25]. Alexander, Chopra, and Weston pioneered the use of neural methods for end-to-end abstractive summarization by introducing an encoder-decoder model with attention. This model was developed in conjunction with previous work in the field of machine translation [4].

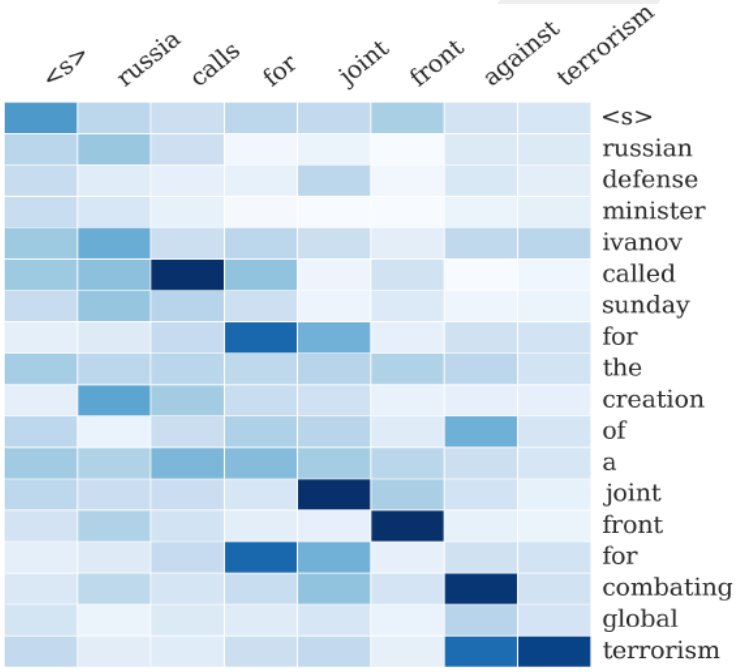

Figure 3.2 An example of the attention-based summarization output [4]

3.3 RNN encoder-decoder

The type of encoder-decoder model is widely used in tasks that include mapping one sequence to another sequence [8] . Due to language is composed of a series of characters, phrases, and sentences, the model is applied to a variety of language-related activities, including machine translation and summarization. The simple encoder-decoder model depicted in Figure 3.3 for language tasks generally consists of two RNNs, with one acting as an encoder and the other as a decoder [26].

As input, a sequence of vectors $x = \{x_1...., x_n\}$ is given by encoder. Typically, each vector is processed into the encoder's hidden state *ht*. When a conventional RNN is used, each vector is processed by performing the following calculation [5]:

$$h_t = f(x_t, h_{t-1}) \tag{3.1}$$

In equation 3.1, f may be a weight matrix multiplication of the inputs and an activation function. However, it is frequently dependent on Long Short-Term Memory (LSTM) cells [7]. The difference between this and a feedforward network is that the input to f includes the hidden state h_{t-1} from the previous time step. The final time step's resulting encoding is often referred to as context vector c [5]. The context vector c represents the final part of the data that is fed to the decoder. The decoder, like the encoder, has its own hidden states s_t, which are measured as :

$$s_t = f(s_{t-1}, c, y_{t-1}) \tag{3.2}$$

The final result is determined as follows:

$$y_t = g(s_t, c, y_{t-1}) \tag{3.3}$$

Where g denotes an activation function that generates probabilities, for example, softmax.

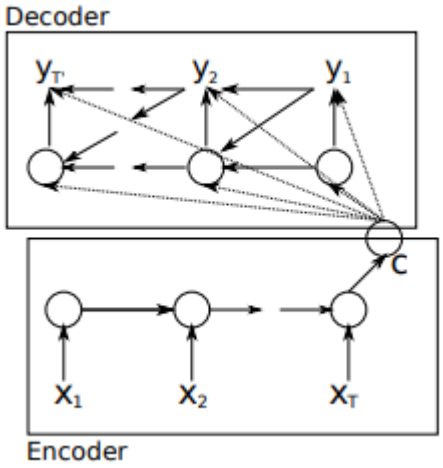

Figure 3.3 Encoder-Decoder model [5]

3.4 Long Short Term Memory

The Long Short Term Memory (LSTM) network is a type of recurrent neural network structure that overcomes the typical RNN's inability to learn long-term dependencies. It was developed for the first time by Hochreiter and Schmidhuber [27] and has since been refined and popularized by a large number of researchers [28]. Through the use of properly designed gates and memory cells, the LSTM's "deep in time" structure enables it to learn when to forget and how long to keep state information.

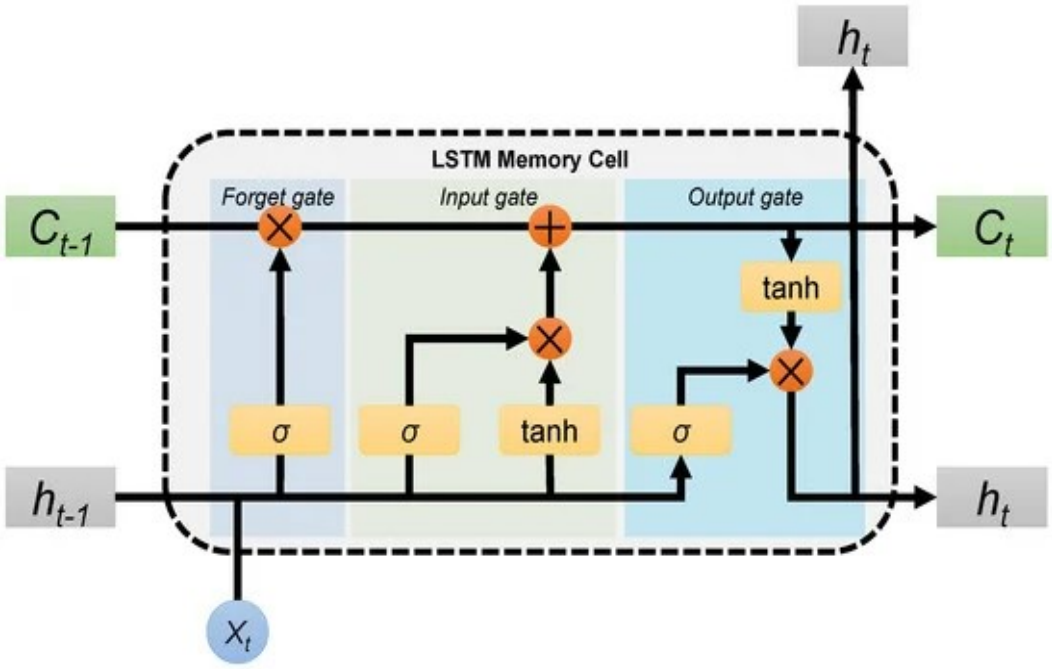

Figure 3.4 Long Short Term Memory [29]

3.5 Language Modelling

The objective of language modeling is to determine the chance that a sequence of text will be the successor of a particular text. Language models are typically used to anticipate the following word in a sequence depending on the preceding word. Additionally, this can occur at a variety of levels, including character, expression, and even sentence. Due to the fact that the data required to train a language model contains unstructured text, this is often considered an unsupervised activity.

3.5.1 Text Representation

When working with natural language, it is necessary to convey the textual information in a machine-readable format. The most often used way of text representation is through the use of word embeddings such as word2vec [30] or GloVe [31], as well as large vocabularies. However, it is conceivable for out of vocabulary (OOV) terms to occur with huge vocabularies as well. Sennrich et al. [32] suggested one strategy

for overcoming OOV. They proposed that rather than utilizing words, subwords be encoded using byte pair encoding (BPE). This method generates embeddings for these subword units on the internal level.

3.5.2 Sequence-to-Sequence

A sequence-to-sequence (seq2seq) model, alternatively referred to as an encoder-decoder model, is used to translate a variable-length input to a variable-length output, where the input and output lengths may differ. Sutskever et al.[7] suggested a two component seq2seq model consisting of an encoder and a decoder. Both components contain a collection of recurrent units, each of which accepts a single element as input. The encoder encodes the input sequence word by word by computing the hidden state *hi* for each timestep *i*. It then transmits the final secret state, h_i, to the decoder, which uses it as its beginning state. The encoder's final hidden state is employed to provide the decoder with a fixed-length vector representation of the full input sequence. The decoder then uses the hidden state to generate output y_i and the next state *si* for each timestep. Figure 3.5 depicts a hybrid encoder-decoder. When the input sequence is very long and the encoder's final hidden state is required to encapsulate all of the information in a single fixed-length vector, information is lost.

