Improving Extractive Summarization of Scholarly Documents using BERT and BiGRU

By

Sheher Bano

00000317550

Supervisor

Dr. Shah Khalid

Department of Computing

A thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science in Computer Science (MS CS)

In

School of Electrical Engineering & Computer Science (SEECS) ,

National University of Sciences and Technology (NUST),

Islamabad, Pakistan.

(May 2023)

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS/MPhil thesis entitled "Improving Extractive Summarization of Scholarly Documents using BERT and BiGRU" written by SHEHER BANO, (Registration No 00000317550), of SEECS has been vetted by the undersigned, found complete in all respects as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS/M Phil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.

Signature: Khalid

Name of Advisor: Dr. Shah Khalid

Date: 11-May-2023

HoD/Associate Dean: ____________________

Date: ____________________

Signature (Dean/Principal): ____________________

Date: ____________________

Approval

It is certified that the contents and form of the thesis entitled "Improving Extractive Summarization of Scholarly Documents using BERT and BiGRU" submitted by SHEHER BANO have been found satisfactory for the requirement of the degree

Advisor : Dr. Shah Khalid

Signature: _ Khalid ________

Date: 11-May-2023

Committee Member 1:Prof. Hasan Ali Khattak

Signature: [illegible]

11-May-2023

Committee Member 2:Dr Farzana Jabeen

Signature: Farzana

Date: 13-May-2023

Dedication

This thesis is dedicated to all the deserving children who do not have access to quality education especially young girls.

Certificate of Originality

I hereby declare that this submission titled "Improving Extractive Summarization of Scholarly Documents using BERT and BiGRU" is my own work. To the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics, which has been acknowledged. I also verified the originality of contents through plagiarism software.

Student Name: SHEHER BANO

Student Signature: [illegible]

Acknowledgments

I would like to express my sincere gratitude to all those who have contributed to my research and the successful completion of my thesis.

First and foremost, Praise be to Allah, the Lord of the worlds, the One who is the sustainer and creator of all that exists. He has the power to bestow honor upon whom He chooses and to humble whom He chooses. None of us can do anything without His will. From the moment I arrived at NUST until the day I left, He was the one who blessed me and opened doors for me, guiding me on the path to success. His generosity and kindness towards me during my research period were immeasurable and cannot be fully repaid. I am grateful for all that He has done for me and offer my sincere thanks and praise to Him.

I would like to express my deepest gratitude to my parents, who have always supported me and believed in me. I would also like to extend a special thanks to my supervisor, Dr. Shah Khalid, who provided me guidance and encouragement throughout my research journey. It was a true honor to work with such a kind and gracious person. I could not have completed my thesis without their help. Lastly, I would like to pay tribute to the National University of Science and Technology (NUST), which has provided me with a wealth of knowledge and opportunities.

Sheher Bano

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
BERT	Bidirectional Encoder Representations from Transformers
BiGRU	Bidirectional Gated Recurrent Unit
DLMNN	Deep Learning Modifier Neural Network
GLUE	General Language Understanding Evaluation
GNN	Graph Neural Network
GRU	Gated Recurrent Unit
HEATS	Hybrid Extractive and Abstractive Text Summarization
HMMs	Hidden Markov Models
LDA	Latent Dirichlet Allocation
LSTM	Long-Short Term Memory
MLM	Masked Language Mode
NLP	Natural Language Processing
NSP	Next Sentence Prediction
NTM	Neural Topic Model
RNN	Recurrent Neural Network
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
S-RNN	Simple Recurrent Neural Network
SQUAD	Stanford Question Answering Dataset
TF-IDF	Term Frequency-Inverse Document Frequency
TF-ISF	Term Frequency-Inverse Sentence Frequency

.

Abstract

Extractive summarization involves selecting and condensing key information from a text document, while preserving the overall meaning and coherence of the original content. There are several extractive summarization methods that have their own benefits and drawbacks. Despite the variety of approaches currently available, none of them are flawless and there is still potential for advancement in the field of automated summarization. One promising approach to extractive summarization is the use of deep learning models, such as BERT. BERT is a multilayer transformer network that has been pretrained on a large dataset for a variety of self-supervised applications, including language translation, question answering, and natural language understanding. However, BERT has a limitation in terms of input length, which makes it less suitable for summarizing long documents. In this study, we suggest an innovative approach that enables the use of BERT for long document summarization. Our method involves dividing the document into smaller chunks, each containing a single sentence. We then use BERT to generate sentence embeddings, and apply an encoder-decoder model on top of these embeddings to generate a summary. The encoder-decoder model is a type of neural network that is commonly used for machine translation and text generation tasks. We carried out experiments with two scholarly datasets, arXiv and PubMed, to evaluate the effectiveness of our approach. The results showed that our technique consistently outperformed several state-of-the-art models for extractive summarization. This demonstrates the potential of our method for improving the efficiency and accuracy of summarization tasks.

CHAPTER 1

Introduction

The exponential increase in digital text on the internet has rendered manual text summarization impractical due to the high costs associated with it in terms of time, effort, and financial resources [1]. However, current technology allows for the utilization of ATS approaches to succinctly capture the entirety of a text's context in a matter of seconds. This technique is particularly beneficial in the field of scientific research, where the sheer volume of text contained within articles can be overwhelming. The implementation of automatic text summarization allows for the efficient extraction of the key points contained within these articles, enabling individuals to stay informed of the latest developments in a timelier manner.

Text summarization is a method of condensing a large block of text into a smaller, more coherent version while still retaining the key points and overall meaning of the original document. The goal of text summarization is to provide a summary of the text that is both concise and representative of the entire context of the data. This can be extremely useful for reducing the time and effort required to read long documents, as well as for conserving storage space for the text data.

The two main approaches for text summarization are: abstractive and extractive [2]. Abstractive summaries are those that are written in the summarizer's own words and attempt to capture the essence of the original text [3]. Machines may find this challenging to do accurately, as it requires a deep understanding of the user's language and can potentially result in biased or misleading summaries. Extractive summarization, on the other hand, involves selecting the most important sentences from the input text and concatenating them to create a summary. This approach is considered to be more reliable and is therefore becoming more popular in industry.

Conventional methods for extractive text summarization include statistical and graph-based techniques such as TextRank [4] and TF-IDF [5, 6]. However, recent advances in NLP have led to the development of neural network-based approaches, such as BERT [7]. These models are trained on labeled data to perform specific tasks and have shown promising results in a variety of applications.

1.1 Motivation

While BERT is a promising method for text summarization, it has significant shortcomings when it comes to tasks that require reasoning with lengthy documents [8]. The self-attention memory of the transformer quadratically rises with the number of input tokens [9], making computation on document scale sequences potentially impossible. Furthermore, at the pre-training stage, BERT usually needs to define a predetermined maximum input length, which is typically 512 tokens [8]. Pre-training the full model on longer sequences is one option, but this requires a significant amount of computing power [9]. Other approaches to extending multi-layer transformer designs to larger sequences without changing the maximum length constraint have been presented. The first approach is to reduce the input sequence to the first 512 tokens by eliminating any text that exceeds the length limit. Obviously, handling huge articles that are frequently longer than this limit is not a viable choice. Another approach is to use the concept of sliding window [10], in which a window slides all over the document. [11] utilized this approach to handle SQUAD documents that were longer than the 512-token limit, while [12] used it for a lengthy document co-reference resolution task. This technique is applicable when the tokens need to be contextualized in their local surroundings because there is no connection between the windows [8]. But for the task of summarization global level understanding of document is necessary, it might be challenging to employ BERT in a way that allows for the acquisition of global context from the whole document. The aim of this work is to develop a BERT-based architecture that can extract important sentences from dense scientific texts, enabling researchers to quickly ascertain the key ideas in research papers. This will be accomplished by examining the suggested technique using the two main datasets for lengthy scientific articles, arXiv[1] and PubMed[2].

1.2 Problem Statement

BERT is a highly advanced language model that can generate context aware text embeddings, but its 512-token input length limitation presents a challenge for summarizing lengthy scientific articles. While BERT has achieved success in a variety of tasks, the input length restriction may hinder its effectiveness for summarization of long documents.

In our research, we propose a solution to this problem by developing a method that allows the use of BERT for summarization of long scientific articles despite this limitation. By utilizing this approach, we hope to overcome the input length constraint and enable the effective application of BERT to long document summarization tasks.

1.3 Contributions

This dissertation addresses the outlined research questions and challenges and presents two key contributions, as follows:

Contribution 1: *BERT-based Extractive Text Summarization of Scholarly Articles: A Novel Architecture-Accepted in ICAIoT2022(IEEE-Turkey): Second International Conference on Artificial Intelligence of Things.*

This part of the thesis presents an innovative Architecture for generating summaries of lengthy scholarly articles. The proposed Framework is based on BERT, which is employed to effectively process extended documents. Specifically, BERT is incorporated into an attention-based encoder-decoder model to capture the global context of the entire document, enabling the generation of concise and informative summaries.

Contribution 2: *Summarization of Scholarly Articles Using BERT and BiGRU: Deep Learning-Based Extractive Approach-Under review in Journal of King Saud University - Computer and Information Sciences.*

This part expands on Contribution 1 by utilizing algorithms and systematic flow diagrams to provide a detailed analysis of the proposed approach. This facilitates a struc-

[1]https://www.kaggle.com/Cornell-University/arxiv

[2]https://pubmed.ncbi.nlm.nih.gov/

tured presentation and a comprehensive understanding of the methodology's potential for application.

1.4 Thesis Organization

Remaining thesis organization is as follow;

1.4.1 Chapter 2: Theoretical Basis for the Study

This chapter provides a detailed overview of the essential concepts that are necessary for understanding the proposed approach.

1.4.2 Chapter 3: Literature Review

This chapter outlines previous work in the field of extractive text summarization as well as how to apply BERT on large text. This chapter discusses the research gaps of prior work in order to have a clear understanding of the significance of the proposed architecture.

1.4.3 Chapter 4: Design and Methodology

This chapter delineates the implementation of the proposed architecture, including the configuration of the baseline model, which serves as a point of comparison. Additionally, this chapter provides relevant information regarding the utilized framework, as well as the training hyperparameters employed in the experiment.

1.4.4 Chapter 5: Results and Discussion

This chapters show the results of proposed model and its comparison with state-of-the-art model with detailed discussion.

1.4.5 Chapter 6: Conclusion and Future Work Directions

This is the last chapter and it concludes all the work done in this thesis. Furthermore, this chapter discusses the future directions for this research work.

At the start of thesis a list of abbreviations is added to help readers understand the terminology used throughout the document. Towards the end of the thesis, Appendix A is included, which contains detailed information about the formulas used, and evaluation metrics applied in the research experiments conducted for this study.

Chapter 2

Theoretical Basis for the Study

2.1 Natural Language Processing (NLP)

The field of natural language processing (NLP), combining computer science, artificial intelligence, and linguistics, has enabled computers to comprehend and engage in human language. Massive amounts of natural language data must be processed, analyzed, and used to extract useful information via NLP systems.

In order to execute tasks that often require human-level language understanding, such as sentiment analysis, text summarization, machine translation, and speech recognition, NLP aims to give computers the ability to comprehend the meaning underlying human language. Tokenization, part-of-speech tagging, parsing, semantic analysis, and other computational techniques are used by NLP systems to analyze and comprehend language input.