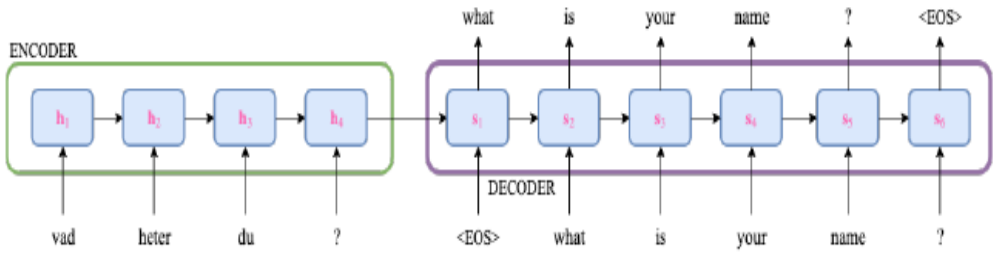

Figure 3.5 Sequence-to-sequence model [7]

3.5.3 Attention

Bahdanau et al. [5] introduced the attention method for machine translation tasks in 2015. The constraint of encoding the entire input sequence into a single fixed context vector was established as a bottleneck in earlier encoder-decoder RNNs [5]. By performing the attention mechanism, the quality of the translation can be improved, especially for longer text sequences.

The attention mechanism has the capability of storing all of the encoder's hidden states, as shown in Figure 3.6 The decoder will then use the information contained in each of these hidden states as input when determining the next word in the output sequence. Specifically, rather than calculating a single common context vector c, each decoding step's context vector is calculated as a weighted total of all previous hidden states ($h_1, ..., h_t$), as shown in equation 3.4 [5].

$$c_i = \sum_{j=1}^{T} \alpha_{ij} h_j \tag{3.4}$$

In matrix $\boldsymbol{a_{ij}}$, the weights are determined as follows:

$$a_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T} \exp\left(e_{ik}\right)} \tag{3.5}$$

Where e_{ij} consider an alignment model that trains to match the hidden states of the encoder to the hidden states of the decoder. The alignment model may be a feedforward neural network that is fed the concatenation of the hidden states of the decoder and encoder. Luong and Manning [33] make additional suggestions, such as dot product attention, which uses the dot product between the decoder hidden state and also the decoder states as the alignment model. By assigning a higher weight to appropriate words at each decoding step, longer sequences can be remembered and the decoder is fed only the most pertinent information. A popular addition to the standard seq2seq model is the use of bidirectional encoders, which encode both the input and the reversed input into hidden states. Concatenation of both the forward and backward

hidden states is then performed and fed to the decoder. It has been demonstrated that bidirectional encoders perform better when encoding longer sequences.

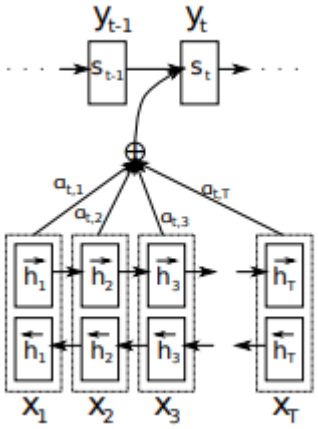

Figure 3.6 Encoder-Decoder with attention [5]

3.6 Transformer

The previous hidden state h_{t-1} is needed to measure the current hidden state h_t in the encoder-decoder model with attention. This sequential architecture reduces training performance by not completely utilizing the processing power of Graphics Processing Units (GPU). To solve this problem, Vaswani et al. [10] created the Transformer model. In this model, recurrence is fully eliminated, and only the attention mechanism in conjunction with feedforward networks is used to map dependencies between the input and output. The Transformer calculates attention using Scaled Dot-Product Attention, which is a scaling factor applied to the Luong dot product attention. It performs a dot product calculation between a query, a key, and a value:

$$\text{Attention}\,(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.6}$$

Scaling is performed using the $\sqrt{d_k}$ term, which d_k is the dimension of the input K. The K, V, and Q matrices each represent something different depending on the

context. They are all outputs from the previous layers in the encoder and decoder. In this sense, the process of focus is referred to as self-attention. In encoder-decoder layers, the Q matrix represents the output of the previous decoder layer, while K and V represent the encoder layer's output. The attention mechanism in this sense operates similarly to the attention mechanism in RNN-based encoder-decoder models [10].

Parallelization of the Scaled Dot-Product process on multiple linear projections of the input results in the output being concatenated and multiplied by an additional weight matrix. The authors refer to this implementation as Multi-Head Attention. The Multi-Head Attention layer's output is later used as the input for a feedforward. Furthermore, residual links are used to bypass the Multi-Headed Attention and feedforward layers. After normalization, the residual connections are applied to the output of these layers.

The both encoder and decoder use the same structure, with the exception that the decoder uses a variant of Multi-Head Attention called Masked Multi-Head Attention. Masked Multi-Head Attention masks potential performance locations, stopping the decoder from ”peeking” into the future during training.

Since the Transformer does not make use of recursion, the authors had to add a positional encoding to the input sequence so that the model could keep track of its order. This was accomplished in the original article by the use of sine and cosine positional encoding, although other types of positional encoding may also be used.

The Transformer design is depicted in Figure 3.7 The model will be fed positionally encoded word embeddings as input, and its output will be converted to probabilities via a linear followed by a softmax layer.

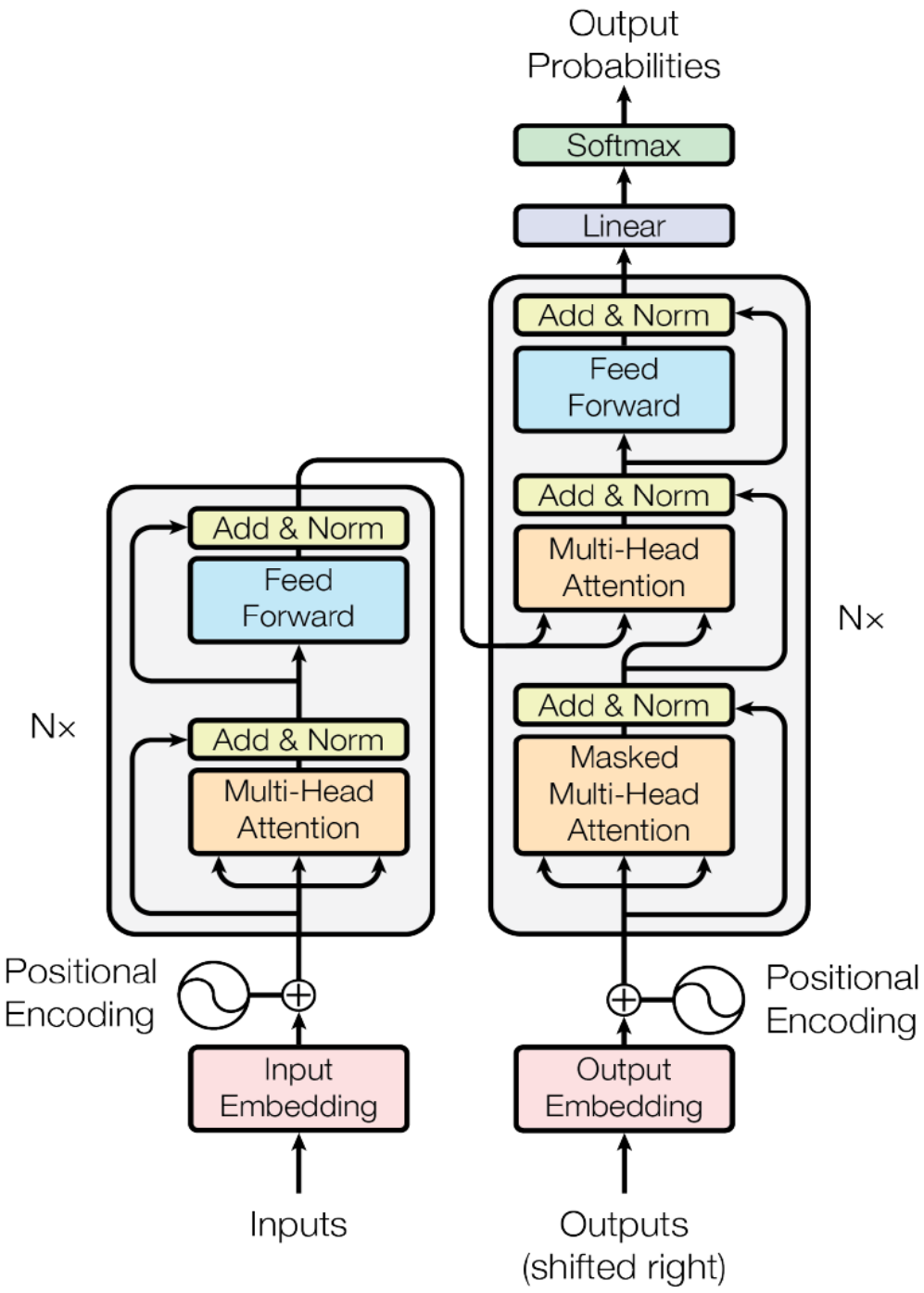

Figure 3.7 Transformer [10]

3.7 Evaluation Metrics

To evaluate the similarity between the two documents. There are some proven metrics that the abstractive summarization task uses to compare summaries written by human which are referred to as references, with the summary provided by the computer is called a candidate or a hypothesis.