The complexity, ambiguity, and context-dependence of human language make it one of the major obstacles for NLP. It can be challenging for computers to understand irony, sarcasm, and other figurative language, and because every language has its own grammatical rules, lexicon, and idioms, it is challenging to create an all-encompassing NLP system.

Despite of these difficulties, NLP has made notable advances recently, and its applications have expanded quickly. NLP is utilized, as an illustration, in chatbots for customer service, virtual assistants, and machine translation systems that let users translate text and speech from one language to another in real-time. Search engines that can comprehend natural language queries and deliver more pertinent results are being developed using NLP in the field of information retrieval.

2.1.1 Text Preprocessing

Preparing unprocessed text input for subsequent analysis and modelling is a critical step in NLP. Text preprocessing aims to clean up and standardize text data so that NLP algorithms can process it properly.

Here are a few steps involved in text preprocessing:

Lowercasing: Lower casing is the process of changing all capitalization in the text to lowercase. This step is crucial because many NLP algorithms interpret words with distinct case types as discrete entities.

Removal of Punctuation: Taking out the punctuation from the text. Punctuation is typically eliminated from text data to lower its dimensionality and make it easier to analyze.

Tokenization: Tokenization involves the process of breaking up the text into smaller pieces, like words or phrases. For the purpose of breaking the text down into manageable chunks for examination, tokenization is crucial.

Eliminating Stop Words: Stop words are often used words like "a," "an," "the," etc. that don't add much significance to the text. To shrink the amount of data and to make the NLP algorithms' processing efficient, these terms can be eliminated.

Lemmatization and Stemming: These are approaches that attempt to reduce words to their most basic form in order to decrease the dimensionality of the data. Lemmatization utilizes advanced techniques for reducing words to their base form, surpassing the capabilities of stemming, which removes suffixes to get words back to their root form.

Eliminating Special Characters and Numerals: Eliminating special characters and numbers from the text can make the data simpler and stop these aspects from confusing NLP algorithms.

Removing HTML and XML tags: These tags must be removed from text data that has been downloaded from the web in order to produce clean text data.

2.1.2 Statistical Based Methods

Natural Language Processing (NLP) relies heavily on statistical approaches since they give us a way to evaluate, comprehend, and predict text input. Following are a few of the typical statistical techniques employed in NLP:

TF-IDF: This approach is used to determine a word's significance within a corpus or text (a large collection of documents). Important words in the document are highlighted by the TF-IDF score since it considers both the frequency of the term in the document and its inverse frequency throughout the full corpus.

Naive Bayes: For classification tasks like sentiment analysis or spam filtering, the probabilistic method known as naive Bayes is applied. The technique uses probabilities to categorize text data into different groups and is based on the Bayes theorem.

Hidden Markov Models (HMMs): These are used to simulate sequences of events, like the words in a phrase or the letters in a word. These models employ probabilities to estimate the most likely series of occurrences given the observed data, .

Latent Dirichlet Allocation (LDA): It is a topic modelling technique that uses a generative probabilistic model. The technique identifies the themes present in a text or corpus and assigns probability to each pair of a document and a subject.

Word Embeddings: In a low-dimensional space, word embeddings serve as dense vectors that represent words. These vectors capture the syntactic and semantic links between words, making it easier for NLP models to process these interactions.

2.1.3 Artificial Neural Networks

Neural networks are now frequently used to produce cutting-edge outcomes in NLP tasks in recent years. An AI system called a neural network uses synthetic neurons to mimic the operation of a biological brain. As a result, the models can learn from their mistakes and advance via repetition. The vast amount of data accessible for training has also influenced the performance of these models. It is important to have a basic understanding of neural network concepts in order to properly appreciate how these models work.

The two type of Artificial Neural Networks are:

1. Single-Layer Neural Network (Perceptron)
2. Multi-Layer Neural Network

Single-Layer Neural Network (Perceptron): To categorize input that can be separated into linear groups, a single-layer perceptron is a basic type of neural network. Based on the design of a biological neuron, it functions by processing inputs as a vector and generating an output in the form of a binary signal. The fundamental parts of a perceptron are an input vector, an output signal, and a model made up of a weight vector and a bias term.

The input vector, x, is denoted by x = (x1, x2,..., xd), where d is the total number of features in the input. The classification of the input data is represented by the output signal, y. A bias term, b, plus a weight vector, w, make up the perceptron model. The strength of each input node is represented by the weight vector, which has the form w = (w1, w2,..., wd). The perceptron's baseline prediction is represented by the bias term.

The prediction function, which is depicted in 2.1.1, serves as the foundation for the perceptron's predictions.

$$y = f(w^T x + b) \tag{2.1.1}$$

Where w^T is the transpose of the weight vector, b is the bias term, and f is an activation function. Based on the inputs, weights, and bias, the activation function is a mathematical function that establishes the output signal. Different neurons can perform different types of activation-related tasks.

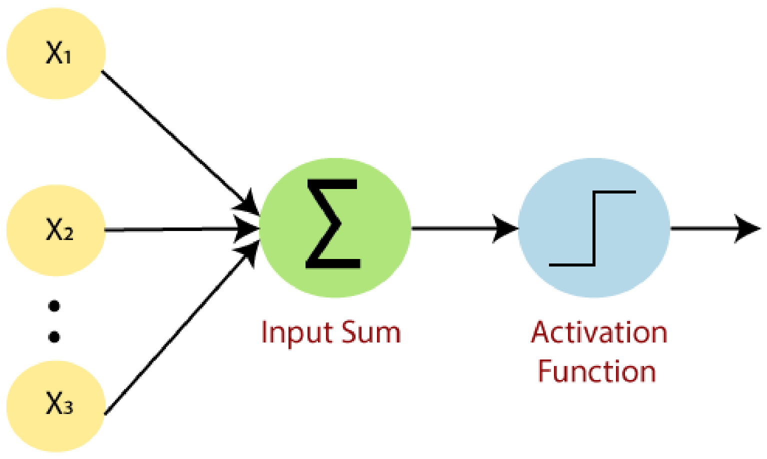


Figure 2.1: Working of Perceptron model: Illustrating how input features are combined to generate a binary prediction.

Fig. 2.1, which illustrates the perceptron-based prediction process, provides an illustration of how the perceptron model works. The final output signal, which represents the categorization of the input data, is created by combining the inputs, weights, and bias through the activation function.

Multi-Layer Neural Network: A perceptron has the restriction that it can only do linear classification and cannot learn features. This restriction can be solved by combining numerous single-layer perceptrons to create a multi-layer neural network with hidden layers. With a single-layer network, it is impossible to learn features or solve non-linear functions, but with Multi-Layer neural network, it is possible.

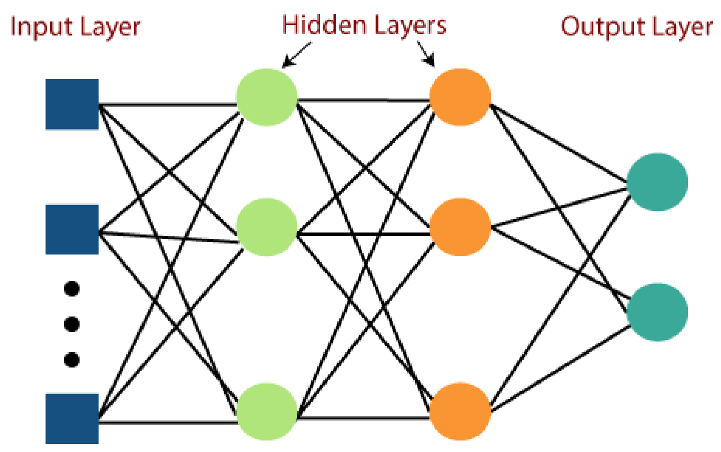


Figure 2.2: Multi-layer neural network with an input layer, two hidden layers, and an output layer. The input data is processed by the hidden layers through weighted computations and activation functions before producing the final output.

Fig. 2.2 depicts an example of a multi-layer neural network. It has an input layer, two hidden layers, and an output layer. The input layer accepts input data while the hidden layers carry out feature learning, enabling the network to understand intricate correlations between inputs and outputs. Based on the data gathered by the hidden layers, the output layer makes the final prediction.

A multi-layer neural network has the advantage of being able to solve non-linear functions, as opposed to a single-layer network, which can only do so. In many applications where the relationship between the inputs and outputs is nonlinear, this feature is essential. A multi-layer neural network created from several perceptrons has the capacity to learn intricate patterns and carry out more challenging tasks.

2.2 Sequential Models

Sequential data refers to data that must keep the order of its elements, while sequential models refer to the models used to handle this kind of data. Sequential models need to be able to recall crucial elements of the sequence in addition to maintaining the sequence's order. For instance, if the data consists of a collection of sentences, it is essential that the model retain certain words, such as the title of the primary subject, in order to comprehend the context of the sentences.

For example, consider this sentence "Y was going to School. He forgot to bring his English notebook" as an example. To recognise that the person is going to school and not somewhere else, the model must be able to recall the keywords "Y" and "School". Sequential models differ from other models in that they can remember crucial sequence elements, which makes them ideal for using with textual data in NLP.

2.2.1 Recurrent Neural Networks (RNNs)

For processing sequential data, a neural network called RNN is utilized. As a result, the network is able to handle sequences of various lengths and can take context and sequence order into account. Loops, which enable the network to transfer information from one stage in the sequence to the next, are incorporated into the network architecture to achieve this.

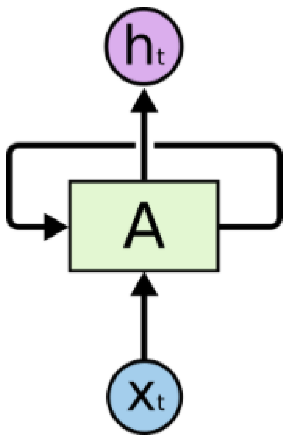


Figure 2.3: Basic RNN structure: an input sequence is processed to generate an output for each step.

A basic RNN structure is depicted in Fig. 2.3, in which the input sequence is processed by the network and an output is generated for each step in the sequence. The same RNN model is unpacked in Fig. 2.4, with the loops shown as links between the various phases.

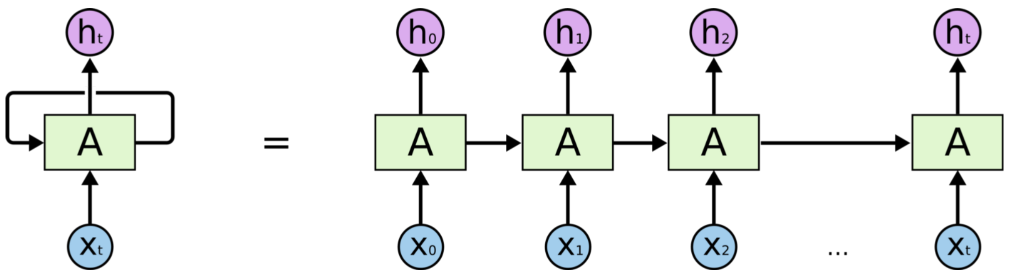


Figure 2.4: Unpacked RNN model: Loops illustrate that how the same weights and biases are applied iteratively to each element of the input sequence, while updating the hidden state.