3.7.1 BLEU

"BLEU (Bilingual evaluation understudy)." [34] is a commonly used text similarity metric. BLEU is a well-established metric for text summarization, despite being originally used for machine translation tasks. As shown in Equation 3.7, the BLEU score is measured using an adjusted n-gram precision and a brevity penalty determined by the candidate and reference lengths.

$$BLEU(x) = BP \cdot \exp\left(\sum_{n=1}^{N} (w_n \cdot \log(p_n))\right) \tag{3.7}$$

Where

$$BP = \begin{cases} 1 & r < c \\ e^{(}1 - r/c\big) & r \geq c \end{cases} \tag{3.8}$$

As a starting point, consider "dealing with people who gossip about you behind your back" as reference and "dealing with people constantly complain" as candidates for the $N = 1$ case. The precision of the n-gram, denoted by p_n in Equation 3.7, is 3/6, as only " dealing ", "with", and "people" from candidate occur in reference. According to Equation 3.8, the brevity penalty is equal to e (1-9/6), which equals 0.6065. For these two sentences, only the verb component is associated, the BLEU score is 0.3033. In another example, we see a new case in which "The adorable cat playing with the ball" is being a candidate, and "A lovely pet enjoying" is the reference. In this case, the precious of n-gram is zero because there are no match words between candidate and reference. Therefore, the BLEU score for these two sentences is zero even though they have almost the same meaning.

3.7.2 ROUGE

ROUGE is another of the automatic evaluation metrics, which is a measure of the number of overlapping n-grams between the produced summary and one or more reference summaries [35]. ROUGE-1, ROUGE-2, and ROUGE-L are the most frequently used versions in previous studies. ROUGE is a widely used metric in part because it has been shown to align with human summary quality assessments [36]. Due to the fact that numerous papers use ROUGE-n to evaluate abstractive summarization models, the metric is well-suited for comparing models with each other. ROUGE-n is defined as:

$$\text{ROUGE-N} = \frac{\sum_{S \in Re\,\text{ferenceSummaries}} \sum_{gram_n \in S} (\text{count}_{\text{match}}(\text{gram}_n))}{\sum_{S \in \text{ReferenceSummaries}} \sum_{gram_n \in S} (\text{count}(\text{gram}_n))} \tag{3.9}$$

As a starting point, consider ”dealing with people who gossip about you behind your back” as reference and ”dealing with people constantly complain” as candidates for the $N = 1$ case. ROUGE-1 is then 1/3, if Equation 3.9 is followed, since only ” dealing ”, ”with”, and ”people” from the reference occur in candidate. In another example, we see a new case in which ”The adorable cat playing with the ball” is being a candidate, and ”A lovely pet enjoying” is the reference. In this case, the result of measurement is zero because there are no match words between candidate and reference. Therefore, the ROUGE score for these two sentences is zero even though they have almost the same meaning.

3.7.3 METEOR

METEOR is also another evaluation metric that is used to evaluate summaries created automatically. METEOR was originally designed for automated machine translation evaluations, as it calculates sentence-level matching score between two sentences, typically a created translation and a reference translation [37]. When a reference summary is available, the METEOR metric can be used to compare it to a system-generated summary.

3.8 Related datasets

Nallapati et al. generated one such data collection for this assignment, which is based on CNN and Dailymail stories[8]. Over 300,000 pairs of human-written text summary pairs comprise the CNN/Daily Mail dataset. The Gigaword corpus, which is similarly based on news items, is another example of a dataset that has been previously used in the literature[12]. This dataset is significantly larger than the CNN/Dailymail counterpart, but it makes use of headlines and/or the first sentence of the text as reference summaries, which are frequently shorter in length than the CNN/Dailymail counterpart [12]. The following table summarizes the most frequently used datasets for abstractive and extractive text summarization.

Table 3.1 Datasets for abstractive and extractive tasks.

Datasets	**References**
CNN / Daily Mail	[38]
Gigaword	[4]
X-Sum	[39]
DUC 2004	[40]
Google Dataset	[41]
Wikihow	[42]
NEWSROOM	[43]
Webis-tldr-17-corpus	[44]
LCST	[13]
Inshorts	[45]

CHAPTER 4

METHODOLOGY

4.1 Datasets

In this study, we use two distinct datasets with different input and target lengths to determine whether or not the LSTM or Transformer model is affected by the length of the text.

4.1.1 Inshorts Dataset

This dataset is a news aggregation and summary service that collects and summarizes news articles from across the web. It contains news headlines and summaries, as well as their sources [45]. The headline column will serve as a summary of the story, while the short column will serve as context. Columns (Source, Time, and Date) that are extraneous to this work are omitted. There are 55,104 pairings of news headlines in this collection. The news data comprises almost 75,000 unique words, whereas the headlines have little under 29,000. The survey's longest news series is 400 words in length, while the average length of news is 368 words, and headlines are 96 and 63 words in length, respectively [46]. For the final train-test split, the following setting was used:

- Number of train samples: 54,104 (98.2%)
- Number of test samples: 1,000 (1.8%)
- Total samples 55,104 (100%)

Figure 4.1 shows the length distribution of inputs (news) and targets (summaries) in the Inshorts news dataset.

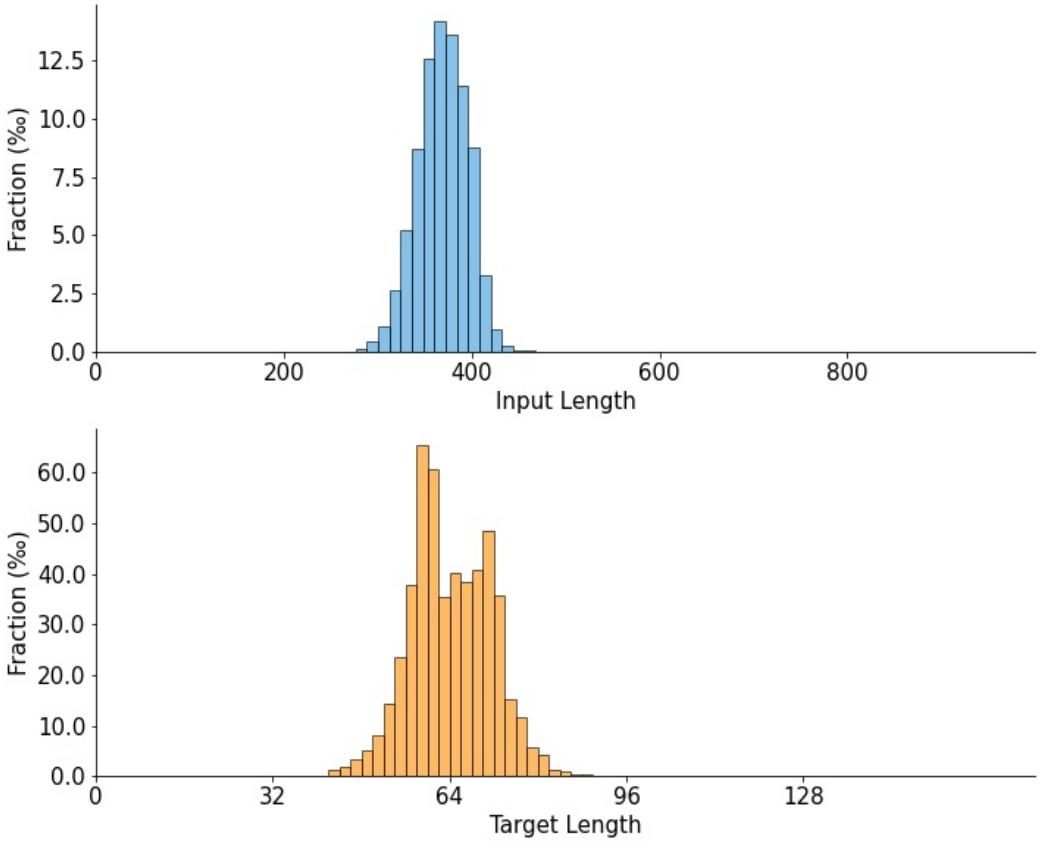

Figure 4.1 Input and target lengths histogram for the Inshorts dataset

4.1.2 CNN-Dailymail Dataset

The CNN/Dailymail data collection contains 311,971 summary-document pairings from news stories. The default setting was applied, which resulted in the following final train-test-validation split:

- Number of train samples: 287,113 (92.0%)
- Number of validation samples: 13,368 (4.3%)
- Number of test samples: 11,490 (3.7%)
- Total samples: 311,971 (100%)

Figure 4.2 shows the length distribution of inputs (articles) and targets (highlights) in the CNN/Dailymail dataset.

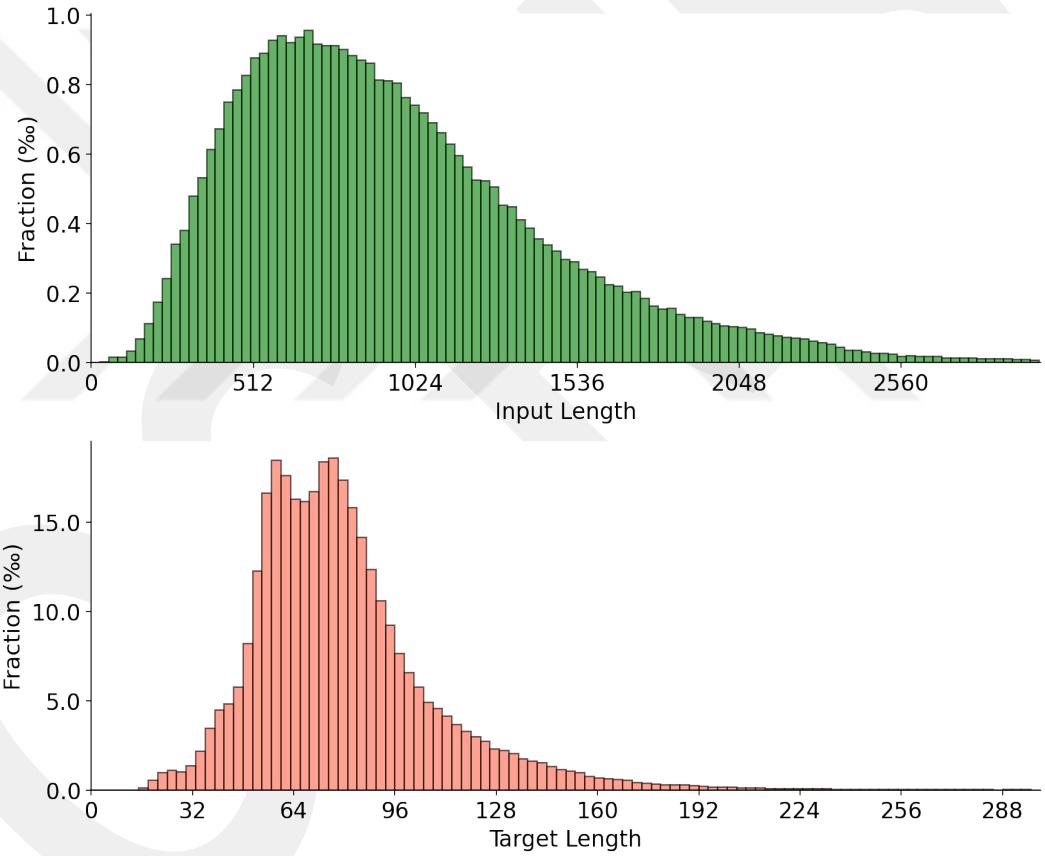

Figure 4.2 Input and target lengths histogram for the CNN/Dailymail dataset

This split has been utilized previously [8],[3], which motivated us to apply it in this thesis as well. The CNN/Dailymail data set utilized in this work is non-anonymised and was obtained from See [38]. The input was tokenized into 32,781 sub-words.

Using sub-words as tokens rather than full-length words allows for reduced vocabulary sizes and enables the decoder to construct phrases it has not encountered during training, which has been found to improve performance on machine translation tasks [32].

4.2 Preprocessing for Transformer

The preprocessing of data is an important step in the development of machine learning applications. TensorFlow's tokenization API essentially encompasses all data cleaning and preprocessing operations. In this study, we perform the necessary preprocessing before training the Transformer with the attention model. Preprocessing steps in this work include the following:

- Padding summary sequences with start ("go") and end ("stop") tokens to aid in identifying their beginning and end of the target.
- Remove all punctuation marks.
- Converts text to lower case. a vocabulary dictionary in which words are arranged according to their frequency of occurrence and a mapping between words and their token equivalents is included.
- Convert textual data to tokens that can be utilized directly in the model.
- Padding/truncating the sequences to a specific length in order to provide a model with more generalized data.

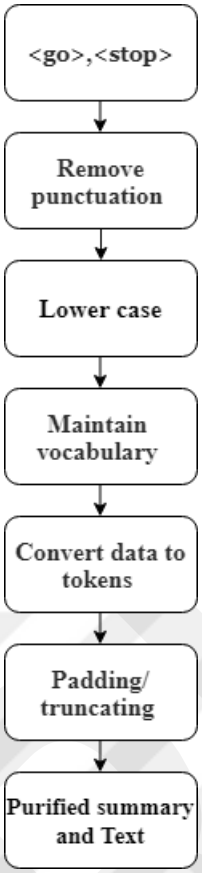

Figure 4.3 Data Preprocessing

4.3 Target Model (Transformer)

The Transformer is implemented using the Transformer model from the TensorFlow library. This model comes with several default hyperparameter sets that can be used for a variety of machine learning tasks. For summarization on the Inshorts and CNN/Dailymail datasets, some of the hyperparameter for the Transformer model is recommended by the authors of the Attention is all you need paper [10]. In the previous study, the different setup transformer was used to produce summaries with high ROUGE scores [47]. The most significant hyperparameters for both datasets can be observed in table 4.1

Table 4.1 Hyper-parameters for Transformer model

Hyperparameter	Value
num_layers	6
d_model	128
dff	512
num_heads	8
dropout	0.1
warmup_steps	4000
Optimizer	Adam
EPOCHS	20
BUFFER_SIZE	20000
BATCH_SIZE	64
encoder_maxlen for Inshorts dataset	55
decoder_maxlen for Inshorts dataset	15
encoder_vocab_size for Inshorts dataset	76362
decoder_vocab_size for Inshorts dataset	29661
encoder_maxlen for CNN/Dailymail dataset	520
decoder_maxlen for CNN/Dailymail dataset	180
encoder_vocab_size for CNN/Dailymail dataset	662056
decoder_vocab_size for CNN/Dailymail dataset	176514

4.4 Hyperparameters

Optimizer

The Adam optimizer is the default optimizer used by the TensorFlow library for the majority of models [48]. The Adam optimizer has three customizable parameters: α, β_1, and β_2, where $alpha$ is the step size and β_1 and β_2 are the decay rates for the first and second moments, respectively.

Dropout

Dropout is a regularisation technique that is used to avoid overfitting in neural networks by deactivating units [49]. Dropout is set to a value between 0 and 1, indicating the probability of a unit being dropped. Dropout values can be defined differently for each layer. The default value is used during training, and this value is set to 0 during inference.

Warmup

The warmup is a technique for beginning training with a slower rate of learning and gradually increasing it after a certain number of steps [50]. Several choices include maintaining a constant initial learning rate or increasing it linearly until the desired number of warmup steps is reached. The learning rate warmup steps hyperparameter specifies the number of warmup steps and is set to the default value for each model. Additionally, there is a learning rate decay that operates in the opposite direction, meaning that after the warm-up process, the learning rate begins to decline again according to the decay scheme selected.

Model Size

Apart from the hyperparameters mentioned previously, the number of layers and attention heads can also be modified. This can be accomplished by either selecting one of the Transformer's many predefined versions from the tensor2tensor library or by modifying the num heads and num hidden layers parameters. The default version is used in this thesis, which employs four encoder and decoder layers, each of which contains eight Multi-Head Attention layers running in parallel.

4.5 Preprocessing for LSTM

We followed several steps during the data pre-processing phase. To begin, we transformed all of the texts to lower case and then separated them. The English language contains numerous contractions, including ain't, aren't, didn't, doesn't, don't, and I'm. As a result, contractions were inserted at the data preprocessing stage. To create a clear text, an unnecessary component must be removed. Regular expressions can be used to eliminate superfluous components. The following step is to eliminate meaningless stop words. After stop words, we lemmatize words that contain variants of the same word. After completing all steps found a purified or clean text.