Various types of RNN models are:

1. Simple RNN
2. Long-Short Term Memory (LSTM)
3. Gated Recurrent Unit (GRU)

Simple RNN Model: The Simple Recurrent Neural Network (S-RNN) was initially proposed by [13] and later investigated by [14] for application in Natural Language Processing (NLP). Sequence tagging and language modelling can be accomplished with the help of S-RNNs. But they also bring up an issue with gradients conveying data necessary for parameter adjustments. The "exploding or vanishing gradients problem" is a phenomenon that develops over time as a result of these gradients' potential to grow or shrink quickly. As a result, the gradients may grow or shrink to the point where the model is unable to train. Later, [15] suggested the LSTM architecture to solve this problem, which was successful in resolving the exploding and vanishing gradient issue.

Long-Short Term Memory (LSTM): These networks are a subclass of Recurrent Neural Networks (RNNs) created for the purpose of learning long-term dependencies [16]. LSTMs consist of memory cells, input and output gates, as well as a forget gate, in contrast to conventional RNNs [17]. The gates decide how much information from earlier states should be maintained, and the memory cell keeps track of dependencies that are present in the input sequence.

Gated Recurrent Unit (GRU): A RNN version called Gated Recurrent Units (GRUs) was created to address the vanishing gradient issue that standard RNNs have [18]. The reset and update gates, which govern the information flow into and out of the memory cell, make up a GRU [19]. The update gate defines how much of the current input should be added to the memory cell, while the reset gate determines how much of the prior state should be forgotten. GRUs have been found to be computationally effective and to be effective in a number of NLP tasks, including text categorization and machine translation [20].

2.2.2 Encoder-Decoder Model

A popular method for completing sequence-to-sequence (seq2seq) Natural Language Processing (NLP) tasks, like translation, is the Encoder-Decoder architecture [21]. Encoder and decoder are the two parts of the architecture. A sequence is fed into the encoder, which creates an encoding vector that the decoder uses as input. In the decoder stage, auto-regression is utilized to predict an output at each step until a predefined end-token is generated. The architecture shown in Figure 2.5 illustrates an RNN encoder-decoder setup for sequence-to-sequence tasks.

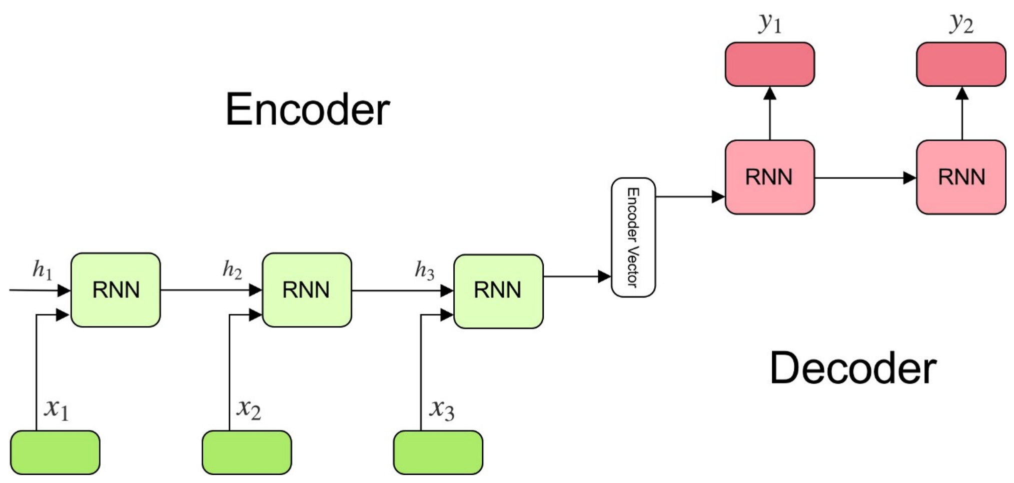


Figure 2.5: Encoder-Decoder model: Where "x" stands for the input sequence,"h" stands for the hidden state and "y" stands for the output sequence.

2.2.3 Attention Mechanism

The input sequence of traditional encoder-decoder models for natural language processing must have a fixed length vector as a restriction. This indicates that the model may struggle to comprehend later portions of a series or perhaps forget earlier portions of lengthy sequences. In [22], authors suggested using attention in encoder-decoder models as a solution for this problem.

Encoder-decoder models with attention enable the model to concentrate on important segments of the input sequence. To do this, the decoder receives all hidden states from the encoders and scores each one based on how closely it matches a word in the input sequence. Instead of depending solely on one context vector, the scores enable the model to choose which portions of the sequence to focus on.

In a translation task, the attention scores demonstrate which elements of the input sequence are most important to the intended result. The attention scores can either be monotonic, which denotes that they take a straight line, or they can be nonmonotonic, denoting a more nuanced relationship between the input and output words. Thus, the inclusion of Attention in encoder-decoder models solves the drawbacks of traditional encoder-decoder models and permits the model to comprehend and remember information from the input sequence more effectively.

2.2.4 Transformers

The encoder and the decoder are two essential parts of the transformer model, which is based on the attention process. [23] proposed this architecture in 2017 as a solution to the issues with recurrent models that prevented parallelization, led to longer training periods. High levels of parallelization, a continuous sequence and an attention-based, non-sequential nature of transformers are all made possible. The encoder and decoder operate together to translate sentences from one language to another in machine translation jobs, which frequently require them. Fig. 2.6 illustrates the architecture of transformer model proposed by [23].

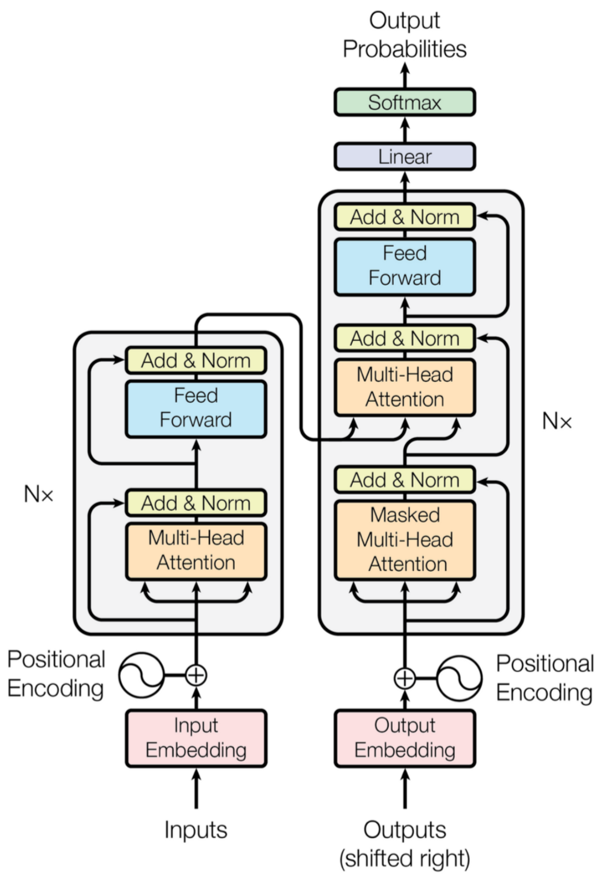


Figure 2.6: Transformer model: A neural network architecture used in natural language processing that includes both an encoder and a decoder composed of multiple layers of self-attention and feedforward neural networks.

Encoder: The encoder is made up of a number of layers, each with two sub-layers. The target sequence and input sequence are the same in the first sub-layer of the multi-head self-attention mechanism. The "multi-head" feature refers to the parallel computation of the attention utilizing scaled dot-product attention. The second sub-layer comprises a simple, fully connected feed-forward network that takes positional information into account. Each sub-output layer is added to its corresponding input using a residual connection and layer normalization. The encoder generates an embedding or vector representation for each word in the input text, representing the words using numerical values.

Decoder: The decoder has a similar structure to the encoder but adds a masked multi-head attention mechanism as a second sub-layer. This mechanism, in contrast to self-attention, inhibits the decoder from attending to following points in the target sequence, preventing predictions from being produced by looking forward. During the translation process, the decoder predicts one word at a time, continuing until the end of the sentence. This prediction is achieved by combining the embeddings generated by the encoder with the previously generated translated sentence.

Limitations of Transformer: Due to their limitation of processing input with a maximum length of 512 tokens, Transformers are constrained to handle inputs within this restriction. This is because as sequence length increases, the amount of memory and processing power required grows quadratically. Sequences longer than 512 tokens can typically be cut in half to fit inside the restriction.

2.3 Bidirectional Encoder Representation from Transformers (BERT)

In 2018, [24] introduced BERT, a pre-trained deep learning model used for NLP tasks. It is built on the Transformer architecture, a neural network that specializes in handling sequential data like text. BERT is engineered to comprehend the context and meaning of words in sentences and can be adapted to a specific NLP task by incorporating some extra layers into the pre-trained model, such as text classification or named entity recognition.

One of BERT's standout features is its bidirectional approach, enabling it to predict the meaning of a word by examining the entire input sequence. Prior language models were unidirectional, only analyzing the words to the left or right of the current word, but not both. By adopting a bidirectional approach, BERT gains a more comprehensive understanding of the context and correlation between words, leading to more precise predictions.

BERT has set the bar high for NLP benchmarks, including the GLUE benchmark and the Stanford Question Answering Dataset (SQuAD). It has also been applied to various areas, such as chatbots, sentiment analysis, and language translation, making it a popular tool for researchers and practitioners in the NLP field.

2.3.1 Embeddings(Input and Output) Generation:

A token embedding layer is used by BERT, like other language models, to transform each input token into a vector representation. The segment and position embeddings are two extra embedding layers that give it a unique edge. Fig. 2.7 shows these three embedding layers.

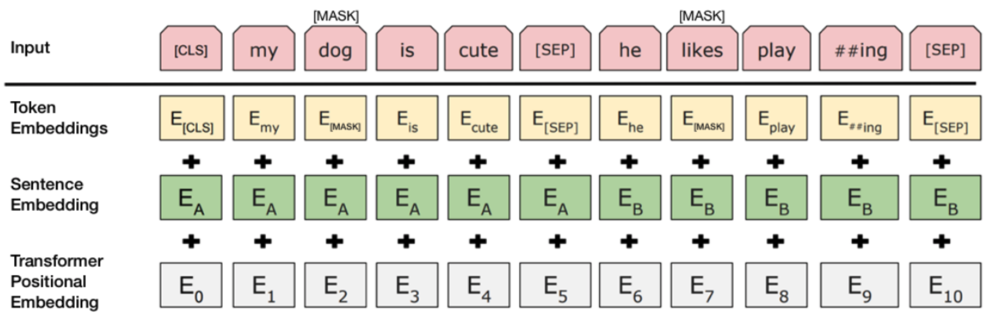

Input
[MASK]
[MASK]
[CLS] my dog is cute [SEP] he likes play ##ing [SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} $E_{[MASK]}$ E_{is} E_{cute} $E_{[SEP]}$ E_{he} $E_{[MASK]}$ E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Sentence Embedding
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Transformer Positional Embedding
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 2.7: Illustration of token, segment, and position embedding layers used by BERT to transform input tokens into vector representations.

Prior to the conversion of input text into a vector representation through embedding layers, WordPiece is employed to segment the text into tokens. Subsequently, the token, segment, and position embeddings are concatenated to generate an input embedding for each token.