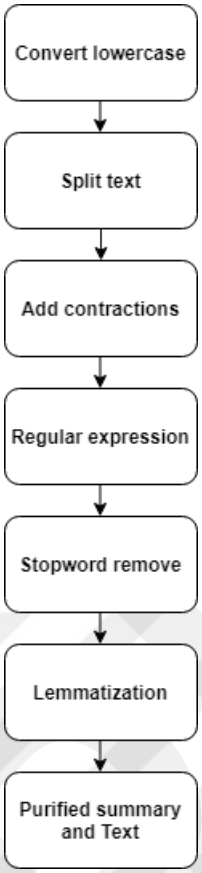

Figure 4.4 Data Preprocessing for LSTM

4.6 Baseline Model (LSTM)

The seq2seq baseline is LSTM based seq2seq this model also implemented using the TensorFlow library. In this study, the encoder is implemented using three LSTM layers, and the decoder with attention mechanism is implemented using one LSTM layer. There are two distinct attention techniques available: Luong and Bahdanau, as well as the option of using with or without bidirectional encoders.Figure 4.5 shows the workflow for abstractive text summarization using LSTM model.

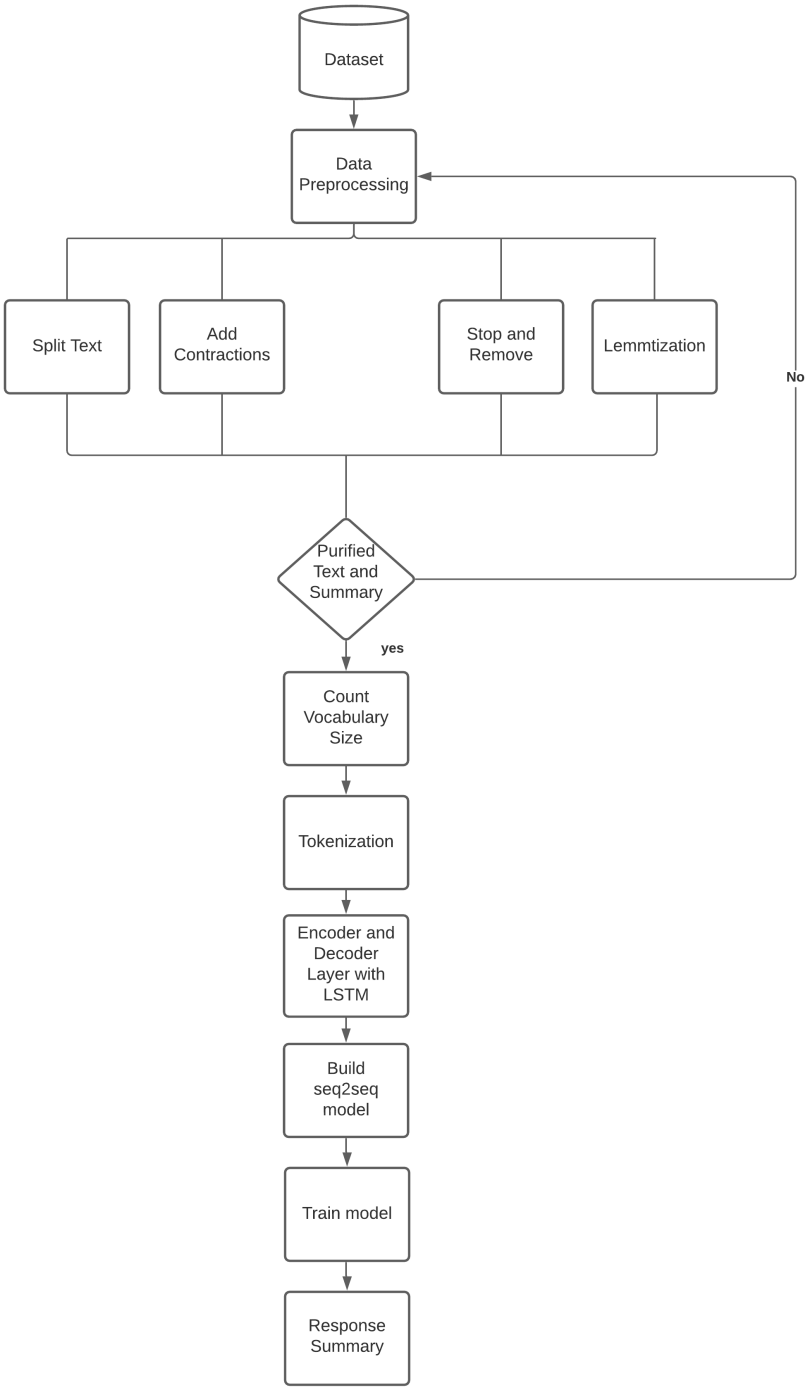

Figure 4.5 Working Flow of LSTM model

4.7 Hardware

The model was trained and evaluated using Google Colab. Google Colab is a free cloud computing service that supports both GPUs and TPUs. When using Google Colab Notebooks, the GPU is made available for free, which helps keep costs down. The GPU is designed for machine learning with Tensorflow, which significantly reduces the time and cost required to train machine learning models. For the storage, Google Drive was used to store the dataset and the model checkpoints.Since, training on GPU requires permanent storage and Google Colab has no permanent storage.

4.8 Evaluation

The test was carried out automatically using ROUGE-1, ROUGE-2, and ROUGE-L F1 results. The ROUGE score is used in part because it has been shown to correlate with human evaluations of summary content [33], but also because it is a widely used measure that allows for comparison with previous studies. Both ROUGE variants may be used to assess recall, accuracy, or a combination of the two using the F1 score. The F1 score was chosen for this study because it is less influenced by summary duration and also offers a good balance of recall and precision [35].

CHAPTER 5

RESULTS AND DISSECTION

This chapter discusses the initial testing findings as well as the final evaluation based on the ROUGE score. The initial test runs were conducted to compare the transformed with the LSTM model, as well as to investigate the influence of altering the maximum input length and utilizing different hyper-parameter settings.

5.1 Loss

The cost of a wrong prediction is loss. That is, the loss is a numerical value reflecting how badly the model predicted on a single instance. The loss is 0 if the model's prediction is perfect; otherwise, it is larger. The objective of model training is to identify a set of weights and biases with a low loss function. In our study, we notice the loss is decreasing which, means the learning process is improving and the models are being trained properly. Figures 5.1 and 5.2 show the loss for LSTM and Transformer models.

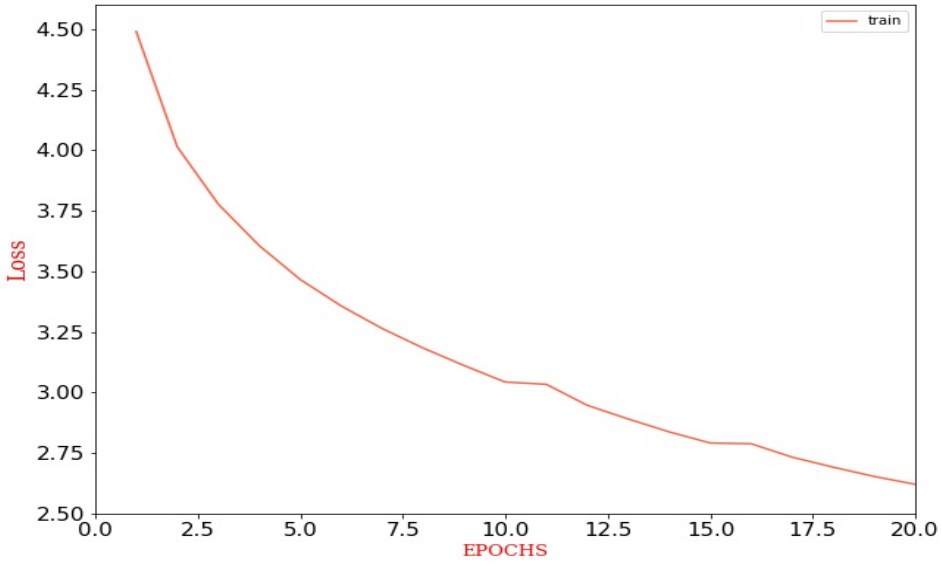

Figure 5.1 Loss for LSTM

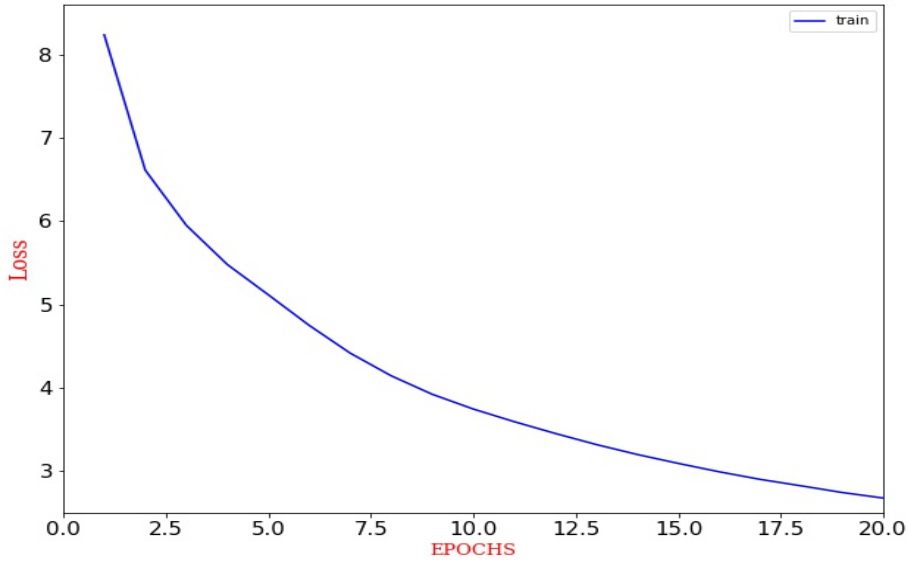

Figure 5.2 Loss for Transformer

5.2 Transformer Model Results

The transformer option initially appeared to perform well after testing the model on new data. However, we needed to verify that the model was tested on data that had the same distribution as the training data or was similar to Inshorts and CNN/Dailymail datasets. Moreover, we first created a special inference step because the model is accustomed to receiving the target sequence as an input to the decoder (Teacher Forcing). But, while testing, we fed the decoder with the past predictions. However, We obtain the output for the first word, append it to the first word, and jointly obtain the third word as the output for the sequence of the first and second words, once again appending this output to the decoder inputs and repeating until the output sequence length reaches maxlen or the predicted word is ”stop”. After evaluating the data using the ROUGE score, the results of Inshorts dataset were significantly better than predicted for such a small amount of data. The results of the Transformer's initial evaluation up to 20 EPOCHS for Inshorts and CNN/Dailymail datasets are shown in Figures 5.3 and 5.4.

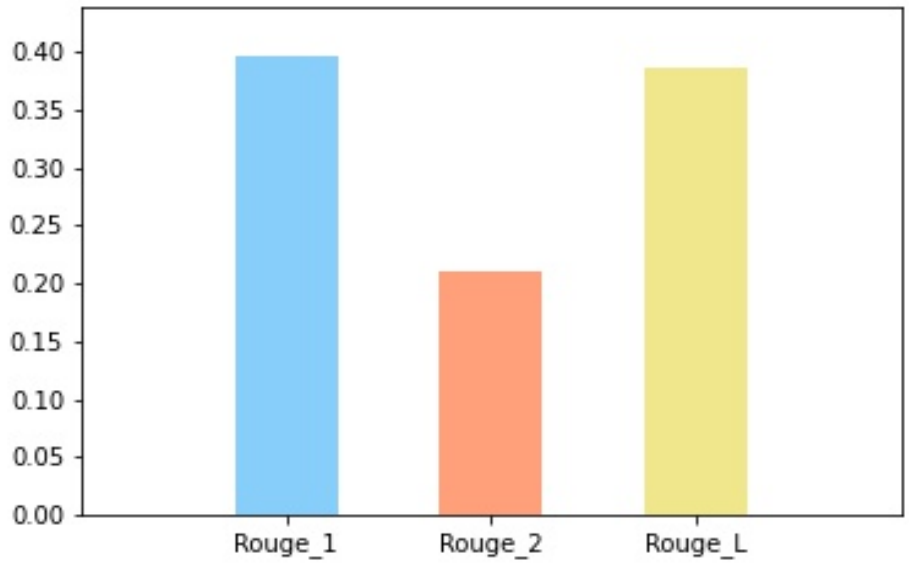

Figure 5.3: ROUGE-1, ROUGE-2 and ROUGE-L F1-scores from initial testing on Transformer up to 20 EPOCHS for Inshorts dataset

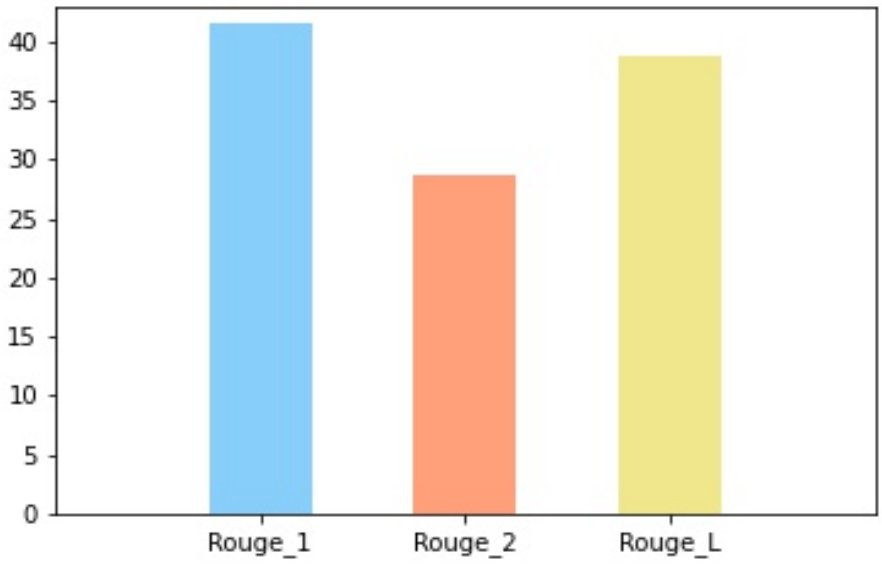

Figure 5.4: ROUGE-1, ROUGE-2 and ROUGE-L F1-scores from initial testing on Transformer up to 20 EPOCHS for CNN/Dailymail dataset

5.3 LSTM Model Results

TensorFlow library has been used to create a sequence-to-sequence model. After the model training is complete, we will be able to generate a machine-generated summary. To create a summary, we take an input text from the dataset and randomly decide the summary length. We used an attention-based encoder to represent the parameter. To achieve faster convergence, the vanilla gradient descent optimizer was employed. Regarding the data pre-processing, 10 percent of the whole dataset was kept for testing, while the remaining 90 percent was retained for training. Total data usage in CNN daily mail dataset is 100000 news stories' text, including 90000 for training and another 1000 for testing. The results of the LSTM's initial evaluation up to 20 EPOCHS are shown in Figures 5.5 and 5.6.

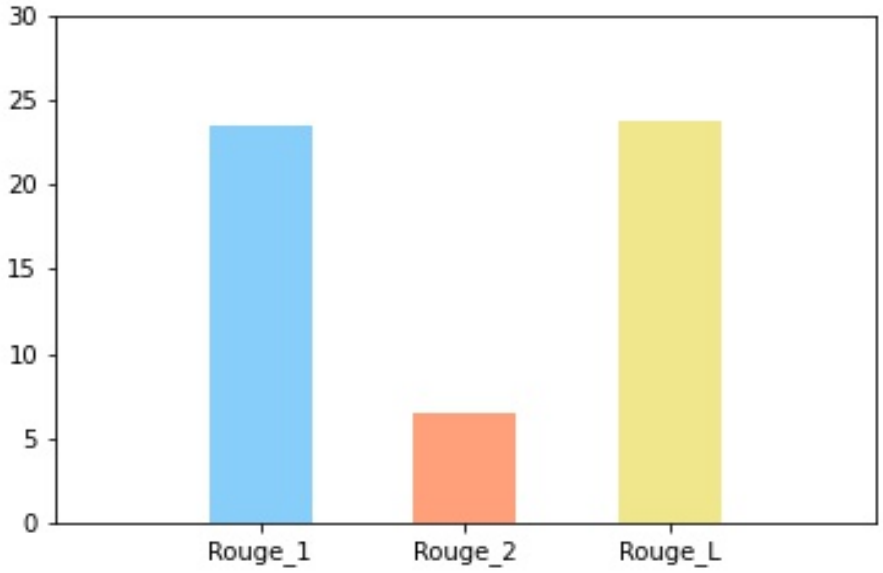

Figure 5.5: ROUGE-1, ROUGE-2 and ROUGE-L F1-scores from initial testing on LSTM up to 20 EPOCHS for Inshorts dataset