Word Piece: BERT employs the WordPiece tokenization approach introduced by [25]. This method uses a fixed vocabulary set to break words into smaller chunks (referred to as WordPieces) in order to handle unusual words better. BERT's vocabulary consists of 30,000 WordPieces. A breakdown of two words into subwords is shown in Fig. 2.8.

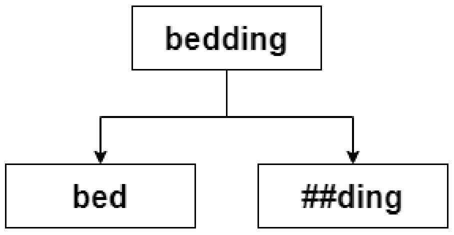


Figure 2.8: Breaking down word into sub-words using WordPiece tokenization.

More letters can be separated from a word the more uncommon it is. The character ## denotes all subwords except the first subword. The words "bed" and "ding," for instance, can be separated from the term "bedding". Due to its close relationship to the word "bed", the subword "bed" effectively conveys the sense of "bedding".

Token Embedding: In the initial stage of the BERT model, the token embedding layer is responsible for creating vector representations of the tokenized input. For every token, this layer produces a 1x768 vector with a hidden size of 768. If there are N input tokens, the token embedding will yield a matrix of shape Nx768 or a tensor of 1xNx768.

Segment Embedding: In BERT, it is possible to process a pair of input sentences, as depicted in Figure 2.10. These input sentences undergo tokenization and are subsequently concatenated to form a pair of tokenized sentences. To distinguish between the two sentences and indicate the sequence as binary, BERT utilizes the [SEP] token. This labeling sequence is then expanded to match the matrix shape of the token embeddings layer, which is Nx768, where N represents the number of tokens. For instance, in the case of the paired input depicted in Figure 2.9, the segment embedding results in a matrix shape of 8x768.

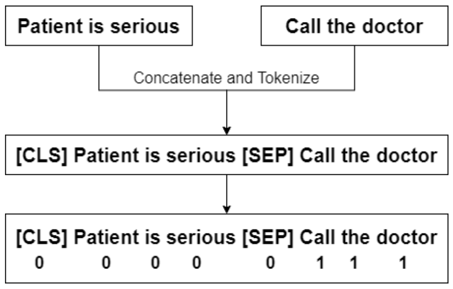

Patient is serious
Call the doctor
Concatenate and Tokenize
[CLS] Patient is serious [SEP] Call the doctor
[CLS] Patient is serious [SEP] Call the doctor
0 0 0 0 0 1 1 1

Figure 2.9: Labels in binary format generated from pairs of inputs.

Position Embedding: Due to its transformer-based architecture, BERT does not process tokens in a sequential manner. This means that in order to maintain the order of the tokens, position embeddings must be utilized. In order to ensure that BERT doesn't forget the order of tokens, position embeddings are utilized.

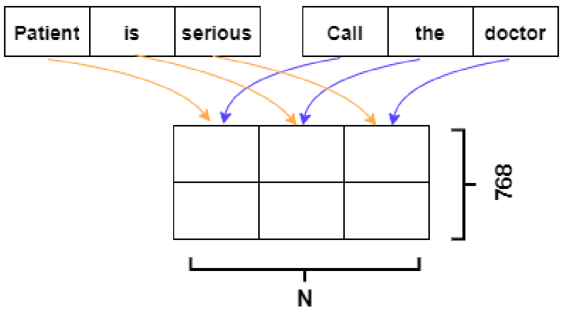


Figure 2.10: Position embeddings.

This layer serves as a look-up table, where each row's index corresponds to a specific token position. As depicted in Fig. 2.10, two sentences with identical vector representations for the words can be represented by the position embeddings. For instance, the words "Patient"-"Call", "is"-"the", and "serious"-"doctor" have the same vector representations for the sentences "Patient is serious" and "Call the doctor", respectively.

2.3.2 Pretraining:

In the pre-training phase, BERT undergoes training on two unsupervised tasks simultaneously: the Masked Language Model (MLM) and Next Sentence Prediction (NSP) [26].

Masked Language Model: In the pre-training phase of BERT, a task called Masked Language Modeling (MLM) is carried out as an unsupervised approach to enable BERT to grasp comprehensive bidirectional representations. This task involves randomly masking 15% of all WordPiece tokens in the input sequence and replacing them with a special [MASK] token. BERT is then trained to identify and predict the masked tokens, which helps it learn the relationships between words and the context in which they appear.

Next Sentence Prediction: It is an unsupervised task that is part of BERT's pre-training phase. Its goal is to help BERT understand the relationship between two sentences. BERT is trained to predict whether a given sentence B follows a sentence A in a document. To achieve this, 50% of the input pairs are composed of sentence pairs where the second sentence is the consecutive sentence in the corpus. The remaining 50% consists of sentence pairs with the same first sentence, but the second sentence is randomly chosen from the corpus. For instance, if sentence A is "The cat sat on the mat," then 50% of the time, sentence B would be "It was a sunny day" (subsequent sentence from the corpus), while the other 50% would be a randomly selected sentence from the corpus.

2.3.3 Fine Tuning:

Fine-tuning refers to the process of customizing the pre-trained BERT model for specific NLP tasks using supervised learning. It entails replacing the fully connected output layers of the pre-trained BERT model with new output layers designed to provide answers to the specific NLP problem at hand. The fine-tuned model undergoes supervised learning on labeled data, updating the weights of the output layers. Since only the output layer weights are modified during fine-tuning, the learning process is relatively swift [24].

2.4 ROUGE: Recall-Oriented Understudy for Gisting Evaluation

Recall-Oriented Understudy for Gisting Evaluation (ROUGE) metrics [1] is used to assess the quality of extracted summaries. ROUGE is a set of metrics that are primarily designed for assessing automatic text summarization systems. It compares the automatically generated summary with the reference summaries.

The main metrics in ROUGE package are ROUGE-N (Counts the number of overlapping n-grams between automatically generated summary and reference summaries), ROUGE-L (checks for the longest common sequence between system generated summary and reference) and ROUGE-S (searches for consecutive words between system generated and reference summaries but these are separated by other words).

Recall in ROUGE gives information about how much reference summary is recovered by system summary

$$\text{ROUGE Recall } (R): \quad R = \frac{\text{Number of overlapping n-grams in generated summary}}{\text{Total number of n-grams in reference summary}}$$

Precision in ROUGE gives information about how much relevant the system generated summary is.

$$\text{ROUGE Precision } (P): \quad P = \frac{\text{Number of overlapping n-grams in generated summary}}{\text{Total number of n-grams in generated summary}}$$

Finding the harmonic mean between recall and precision will find the F-Score in ROUGE.

$$\text{ROUGE F-measure } (F): \quad F = \frac{2 \times P \times R}{P + R}$$

2.5 Summary

This chapter describes the basic concepts that are important for understanding the entire research study as it covers fundamental concepts necessary to grasp the topic. It explains Natural Language Processing (NLP), which combines linguistics and artificial intelligence to help computers understand human language. The chapter also delves into BERT, a groundbreaking model in NLP that uses deep neural networks. It provides a step-by-step breakdown of BERT's operation, including attention mechanisms, transformer architecture, and training processes. By exploring these topics comprehensively, readers gain a solid foundation to navigate the subsequent sections of the study with clarity and understanding, enabling them to appreciate the methodology and findings more effectively. The upcoming chapter outlines the research work that is closely linked to the research carried out in this thesis.

Chapter 3

Literature Review

Extractive Text Summarization has been extensively researched in Natural Language Processing, with broad applications in various domains, including biomedical research, digital humanities, news media, and social media analysis. Existing literature has proposed numerous methodologies for generating document summaries. To provide a comprehensive background and rationale for our proposed approach, we systematically categorize related works based on the methodologies that are utilized for extractive text summarization. Table 3.1 summarizes the literature on extractive summarization approaches.

Author (s)	**Paper Description**	**Key Features**
	Statistical-Based Methods	
[27]	A process for automatically summarizing a text that uses multivariate statistical techniques and is divided into two stages: training and testing. The system is trained using manual summaries in order to create a summarizer that works similarly to a human.	Utilization of multivariate statistical methods to create a summarizer that functions like a human.

[28]	A statistical methodology that is based on extracting numerous aspects and verifying the accuracy using the BBC Sports Article dataset and sports articles from various newspapers is proposed.	A statistical technique is used to extract variables such TF-ISF, sentence length, position, sentence-to-sentence coherence, proper nouns, and pronouns.
Graph-Based Methods		
[29]	A new method that makes use of the Maximum Independent Set and the text-processing software KUSH for extractive summarization of text documents.	By locating and deleting the nodes on the graph that make up the Independent Set, semantic coherence across phrases is preserved.
[30]	An approach to graph-based summarization that uses topic modelling and semantic measures to give edges of the graph weights based on how similar the sentences are to one another and how similar they are to the rest of the document.	By adding topic modelling and semantic measures to give weights to edges of the graph based on sentence similarity and similarity to the entire document, improved summary quality has been achieved.
[31]	A heterogeneous graph-based neural network for extractive summarization (HETERSUMGRAPH) that introduces semantic nodes with various granularities and document nodes to capture cross-sentence interactions.	The addition of various node types to graph-based neural networks enhances cross-sentence relationships and expands their capacity for multi-document summarization.

[32]	Suggested a method that ranks sentences according to their significance and relevance to the entire text using graph-based approaches, enabling the automatic extraction of the most informative sentences.	Using a graph-based ranking algorithm, an unsupervised method for automatic sentence extraction is shown.
[33]	A method of extractive summarization based on a graph reduction technique known as the triangle counting approach, which consists of three major phases: graph representation, triangle construction, and bit vector representation for the triangle's nodes, to create the summary based on bit vector values.	Using the graph reduction technique to improve the extractive summarization process.
	Topic-Based Methods	
[34]	Shows how employing topic-based representations might improve the effectiveness of extractive summarising, demonstrating the potential benefits of using probabilistic topic models in extractive summarization techniques.	Using the Term Frequency Inverse Document Frequency (TF-IDF) value to represent a sentence.
[35]	A topic modelling approach for extractive lexical knowledge-rich text summarization was proposed.	Applied a smoothing technique and four separate variations with various sentence weighting techniques.

[36]	The use of topic modelling to identify latent topics in text documents and clustering the content to produce extractive summaries for each cluster is offered as a novel method for summarising text documents.	Used clustering technique based on latent topics.
Clustering-Based Methods		
[37]	Performed sentence clustering by ranking each one's importance according to three criteria: keywords, term scores, and average cosine similarity to the content.	Implementation of partitioning-based clustering methods.
[38]	Used a fitness function that takes both redundancy and coverage into account to employ a genetic clustering strategy that groups sentences based on the similarity of text topics.	Application of the genetic clustering algorithm for summarization task.
[39]	In order to enhance the effectiveness of summarising, a clustering-based summarization model was created, and related difficulties such as sentence clustering, cluster ordering, and representative sentence selection were addressed.	Data mining document clustering methods were used.
Fuzzy Logic-Based Methods		

[40]	Use of fuzzy inference system to score each sentence by taking into account a number of characteristics, including the title feature, sentence length, term weight, etc. and producing a ranking based on the weighted average of all the features.	Used a fuzzy inference algorithm to assign a score to each sentence.
[41]	Shows how fuzzy logic can be used to extract sentences for summarization and how this can increase the efficacy and accuracy of the process.	For each sentence, features were extracted that included the sentence's length, placement, similarity to other sentences, etc.
[42]	To provide thorough summaries, this method combines graph-based text representation, fuzzy logic for summary optimization , and sentence clustering for comprehensive summaries.	using a graph model to extract features, clustering words to identify textual subtopics before generating the summary, and using fuzzy logic (FL) to maximise summary creation.
[43]	A novel extractive automatic text summarizer that makes use of a variety of strategies, including fuzzy logic, bushy paths, and wordnet synonyms, is shown. It bases its argument on the concept that the summary should contain all of the sentences deemed significant by the three approaches.	Usage of a set of fuzzy IF-THEN rules, a bushy path, wordnet synonyms, and fuzzy logic.

[44]	A method that was tested on 30 news documents and uses fuzzy logic and feature-based extraction to find key sentences increased summary quality and may be expanded to multi-document summarization.	The application of fuzzy logic and feature-based extraction, with the automatic selection of fuzzy inference rules based on the input news category.
Neural Network-Based Methods		
[45]	To increase the quality of the summary, a text summarization technique using genetic algorithm and pre-trained neural networks was proposed.	Combined use of neural network and genetic algorithm.
[46]	In order to enhance phrase representation and document summary, a unique extractive summarization model that makes use of a graph neural network and cooperative learning of latent topics is developed.	Relationships between sentences and the whole document were captured.
[47]	The system uses DLMNN to classify the sentences as informative or non-informative after pre-processing the input document, extracting and selecting key characteristics, computing support and confidence measures, and entropy scores.	By examining entropy levels, Deep Learning Modifier Neural Network (DLMNN) is used to extract important information from documents.

[48]	SummaRuNNer, a recurrent neural network-based tool for extracting significant sentences for summary, was introduced.	By using abstractive summaries rather than extractive labels, a novel approach to training an extractive summarization model is put forth.
BERT-Based Methods		
[49]	Described how to develop BERTSUM, a cutting-edge extractive summarization system, using BERT, a pre-trained transformer model.	Testing out various summarization layers that can be used with BERT.
[50]	For multi-document summarization, a unique hybrid framework termed HEATS was presented.	N-Gram model and RNN-LSTMCNN deep learning combined to produce effective summaries.
[51]	With a focus on sentence embedding and topic modelling for better semantic representation and sentence selection techniques for improved summarization performance, the proposed automatic text summarization system uses Sentence-BERT (SBERT) and density peaks clustering to improve extractive text summarization.	Sentence-BERT was used to improve sentence embeddings and topic modelling to improve the summarization process.

[52]	The "lecture summarization service" is a Python-based service that uses the BERT model for embeddings and the K-Means clustering method to extract key sentences to generate a summary.	Utilized K-means clustering followed by BERT.
[53]	For better performance, an unsupervised summarization method based on BERT transfer learning and fine-tuning was proposed.	Transfer learning from a fine-tuned BERT is used for multiple tasks.

Table 3.1: State-of-the-Art Approcahes for Extractive Text Summarization

3.1 Statistical-Based Methods

These techniques involve performing statistical analysis on a set of features in order to identify important sentences from the original text. According to [54], the most important sentence is determined by factors such as favourable positioning or frequency. The application of the multivariate statistical technique by [27] for the task of summarization produces a model that imitates how a human would summarize the text, offering a more human-like interaction between phrases, leading to a more accurate and informative summary. In order to produce a short and accurate summary, [28] presents a statistical method for extracting text summarization of sports publications.

3.2 Graph-Based Methods

These approaches generate a graphical representation of the text, where each sentence is depicted as a node, and the connections between sentences are depicted as edges. The algorithm then identifies the most important nodes, which correspond to the most informative sentences for inclusion in the summary [4] . The authors of [29] introduced a unique approach for extractive summarization based on the Maximum Independent Set and the KUSH text processing tool. The method entails locating and eliminating phrases from the summary that correspond to nodes in the independent collection. This strategy is based on the notion of recognising statements that are not mutually reliant in order to construct a more logical and informative summary. Another graph-based summarization method that takes into account how related phrases are to one another and to the entire document is suggested by [30]. Similarity between phrases and similarity to the topics of the entire document are the two attributes used by the technique to apply weights to the edges of the graph. The links between phrases can be captured with this method, producing a summary that is more accurate and instructive. HETERSUMGRAPH is a technique for extractive document summarization that is based on a heterogeneous graph-based neural network and was introduced by [31]. The network consists of many kinds of semantic nodes to record cross-sentence relationships. The links between phrases can be captured with this method, producing a summary that is more accurate and instructive. The technique is adaptable to include summarising many documents. Unsupervised automatic sentence extraction using graph-based ranking algorithms has been proposed in [32]. On benchmarks for text summarization, it can perform better. The triangle counting method makes use of a graph-based representation of the text and is able to recognize and extract important facts and ideas from the text, producing a more precise and thorough summary. Triangle counting is the foundation of a reduction strategy proposed in [33] that aims to enhance the effectiveness of extractive summarization. This strategy is based on the notion of condensing the graph representation of the text in order to isolate the key phrases and produce a more precise and educational summary.

3.3 Topic-Based Methods

Topic-based approaches for extractive text summarization entail selecting the most pertinent sentences based on their semantic closeness to the topic of the document, as demonstrated in the study by [55] on using latent semantic analysis for summarization. The study by [34] investigates the effects of probabilistic topic model-based word representations on extractive summarization that is sentence-based. The goal of summary extraction is formulated as a binary classification issue, and several machine learning techniques are tested to investigate various scenarios. For Hindi novels and stories, [35] suggested a extractive lexical knowledge-rich topic modelling text summarization approach. To overcome the lack of an existing corpus, they built four alternative variations employing various sentence weighting systems and produced a corpus of Hindi novels and stories. A technique is proposed in [36], in which topic modelling is used to determine the major topics in a document to be summarized, the text is then split into clusters based on those topics, and the summaries of each cluster are merged to generate the document's final summary.

3.4 Clustering-Based Methods

These methods group similar sentences together and then select representative sentences from each cluster, as demonstrated in the study by [56] on using cluster-based summarization for multi-document summarization. The authors of [57] demonstrated how to apply k-means clustering, word frequency-inverse document frequency, and tokenization techniques for summarization tasks.In [37], a clustering method is presented that evaluates the importance of each phrase based on three criteria: term score, keywords, and average cosine similarity. The summary is generated by selecting phrases using both k-means and fuzzy C-means techniques, employing two similarity calculation methods. The results are obtained for compression levels ranging from 20 to 60 percent. SENCLUS is a new technique for extractive text summarization that was introduced in [38]. It makes use of the evolutionary clustering method, the SENCLUS sentence grouping algorithm, and a fitness function that accounts for both redundancy and coverage. The most important sentences for each topic are then chosen to be included in the extracted summary using a scoring system. In [39], the application of K-means clustering to summary generation and the impact of cluster size on summary quality are discussed. To enhance the performance of summarization, they created a clustering-based technique and dealt with associated issues.

3.5 Fuzzy Logic-Based Methods

Fuzzy logic-based approaches for text summarization extraction utilize fuzzy logic, a reasoning method that resembles human thinking and offers a more intricate way of representing sentence relevance [58]. A technique for extractive summarization utilizing a fuzzy inference system is proposed in [40]. It scores and ranks phrases for inclusion in the summary depending on several properties such as title, length, etc. The extraction strategy for text summarization is focused in [41]. Specifically, sentence selection by utilising sentence weighting, identification of significant features, and proposed text summarization based on fuzzy logic to improve the quality of the summary were covered. In order to maximize the development of summaries, automatic text summarization method is presented in [42]. There are five stages in this extractive method: pre-processing, text representation, feature extraction, sentence grouping, and sentence ranking. It can be used for single or many documents. The most importatnt sentences from a text are identified and extracted using a combination of different strategies in the extractive automatic text summarization method introduced in [43]. To assess the significance of a statement, the method uses wordnet synonyms, the bushy path algorithm, and fuzzy logic. The extraction of a set of attributes from each sentence includes the usage of fuzzy IF-THEN rules as well as features like sentence length, position, centrality, and numerical information. Utilizing wordnet synonyms also accounts for the semantics of the text. In order to extract the important sentences from a text, [44] combined feature-based extraction with fuzzy logic. The findings showed that using fuzzy logic to text summarization enhances the quality of the summaries as a whole. News stories that were categorised into categories like sports, politics, and weather made up the input papers. The system also incorporates an automatic selection of fuzzy inference rules based on the kind of news being summarised in order to provide the best summaries.

3.6 Neural Network-Based Methods

Deep learning models, such as convolutional and recurrent neural networks, are used in neural network-based approaches for extractive text summarization to automatically learn the relevance of phrases from the input text [59].For extractive summarization of single documents, an approach that combines neural networks and evolutionary algorithms is proposed in [45]. The genetic algorithm is employed to optimize this fitness function and extract the most relevant sentences for the summary. The technique uses a fitness function, defined by a neural network, to evaluate the quality of the summary based on properties such as theme similarity, cohesion, sentiment, readability, and more. In [46], authors used a graph neural network (GNN) to record the relationships between phrases in a document and a joint neural topic model (NTM) to find and incorporate latent topics in the text in order to increase the efficacy of extractive text summarization. In order to automatically shrink the size of text documents while maintaining crucial information, [47] introduces a novel method for text summarizing that makes use of a deep learning classifier. The classifier goes through several processes, such as pre-processing the input text, extracting features, choosing the most pertinent features, calculating entropy values, and categorizing the sentences into informative and non-informative groups. According to their entropy values, the most informative sentences are chosen to create the final summary. A model that is based on a simple recurrent network and allows for interpretable visualization of its decisions is proposed in [48]. A novel training strategy that allows extractive models to be trained using abstractive summaries is also described.

3.7 BERT-Based Methods:

BERT-based approaches for extractive text summarization employ pre-trained transformer models to build sentence embeddings and choose the most informative phrases for the summary [49].The authors of [49] suggested a new model called BERTSUM that applies extractive summarization using a pre-trained transformer model called BERT. It was discovered through tests on two sizable datasets that BERTSUM with inter-sentence transformer layers outperformed the others. A similar technique for text summarization utilizing a hybrid framework called HEATS was described by [50], which combines the N-gram model with deep learning models RNN-LSTMCNN for improved performance in terms of syntactic and semantic coherence. Sentence-BERT (SBERT) for sentence embedding and topic modelling, together with cluster-based density peaks clustering for sentence selection, are used in [51] to propose an enhanced automatic text summarization system. The "lecture summarizing service," a python-based RESTful service, was introduced in [52] and uses the BERT model and K-Means clustering to determine the most crucial sentences from a lecture and then displays the summary to the user. Last but not least, [53] presented a fresh approach to extractive multi-document summarization that makes use of transfer learning from the BERT sentence embedding model and is improved on supervised intermediate tasks from GLUE benchmark datasets using both single-task and multi-task learning techniques.