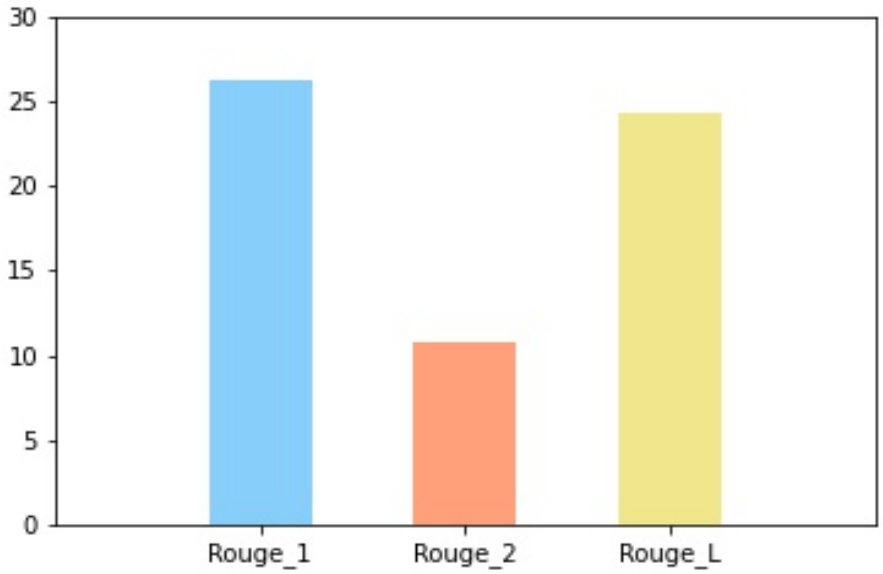

Figure 5.6: ROUGE-1, ROUGE-2 and ROUGE-L F1-scores from initial testing on LSTM up to 20 EPOCHS for CNN/Dailymail dataset

5.4 Truncation Length

How input and output lengths are reduced has an effect on both training duration and evaluation score. In this thesis study, a comparable truncation of the input, 520 subwords, was applied as in [3]. Another popular technique has been to truncate the output to 75 bytes before evaluating it using ROUGE, as this was a standard set at prior Document Understanding (DUC) conferences [8]. When comparing the Transformer to the seq2seq baseline, it was discovered that how output lengths are truncated has an effect on both the seq2seq baseline and the Transformer model scores. We anticipated a higher ROUGE score, but truncating had a greater impact on the time required to train our models. This decreases the amount of time spent on training. The explanation for this is not explored in detail, however it could be related to the issue with the datasets' size.

5.5 Comparison of Training Time

Due to the fact that the models are trained up to a specified number of training steps, a comparison of their training speed is also made. Figure 5.7 show an overview of the overall training time up to 10 EPOCHS for both models. As can be seen from the Figures, the Transformer trains faster than the RNN models.

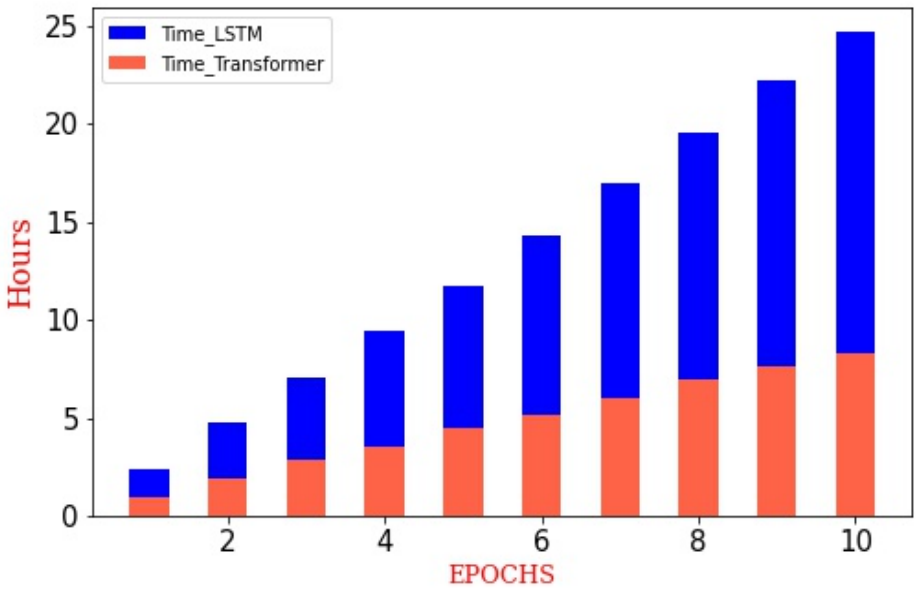

Figure 5.7: A comparison of the training times, in hours, for all tested models up to 10 EPOCHS

5.6 Comparison of the Seq2seq Baseline and Transformer model

ROUGE-1, ROUGE-2, and ROUGE-L F1-scores are shown in Table 5.1 for the models that achieved the greatest scores during training, up to 20 EPOCHS, which in this case was a transformer placed and lengthy attention with a bidirectional encoder. Both models were examined using 100 randomly chosen samples from the test set with different input lengths. In every category and for all input lengths, the Transformer model outperforms the seq2seq baseline.

Table 5.1: The average ROUGE F1 scores for the Transformer and seq2seq baseline for Inshorts dataset with varying maximum input lengths.

Model	ROUGE-1	ROUGE-2	ROUGE-L	Dataset
Transformer	39.63	21.07	38.56	Inshorts
LSTM	23.45	6.53	23.74	Inshorts
Transformer	41.56	28.60	38.71	CNN/Dailymail
LSTM	26.24	10.89	24.57	CNN/Dailymail

5.7 The Outputs

In table 5.2, 5.3, 5.4, 5.5 some examples are shown displaying the summary generation from the Transformer model. The example summaries are generated from the test set of both Inshorts and CNN/Dailymail datasets.

Table 5.2 First example of Inshorts dataset output

Text	japanese conglomerate toshiba plans to sell part of the chip business with an aim to recover from the previous 1 3 billion accounting scandal as per the reports from reuters the company reportedly has started accepting bids with an early interest shown by the development bank of japan who has already invested in the semiconductor operations of tokyo based seiko holdings corporation.
Original summary	toshiba to sell part of its chip business
Transformer output	toshiba to sell its first chip business

Table 5.3 Second example of Inshorts dataset output

Text	china is planning to develop an 34 exascale 34 supercomputer that will be capable of performing more than a billion billion calculations per second the prototype for such a computer will be released in either 2017 or 2018 the supercomputer will be 1 000 times more powerful than china's tianhe 1a which was identified as the world 39 s fastest computing system in 2010.
Original summary	china to develop new supercomputer in 2 years
Transformer output	china to develop the supercomputer in one year

Despite the fact that the summary in a small dataset produces positive results, ROUGE scores are often low for short summaries and appear to grow for datasets with longer summary length budgets, even when document length also increases significantly, as appears in the tables below.

Table 5.4 First example of CNN/Dailymail dataset output

Text	thomas mulller l bayern dominate years come pep guardiola takes final training session ahead bayern munich clash manchester city time reflecting neuer wednesday game city champions league campaign focus sense bayern team might emulate barcelona side made core spain team 2008 2012 club country dominated football neuer jerome boateng philipp lahm bastian schweinsteiger mario gotze thomas muller perhaps capability the bundesliga cup basics said neuer but season chance play two finals berlin everybody wants it great thing play final capital city referring 2015 champions league final german fa cup final feels though bayern munich time taken original interview ludger schulze oliver trust.
Original summary	goalkeeper manuel neuer won the 2014 world cup with germany neuer did not realise the impact before meeting fans on holiday bayern munich no 1 now wants to win champions league final in berlin the 28 year old won the bundesliga and german cup double last season bayern munich play manchester city in their opening match on wednesday.
Transformer output	the 2014 world cup winner was found guilty of attempted murder in the 2012 he was sent to the hospital in the early hours of the day after the incident the 33 year old has been in the top three since the end of the season

Table 5.5 Second example of CNN/Dailymail dataset output

Text	cnn one worst hit cities philippines tacloban completely devastated typhoon haiyan capital city leyte province nearly 400 miles south east manila one first places hit storm around 200 000 people made homeless matter hours wall ocean water flattened houses tacloban city stadium one handful structures area withstand storm despite coastal location used refugee center thousands people taken shelter scroll slider see stadium storm cnn ivan watson toured typhoon devastated tacloban region air read personal account destruction saw .
Original summary	the stadium was one of just a handful of structures in tacloban to withstand the storm it is now being used as a refugee center where thousands have taken shelter scroll the slider above to see the tacloban stadium before and after the storm recommended
Transformer output	the 20 year old was found dead in a car park in the early hours of the weekend he was found dead in a car park in the early hours of sunday morning he was the first time he was in the country since 2007