3.8 Our Proposed Approach in State-of-the-Art

Automated summarization has been a long-standing research topic, and various methods have been developed to extract essential information from text documents. Although statistical-based methods have been commonly utilized for extractive summarization, recent advancements in deep learning have demonstrated promising results in capturing the semantic and contextual meaning of text. In this study, we propose a novel approach that employs a transformer-based model known as BERT to improve the summarization of long scholarly documents. Our proposed approach offers several benefits over existing methods. Firstly, it outperforms traditional statistical-based and other non-deep learning-based methods in capturing the contextual and semantic meaning of the document, owing to its use of BERT as the underlying model. Secondly, by dividing the document into smaller chunks, we overcome the input length limitation of BERT, which is a common issue in transformer-based summarization methods. Additionally, our use of an encoder-decoder model on top of the BERT sentence embeddings generates more coherent and readable summaries, significantly improving over previous methods. Lastly, we demonstrate that our approach outperforms several state-of-the-art models for extractive summarization on two scholarly datasets, arXiv and PubMed, not only surpassing statistical-based methods but also outperforming several deep learning-based methods.

3.9 Summary

This chapter provides a comprehensive overview of the existing body of work in the domain of extractive text summarization. It encompasses an extensive array of methods employed in this field, including statistical-based approaches, graph-based techniques, topic modeling-based methodologies, neural network-based models, and BERT-based methods, among others. The chapter not only explores these diverse approaches but also introduces our proposed approach, which represents a significant advancement in the current state-of-the-art. Building upon the foundation laid in this chapter, the subsequent chapter delves into a novel technique that combines BERT and BiGRU architectures for the purpose of extractive text summarization specifically tailored to scholarly articles.

Chapter 4

Design and Methodology

4.1 Datasets

In this section, we will highlight the characteristics and qualities of the two datasets used in this research work.

4.1.1 ArXiv Dataset

The arXiv dataset is a collection of scientific articles and pre-print publications that can be used to do text summarization. In [60], authors identified this dataset as a great resource for text summarization research, providing a broad and comprehensive collection of scientific publications that can be used to train and assess automatic text summarization models. Pre-prints from different fields like physics, mathematics, computer science, and other subjects are included in the arXiv dataset. These papers were deposited in the arXiv repository by their authors in order to share their work with the scientific community prior to publication. The dataset has been divided into training, validation, and test sets, allowing text summarization models to be evaluated. Table 4.1 shows statistics of arXiv dataset.

Table 4.1: Statistics of arXiv dataset.

No. of Documents			**Average No.of Sentences**		**Average No.of Tokens**	
Training	**Validation**	**Test**	**Document**	**Summary**	**Document**	**Summary**
203037	6436	6440	204	5.6	5038	165

Data Fields:

1. **Sentences:** a string that includes the paper's body.
2. **Gold:** a string containing the paper's abstract.
3. **ID:** Unique Id for each paper.

4.1.2 PubMed Dataset

The PubMed dataset is a popular resource in text summarization research and it is also introduced by [60]. It is made up of lenghty scientific papers from the website PubMed.com, with their abstracts serving as reference summaries. This dataset's goal is to provide a broad and complete collection of scientific publications for training and evaluating text summarization models. The articles in this dataset include a wide range of topics consists of scientific articles published in various fields of medicine and life sciences, making it a significant resource for researchers for evaluation of summarization models. The dataset has been divided into training, validation, and test sets, allowing text summarization models to be evaluated. Table 4.2 shows statistics of pubMed dataset.

Table 4.2: Statistics of pubMed dataset.

No. of Documents			Average No.of Sentences		Average No.of Tokens	
Training	**Validation**	**Test**	**Document**	**Summary**	**Document**	**Summary**
119224	6633	6658	88	6.8	3235	205

Data Fields:

1. **Sentences:** a string that includes the paper's body.
2. **Gold:** a string containing the paper's abstract.
3. **ID:** Unique Id for each paper.

In order to make use of the datasets in our experiments, we followed the dataset splits as proposed by [60]. This ensured that we are using the same data splits as previous research in the field, allowing for a fair and consistent evaluation of our models.

4.2 Proposed Metodology

The proposed solution for the task of text summarization builds on previous work that employs BiGRU on top of BERT for tasks such as entity recognition, text classification, and question-answering, as demonstrated in [61, 62, 63, 64]. In this work, we present an architecture that combines BERT with an attention-based Encoder-Decoder framework to capture global context of whole document, as illustrated in Figure 4.1.

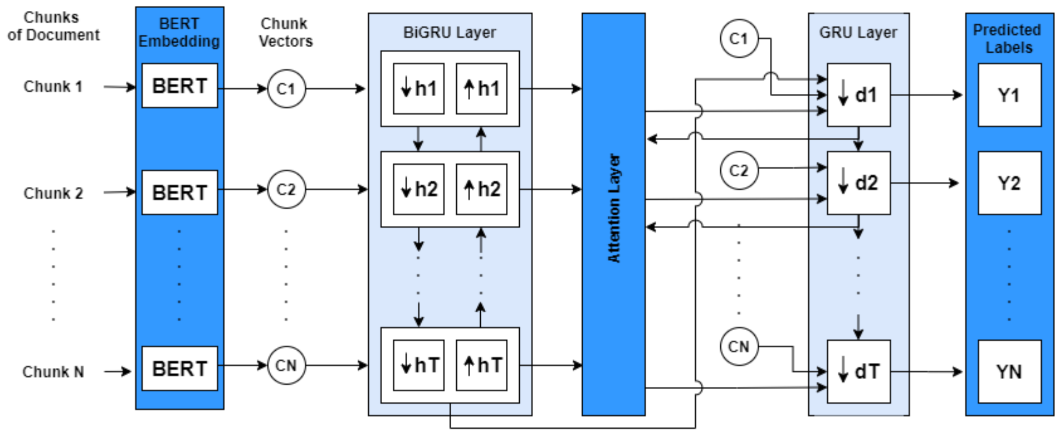


Figure 4.1: Proposed model architecture: Document is chunked into sentences and passed through BERT to obtain contextual embeddings. Encoder model propagates information between chunks, while a Decoder generates sentence labels using attention.

Given an arbitrary-length document D, we first divide it into several chunks, each containing one sentence, and apply BERT to each chunk to obtain context-aware embeddings. Next, an Encoder model is used to propagate information between the chunks, and a Decoder generates a weighted source vector using an attention layer to predict the label for each sentence. The model performs binary classification, where a label of 1 indicates that a sentence is a viable candidate for summary and 0 indicates that the sentence is not a suitable candidate. The details of each module are explained in the following subsections.

4.2.1 Chunk Vector Generation

To tackle the challenge of processing long documents for text summarization, we divide the input document into smaller, manageable chunks, with each chunk consisting of a single sentence. This approach helps us avoid exceeding BERT's input limit and ensures that each sentence can be processed effectively.

We assign labels to each chunk, labeling them as $chunk_1$, $chunk_2$, and so on, up to $chunk_N$. Each sentence in a chunk has n tokens, making it easier for our proposed model to handle the data. To follow BERT conventions, we append the special tokens [CLS] and [SEP] to the beginning and end of each chunk. These tokens help BERT understand the context of the sentence and ensure that the model is processing the data effectively.

Next, we process all chunks through a pre-trained BERT model, generating contextual embeddings for each chunk. These embeddings represent each sentence as a vector and capture the nuances of the sentence's meaning and context. These chunk vectors are then used as input for the model's subsequent layers, enabling the model to learn from the input data and make accurate predictions.

Our approach to chunk vector generation provides a scalable and effective way to process long documents for text summarization.

4.2.2 Information Propagation

In the previous subsection, we discussed how BERT-generated chunk vectors are used to represent contextual information within individual sentence chunks. However, these vectors lack global information, and therefore, an additional step is required to propagate information between sentence chunks. In this subsection, we describe the Information Propagation module, which utilizes a Bidirectional Gated Recurrent Unit (BiGRU) to propagate information between chunks.

The BiGRU is a type of Recurrent Neural Network (RNN) that was first proposed by [65]. It consists of two GRUs, one processing the sentence chunks in the forward direction, and the other processing them in the backward direction. The BiGRU hidden states are computed using Equations 4.2.1 and 4.2.2, which generate two context vectors, one for the forward pass and one for the backward pass. The final concatenated state is computed using Equation 4.2.3. This concatenated vector, $\overrightarrow{z}$, contains global information from all sentence chunks.

$$\overrightarrow{h}_t = EGRU(\overrightarrow{c}_t, \overrightarrow{h}_{t-1}) \tag{4.2.1}$$

$$\overleftarrow{h}_t = EGRU(\overleftarrow{c}_t, \overleftarrow{h}_{t-1}) \tag{4.2.2}$$

where c_t is the BERT-generated input chunk vector and h_t is the hidden vector at time t. These equations will produce two context vectors, one for the forward pass and one for the backward pass. BiGRU's final concatenated state is given by

$$\overrightarrow{z} = [\overrightarrow{h}_T, \overleftarrow{h}_T] \tag{4.2.3}$$

The final hidden states from the forward and backward passes are shown here as $\overrightarrow{h}_T$ and $\overleftarrow{h}_T$, respectively. The concatenation operation is indicated by the square brackets [].

We use a Gated Recurrent Unit (GRU) as the decoder in our model. However, since the GRU is not bidirectional, it can only take one context vector as input. To solve this problem, we concatenate the two context vectors generated by the forward and backward passes. The concatenation is achieved by passing the vectors through a linear layer 1 and using a tanh activation function, as shown in Equation 4.2.4. The resulting vector, d_0, is used as the initial hidden state for the GRU decoder. The Information Propagation module plays a crucial role in the model's ability to capture global information and propagate it throughout the sentence chunks.

$$z = \tanh(l(\overrightarrow{h}_T, \overleftarrow{h}_T)) = d_0 \tag{4.2.4}$$

4.2.3 Attention Mechanism

The attention layer takes all hidden states of the BiGRU as well as the prior hidden state of the GRU to produce an attention vector, denoted as a_t. When predicting the label for each sentence, this vector represents the sentence in the document that should be paid the most attention.

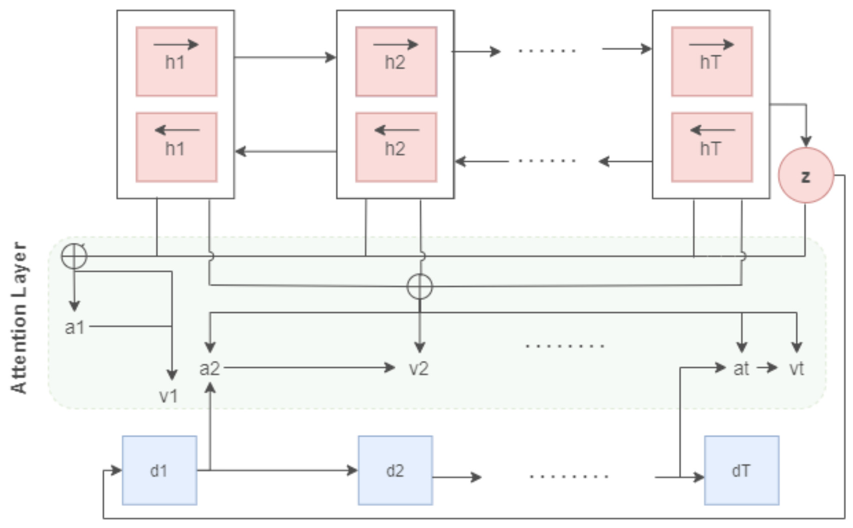


Figure 4.2: Attention Layer: Produces an attention vector (at) using BiGRU and GRU hidden states. The vector identifies the sentence in the document that requires the most attention during label prediction.

The following equation is used to determine the attention vector:

$$\hat{a}_t = uE_t \tag{4.2.5}$$

where E is the energy between the GRU's previous hidden state, d_{t-1}, and all BiGRU's hidden states, H. The hidden states are concatenated and sent through a linear layer termed as attn, together with the tanh activation function, to determine this energy. The variable u specifies the weights for the weighted total energy of all hidden states of the BiGRU, indicating how much attention each sentence in the source text should get.

The following equation shows energy calculation:

$$E_t = tanh(attn(d_{t-1}, H)) \tag{4.2.6}$$

Finally, the attention vector is sent through a softmax layer to guarantee that all values are between 0 and 1 and that the total of all vector elements is 1.

$$a_t = softmax(\hat{a}_t) \tag{4.2.7}$$

The attention vector is then utilized to generate a weighted source vector, represented as v_t, with weights equal to the weighted sum of the BiGRU hidden states, H, and the attention vector, a_t.

$$v_t = a_t H \tag{4.2.8}$$

4.2.4 Label Prediction and Summary Generation

GRU's previous hidden state, i-e d_{t-1}, and the embedded chunk vector c_t concatenated with the weighted source vector are then passed to GRU.

$$d_t = DGRU(c_t, v_t, d_{t-1}) \tag{4.2.9}$$

To predict the label for each sentence, the concatenation of c_t, v_t, and d_t is processed through a softmax and argmax.

$$\hat{y}_t = softmax(c_t, d_t, v_t) \tag{4.2.10}$$

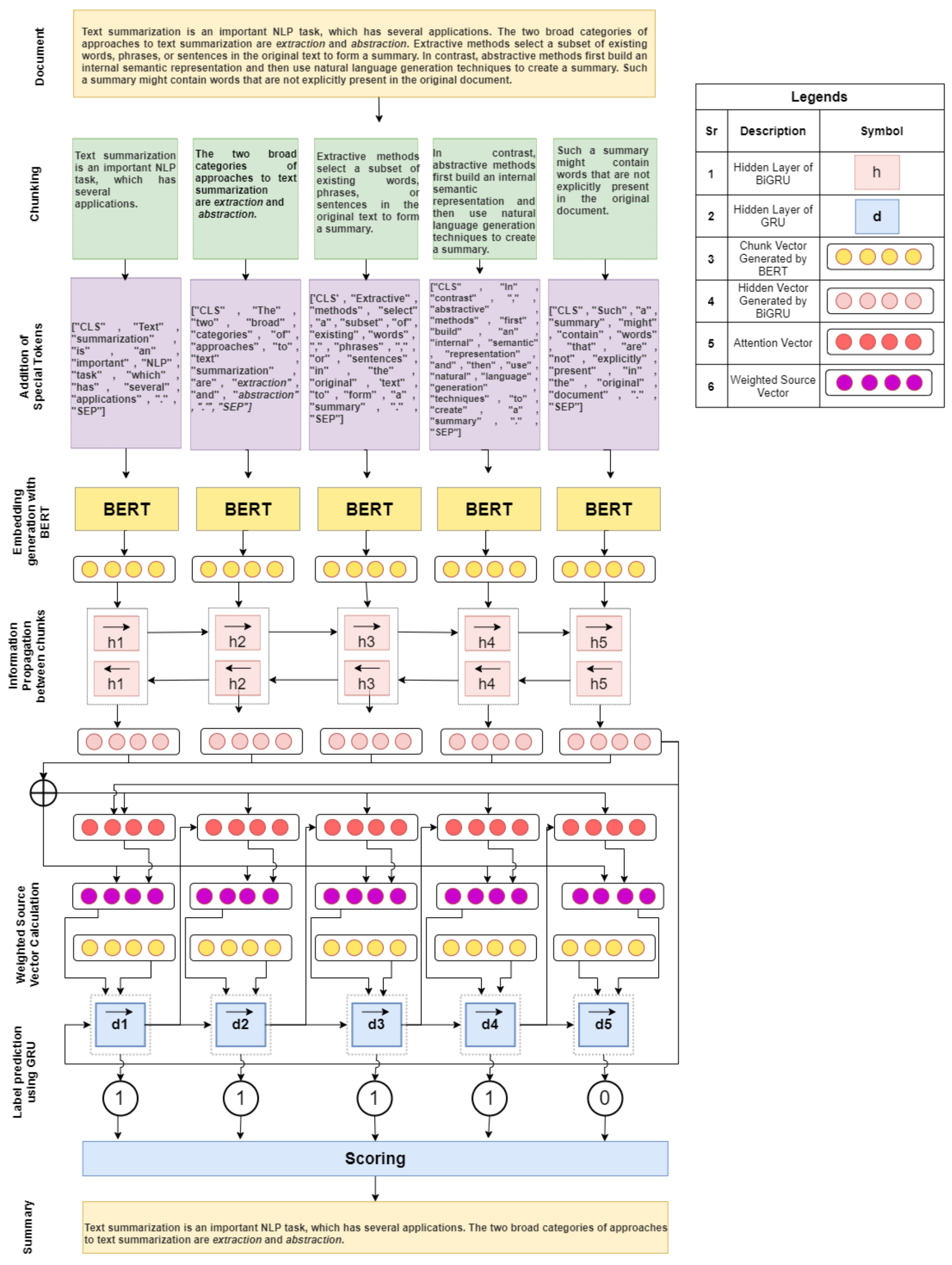
$$y_t = argmax(\hat{y}_t) \tag{4.2.11}$$

where y_t is the predicted label for the chunk t in a document.

Finally, the top K sentences labeled as "1" in the document, in order of appearance, are chosen to be included in the summary. This is accomplished by choosing the sentences that are most likely to be pertinent to the summary using the labels that are predicted by GRU.

4.3 Materializing the Proposed Methodology: Flow and Algorithms

The schematic flow of the proposed BERT-based model for extractive text summarization is depicted in detail in Fig. 4.3. This flow serves as a visual representation of the various steps involved in the summarization process and provides a useful framework for implementing the proposed model.


Document
Chunking
Addition of Special Tokens
Embedding generation with BERT
Information Propagation between chunks
Weighted Source Vector Calculation
Label prediction using GRU
Summary
BERT
h1
h2
h3
h4
h5
d1
d2
d3
d4
d5
1
1
1
1
0
Scoring
Legends
Sr
Description
Symbol
1 Hidden Layer of BiGRU h
2 Hidden Layer of GRU d
3 Chunk Vector Generated by BERT
4 Hidden Vector Generated by BiGRU
5 Attention Vector
6 Weighted Source Vector

Text summarization is an important NLP task, which has several applications. The two broad categories of approaches to text summarization are *extraction* and *abstraction.* Extractive methods select a subset of existing words, phrases, or sentences in the original text to form a summary. In contrast, abstractive methods first build an internal semantic representation and then use natural language generation techniques to create a summary. Such a summary might contain words that are not explicitly present in the original document.

Text summarization is an important NLP task, which has several applications.

The two broad categories approaches to text summarization are *extraction* and *abstraction.*

Extractive methods select a subset of existing words, phrases, or sentences in the original text to form a summary.

In contrast, abstractive methods first build an internal semantic representation and then use natural language generation techniques to create a summary.

Such a summary might contain words that are not explicitly present in the original document.

["CLS" , "Text" , "summarization" , "is" , "an" , "important" , "NLP" , "task" , "which" , "has" , "several" , "applications" , "." , "SEP"]

["CLS" , "The" , "two" , "broad" , "categories" , "of" , "approaches" , "to" , "text" , "summarization" , "are" , "*extraction*" , "and" , "*abstraction*" , "." , "SEP"]

["CLS" , "Extractive" , "methods" , "select" , "a" , "subset" , "of" , "existing" , "words" , "," , "phrases" , "," , "or" , "sentences" , "in" , "the" , "original" , "text" , "to" , "form" , "a" , "summary" , "." , "SEP"]

["CLS" , "In" , "contrast" , "," , "abstractive" , "methods" , "first" , "build" , "an" , "internal" , "semantic" , "representation" , "and" , "then" , "use" , "natural" , "language" , "generation" , "techniques" , "to" , "create" , "a" , "summary" , "." , "SEP"]

["CLS" , "Such" , "a" , "summary" , "might" , "contain" , "words" , "that" , "are" , "not" , "explicitly" , "present" , "in" , "the" , "original" , "document" , "." , "SEP"]

Text summarization is an important NLP task, which has several applications. The two broad categories of approaches to text summarization are *extraction* and *abstraction.*

Figure 4.3: Proposed BERT-based model for extractive text summarization. A schematic flow diagram illustrating the steps involved in the summarization process and serving as a framework for implementing the model.

The suggested system's implementation can be carried out in the following steps:

Algorithm 1 Chunks Embedding Generation

```
  Input: Document
  Output: Vectors for each sentence
1- Divide document into multiple chunks where each chunk contains one sentence
2- For each chunk in Document
     Generate Embedding Vector c using BERT.
  End
3- Return Concatenation of all vectors generated by BERT
```

Algorithm 2 Information Propagation

```
  Input: Concatenation of all vectors generated by BERT
  Output: Annotations
1- Pass all vectors generated by BERT as input to BiGRU
2- Calculate forward hidden states and backward hidden states at each time step using
     $\overrightarrow{h}_t \leftarrow EGRU(\overrightarrow{c}_t, \overrightarrow{h}_{t-1})$
     $\overleftarrow{h}_t \leftarrow EGRU(\overleftarrow{c}_t, \overleftarrow{h}_{t-1})$
3- Concatenate forward hidden states and backward hidden states at each time step using
     $z \leftarrow \tanh(l(\overrightarrow{h}_t, \overleftarrow{h}_t))$
4- Follow the step 3 to concatenate final hidden states of forward and backward pass to set as initial hidden state for GRU
```

Algorithm 3 Attention Mechanism

```
  Input: All Hidden States of BiGRU and Last hidden state of GRU
  Output: Attention Vector
1- Calculate energy between all hidden states of BiGRU and last hidden state of GRU
     $\hat{E}_t \leftarrow attn(d_{t-1}, H)$
     $E_t \leftarrow tanh(\hat{E}_t)$
2- Generate attention vector
     $\hat{a}_t \leftarrow uE_t$
     $a_t \leftarrow softmax(\hat{a}_t)$
```

Algorithm 4 Summary Generation

```
  Input: Attention Vector, Embedded Chunk Vector
  Output: Summary
1- Calculate weighted source vector by using Hidden states of BiGRU and consider attention vector as weights
    $v_t \leftarrow a_t H$
2- Pass $v_t$, $c_t$ and $d_{t-1}$ to GRU
    $d_t \leftarrow DGRU(c_t, v_t, d_{t1})$
3- Now pass the $c_t$, $d_t$ and $v_t$ through softmax activation function to generate labels for each chunk
    $\hat{y}_t \leftarrow softmax(c_t, d_t, v_t)$
4- Apply argmax
    $y_t \leftarrow argmax(\hat{y}_t)$
5- For summary, select top K sentences that are labeled as '1'
```

In step 1 of Algorithm 1, the input document is broken into smaller, more manageable chunks. This is a vital stage since it helps to break down a potentially enormous and complex document into smaller, more easily processable chunks. This allows the model that follow to process the information more efficiently without being overwhelmed by the sheer volume of data.