CHAPTER 6

CONCLUSION

The Transformer was compared against an RNN-basedseq2seq model with attention using the ROUGE score as an automatic measure. The research question, "How does the Transformer model perform on the abstractive summarization task when compared to a seq2seq baseline, as measured by the ROUGE metric?" reveals that the Transformer produces higher average ROUGE scores. When comparing average ROUGE scores for different input length subwords, the Transformer outperforms the seq2seq baseline by 15.32 in ROUGE-1, 17.71 in ROUGE-2, and 14.14 in ROUGE-L. Additionally, the Transformer appears to be less impacted by changes in input length. In response to the second research question, "How does the Transformer model compare to a seq2seq baseline when evaluating the readability and grammar of generated summaries using the ROUGE score?" the results indicate that the Transformer outperforms the LSTM-based seq2seq model in all categories evaluated. The most significant distinction is in the area of non-redundancy, which is connected to the repeating problem encountered when using the seq2seq baseline model. Along with improving assessment scores, the Transformer trains faster and can handle longer input and output sequences without running out of memory. Moreover, the readability of the generated summaries was not evaluated in comparison to the human-written reference summaries in the CNN/Dailymail and Inshorts datasets. This would be a fascinating area to investigate because it could provide more insight about the readability of the system-generated summaries. However, this evaluation was omitted due to the study becoming too lengthy. In the future study, we plan to improve ROUGE metrics and compare the results to human-written reference summaries in the CNN/Dailymail and Inshorts datasets, as ROUGE metrics currently only analyze the symmetry and do not

have the ability to quantify the semantics.

REFERENCES

[1] M. Gambhir and V. Gupta, "Recent automatic text summarization techniques: a survey,"*Artificial Intelligence Review*, vol. 47, no. 1, pp. 1–66, 2017.

[2] V. Dalal and L. Malik, "A Survey of Extractive and Abstractive Text Summarization Techniques," in *2013 6th International Conference on Emerging Trends in Engineering and Technology.* IEEE, 2013, pp.109–110..

[3] M. Allahyari *et al.*, "Text summarization techniques: a brief survey," *arXiv preprint arXiv:1707.02268*, 2017.

[4] A. M. Rush and J. Weston, "A neural attention model for abstractive sentence summarization," *arXiv preprint arXiv:1509.00685,* 2015.

[5] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate, " *arXiv preprint arXiv:1409.0473*, 2014.

[6] W. Zeng, W. Luo, S. Fidler, and R. Urtasun, "Efficient summarization with read-again and copy mechanism," *arXiv preprint arXiv:1611.03382*, 2016.

[7] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[8] R. Nallapati, B. Zhou, C. dos Santos, C. Gulcehre, and B. Xiang, "Abstractive text summarization using sequence-to-sequence RNNs and beyond," *arXivpreprint arXiv:1602.06023*, 2016.

[9] P. J. Liu *et al.*, "Generating wikipedia by summarizing long sequences," *arXiv preprint arXiv:1801.10198*, 2018.

[10] A. Vaswani *et al.*, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[11] S. Chopra, M. Auli, and A. M. Rush, "Abstractive sentence summarization with attentive recurrent neural networks," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Lang. Technol.*, 2016, pp. 93–98

[12] Y.-S. Wang and H.-Y. Lee, "Learning to Encode Text as Human-Readable Summaries using Generative Adversarial Networks," 2018.

[13] B. Hu, Q. Chen, and F. Zhu, "LCSTS: A large scale chinese short text summarization dataset," in *Proc. Conf. Empirical Methods Natural Lang. Process*, 2015.

[14] Y. Zhang, "Evaluating the Factual Correctness for Abstractive Summarization," Internet: https://www-cs.stanford.edu/ yuhuiz/ assets/reports/factual.pdf [Apr. 15, 2021]

[15] J. Gu, Z. Lu, and H. Li, "Incorporating copying mechanism in sequence-to-sequence learning," *arXiv preprint arXiv:1603.06393*, 2016.

[16] Q. Chen, X. Zhu, Z. Ling, S. Wei, and H. Jiang, "Distraction-based neural networks for document summarization," *arXiv preprint arXiv:1610.08462*, 2016.

[17] A. See, P. J .Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks,"*arXiv preprint arXiv:1704.04368,* 2017.

[18] Y. C. Chen and M. Bansal, "Fast abstractive summarization with reinforce-selected sentence rewriting,"*arXiv preprint arXiv:1805.11080*, 2018.

[19] X. Duan, H. Yu, M. Yin, M. Zhang, W. Luo, and Y. Zhang, "Contrastive attention mechanism for abstractive sentence summarization," *arXiv preprint arXiv:1910.13114*, 2019.

[20] A. M. Zaki, M. I. Khalil, and H. M. Abbas, "AMHARIC abstractive text summarization,"*arXiv preprint arXiv:2003.13721*, 2020.

[21] A. K. Masum, S. Abujar, M. A. Talukder, A. K. Rabby, and S. A. Hossain, "Abstractive method of text summarization with sequence-to-sequence RNNs," in *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE, 2019, pp. 1–5.

[22] C. Zhu, R. Xu, M. Zeng, and X. Huang, "A Hierarchical Network for Abstractive Meeting Summarization with Cross-Domain Pretraining,"*arXiv preprint arXiv:2004.02016*, 2020.

[23] I. Saito *et al.,* "Length-controllable Abstractive Summarization by Guiding with Summary Prototype,"*arXiv preprint arXiv:2001.07331*, 2020.

[24] L. Gonçalves, "Automatic Text Summarization with Machine Learning." Internet: https://medium.com/luisfredgs/automatic-text-summarization-with-machine-learning-an-overview-68ded5717a25 [Feb. 15, 2021].

[25] K. Woodsend and M. Lapata, "Learning to simplify sentences with quasi-synchronous grammar and integer programming," in *n Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, 2011, pp. 409–420.

[26] K. Cho, B. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[27] Hochreiter, Sepp, and Schmidhuber, "Long Short-Term Memory," *Neural computation*, pp. 1735-1780, 1997 .

[28] K.Kazuya, "Supervised sequence labelling with recurrent neural networks." *Ph. D. thesis*, 2008.

[29] H. Fan, M. Jiang, L. Xu, H. Zhu, J. Cheng, and J. Jiang, "Comparison of Long Short Term Memory Networks and the Hydrological Model in RunoffSimulation," *Water*, vol. 12, no. 1, p. 175, 2020.

[30] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems,* 2013, pp. 3111–3119.

[31] J. Pennington and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[32] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909*, 2015.

[33] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[34] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: a Method for Automatic Evaluation of Machine Translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics,* 2002, pp. 311–318.

[35] C. Yew Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text summarization branches out,* 2004, pp. 74–81.

[36] I. Mani, "Summarization evaluation: An overview," 2001.

[37] M. Denkowski and A. Lavie, "Meteor universal: Language specific translation evaluation for any target language," in *Proceedings of the ninth workshop on statistical machine translation*, 2014, pp. 376–380.

[38] Abisee., "cnn-dailymail," Internet: https://github.com/abisee/cnn-dailymail [Jun. 20, 2021].

[39] S. Narayan , S. B. Cohen, and M. Lapata, "Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization," *arXiv preprint arXiv:1808.08745,* 2018.

[40] "DUC 2004." Internet: https://duc.nist.gov/duc2004/ [Mar. 4, 2021].

[41] katja-f, "Google Dataset," Internet: https://github.com/google-research-datasets/sentence-compression [Apr. 10, 2021].

[42] "WikiHow-Dataset," Internet: https://github.com/mahnazkoupa-ee/WikiHow-Dataset [Mar. 7, 2021].

[43] "Newsroom." Internet: https://summari.es [Apr. 11, 2021].

[44] "webis-tldr-17-corpus," Internet: https://github.com/webis-de/webis-tldr-17-corpus [Apr. 12, 2021].

[45] Shashi, "Inshorts News Data," Internet: https://www.kaggle.com/shashichander0-09/inshorts-news-data [Jan. 4, 2021].

[46] R. Jagtap, "Abstractive Text Summarization Using Transformers," Internet: https://medium.com/swlh/abstractive-text-summarization-using-transformers-3e774cc42453, Jun. 11, 2020, [Jan. 4, 2021].

[47] P. J. Liu, M. Saleh, and E. Pot, "Generating wikipedia by summarizing long sequences," *arXiv preprint arXiv:1801.10198*, 2018.

[48] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[49] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, pp. 1929–1958, 2014.

[50] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.

[51] M. Popel and O. Bojar, "Training tips for the transformer model," *arXiv preprint arXiv:1804.00247*, 2018.