In step 2, these chunks are transferred one by one to the BERT model, a state-of-the-art language representation model. The BERT model use these chunks to construct embeddings, which are numerical representations of the content of each chunk. These embeddings capture the semantic and syntactic information available in each chunk, allowing them to be processed meaningfully. This is crucial because it provides for a more accurate depiction of the chunk's meaning and context, making it easier for the algorithms that follow to interpret and process it.

In step 3, after all of the embeddings have been produced, they are concatenated and delivered to Algorithm 2. Algorithm 2 is in charge of propagating information between chunks and locating annotations related to the document's content. It identifies annotations by using the embeddings as inputs. This phase is important since it contributes to build a complete understanting of the document's content and can provide vaueable context that would be lost otherwise.

These annotations are subsequently submitted to Algorithm 3, which employs them in the generation of an attention vector. This vector shows the relative importance of each chunk in the document and aids in highlighting the most significant content. Algorithm 3 does this by quickly identifying which parts of the document are most relevant, making it easy to find the key sentences.

Finally, Algorithm 3 sends the attention vector it generated to Algorithm 4, which utilizes it to generate a summary of the document.

In conclusion, the entire process is consists of breaking down the document into chunks, generating embeddings, propagating information, generating an attention vector, and producing a summary in order to give a comprehensive and efficient means of processing huge and complex documents.

4.4 Implementation Details

4.4.1 Binary Label Generation

Annotated data must be used to train machine learning models for extractive summarization. In our case, the annotations must take the form of binary labels at the sentence level, indicating whether or not each sentence in a text document should be included in the final summary. These annotations are essential in teaching the model how to recognize the key phrases in a document and produce a consice summary that accurately represents its main ideas.

The approach given by [66] is used to compute these annotations. This method entails categorizing each sentence in the text by comparing the extracted summary's ROUGE-1 score to the gold-standard summary associated with each article.

It should be noted that these labels are only utilized during the training phase and not during the evaluation phase. The extracted summaries are evaluated by comparing them to the gold-standard summaries provided in the datasets. This ensures that the model is learning to provide high-quality summaries that capture the most significant information from the original text.

4.4.2 LEAD Model

The LEAD approach is a straightforward yet effective method for generating a summary of a given article by picking the starting sentences. This method provides as a baseline against which the model developed in this research can be compared.

Previous research findings, such as [49], suggested that LEAD can serve as a strong foundation for extractive text summarization tasks. The authors had a clear starting point for measuring the performance of their models by utilizing LEAD as a baseline. This gave the authors significant insights into the strengths and shortcomings of the various models and allowed them to compare their findings to earlier studies.

Overall, the use of LEAD as a baseline is a critical component of the evaluation process, providing a strong foundation for the comparison and analysis of the proposed model developed in this work.

4.4.3 Oracle Model

In extractive text summarization, the Oracle model serves as a standard for evaluating the performance of other models. It limits their performance since the models cannot outperform the Oracle summary.

Oracle summaries can be generated using one of two algorithms: the greedy algorithm or the combination algorithm. The greedy approach calculates and maximizes the ROUGE score between sentences in the gold summary and the article. This process is utilized to generate the oracle summary. To construct the oracle summary, the combination method considers all potential sentence combinations and maximizes the ROUGE score.

However, as [49] points out, the combination technique can result in poor performance, particularly when picking more than three sentences. As a result, we chose to utilize the greedy method throughout the experiments. The greedy method was faster and more efficient, and it produced good results without sacrificing accuracy.

We had a clear range for the ROUGE scores that we intended our model to perform within by using both LEAD and the Oracle model. The model should outperform the LEAD score, but not necessarily the Oracle score, because the Oracle score acted as a performance upper limit.

4.4.4 Baseline Models

1. **BERTSUMEXT:**
   We considered BERTSUMEXT proposed by [67] which is the state-of-the-art model for extractive text summarization as our baseline for comparison with our model. It is an innovation in the field of extractive text summarization that is based on the popular BERT language model. It has been fine-tuned expressly for the purpose of summarization, with an emphasis on optimising its performance for this unique use case. However, one of the difficulties in employing BERT for summarization is its inability to scale to long content. As suggested in the original paper, we only took into account the first 800 tokens because this approach cannot scale to long text.

2. **BERTSUMEXT with Sliding Window:**
   We also implemented BERTSUMEXT with Sliding Window, which is an extension of the BERTSUMEXT architecture designed to accommodate even longer texts. This was accomplished by employing the sliding window approach proposed by [68]. The sliding window method breaks a long text into many fixed-size windows; in our example, we considered 800 tokens in each window and processed each window independently. BERTSUMEXT with Sliding Window overcomes the constraints of the previous BERTSUMEXT approach by dividing the document down into smaller sections, and can now accommodate much longer text, however there is no connection between multiple windows, and global context cannot be captured.

4.4.5 Hyper-parameters

For Proposed Model

In order to reduce memory usage during the training process of our model, the batch size was set to **6**. To optimize the model's parameters, a learning rate of **0.00001** was chosen as an initial value. The model was planned to be trained for 8 epochs, but after seeing that the loss value did not decrease after the 5th epoch, training was halted and the model was trained for only **5** epochs. The Adam optimization algorithm [69] was employed with a drop-out value of **0.5** to further refine the model. This combination of batch size, learning rate, number of epochs, and optimizer assisted in training the model effectively while avoiding overfitting and maintaining a balance between computational efficiency and model accuracy.

For Baseline Models

In our baseline experiments, we began with an existing version of BERTSUMEXT and adapted it to operate with sliding windows. The pre-trained "bert-base-uncased" BERT model and its associated settings were used. The sliding windows were **800** tokens wide, with a **300-token** overlap between them. If a sentence appeared in more than one window, we choose the one with the greatest context and used the [CLS] representation from that window. Finally, we used the Adam optimization technique [69] to fine-tune the model with a learning rate of **0.00001**.

4.4.6 Prediction

Following training, the model was used to generate predictions on the arXiv and pubMed test sets. For each article in the test set, the model generated a label for each sentence. The top K sentences according to their appearance in the article (where K is the number of oracle sentences utilized throughout the training procedure), were then chosen as the article summary. This technique effectively allowed the model to summarize each article in the test set by picking the most significant and relevant sentences based on the scores given during prediction.

4.4.7 Hardware

We used Google Colab Pro+ platform for our model training. Google Colab is a free online platform that lets users build and run Python code directly in their browser. However, due to the free version's limited memory of 12GB, we opted for the paid version, Google Colab Pro+, which provides a broader range of resources and functionality. We got access to additional RAM of up to 54GB and extended runtimes of 24 hours with Google Colab Pro+, compared to the typical 12 hours in the free version. Furthermore, the platform gives access to high-performance GPUs such as Nvidia K80, T4, P4, and P100 in free version and more powerful GPU's in premium version which includes A100 Tensor Core or NVIDIA V100 etc. We were assigned the powerful NVIDIA V100 GPU during our runtime.

4.5 Evaluation Metric

We employed a set of metrics called as Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [1] to assess the performance of our proposed summarization model. ROUGE is a popular tool for comparing the generated summary to reference summaries in order to assess the quality of text summarization systems. The ROUGE package has three major metrics: ROUGE-N, ROUGE-L, and ROUGE-S. ROUGE-N counts the number of n-grams that overlap between the reference summary and the created summary. ROUGE-L looks for the sequence that has the most similarities between the two summaries. ROUGE-S considers consecutive terms shared by both summaries, even if they are separated by other words. We considered ROUGE-N (ROUGE-1 and ROUGE-2) and ROUGE-L from the three metrics for evaluation purpose.

4.6 Summary

In this chapter, we provide a detailed explanation of how we implemented the suggested architecture. We describe the specific steps we followed to make it work effectively. Additionally, we discuss the configuration of the baseline model that we used as a reference for comparing the performance of our architecture. We also give insights into the framework we used for the implementation, explaining its capabilities and how it functions. Moreover, we explore the training hyperparameters, which are settings and values that we carefully selected to make the model learn optimally. By presenting this comprehensive overview, readers can understand the technical aspects of how we implemented the proposed architecture and the important factors we considered to ensure a strong and dependable experimentation process. The subsequent chapter is dedicated to presenting the results obtained through the conducted experiments.

Chapter 5

Results and Discussion

In Table 5.1, we demonstrate our main results on both datasets. In addition to BERT-based baselines that we re-implemented, we conducted a comparative analysis with other techniques, as cited from references [67, 69, 57]. For clarity and conciseness, the table is separated into four sections. The results for the Lead and Oracle methods which serves as a lower and upper bound for ROUGE score are displayed in the first section. The second section focuses on traditional summarization approaches. The third section demonstrates various extractive summarization strategies. Finally, in the fourth section, the suggested model is compared to the baseline models, showing its performance and efficiency in comparison to these approaches.

From Table 5.1, it is worth noticing that our approach demonstrates good performance in the task of summarizing lengthy documents while simultaneously preserving both informativeness, as evaluated by ROUGE-1, and fluency, as evaluated by ROUGE-L, in the generated summaries. In addition, our approach outperforms several previously published state-of-the-art models and BERT-based baselines, including BERTSUMEXT, which truncates the document to first 800 tokens and proves to be less successful due to difficulty in handling long documents. Notably, the sliding window adaptation of BERTSUMEXT, which allows for processing longer documents, demonstrates outcomes that closely align with our approach. However, our method still outperforms this version, highlighting its capacity to capture global context of document, which is essential for producing effective summaries of long documents.

Models	**pubMed**			**arXiv**		
	R1	**R2**	**RL**	**R1**	**R2**	**RL**
Lead	32.61	13.00	24.29	34.12	8.96	21.23
Oracle	56.02	26.67	40.39	52.98	23.45	36.92
LSA	33.89	9.93	29.70	29.91	7.42	25.67
LexRank	39.19	13.89	34.59	33.85	10.73	28.99
Cheng and Lapata(2016)	43.89	18.53	30.17	42.24	15.97	27.88
Match-Sum	41.21	14.91	36.75	40.59	12.98	32.64
SummaRuNNer	43.89	18.78	30.36	42.81	16.52	28.23
Topic-GraphSum	45.95	20.81	33.97	44.03	18.52	32.41
Seq2seq-local and global	44.85	19.70	31.43	43.62	17.36	29.14
SSN-DM	46.73	21.00	34.10	45.03	19.03	32.58
SSN-DM + discourse	46.52	20.94	35.20	44.90	19.06	32.77
BERTSUMEXT	41.82	14.98	37.02	41.43	14.01	35.22
BERTSUMEXT with Sliding Window	44.87	20.21	39.38	43.13	15.48	**35.99**
BERT with Encoder-Decoder (Ours)	**47.01**	**21.25**	**39.68**	**46.73**	**19.42**	35.42

Table 5.1: Comparative analysis of proposed model with previously published Approaches. Here R1, R2 and RL reperesents ROUGE-1, ROUGE-2 and ROUGE-L respectively.

5.1 Analysis

To analyze the impact of sentence positions in the input document, we conducted a study that aimed to identify the distribution of sentence positions in a summary. To do this, we created histograms as shown in Figure 5.1, which illustrate the frequency of sentence positions in both the Oracle summary and the predicted positions generated by various baseline models. Specifically, we examined the performance of our proposed model, as well as BERTSUMEXT and BERTSUMEXT with Sliding Window.

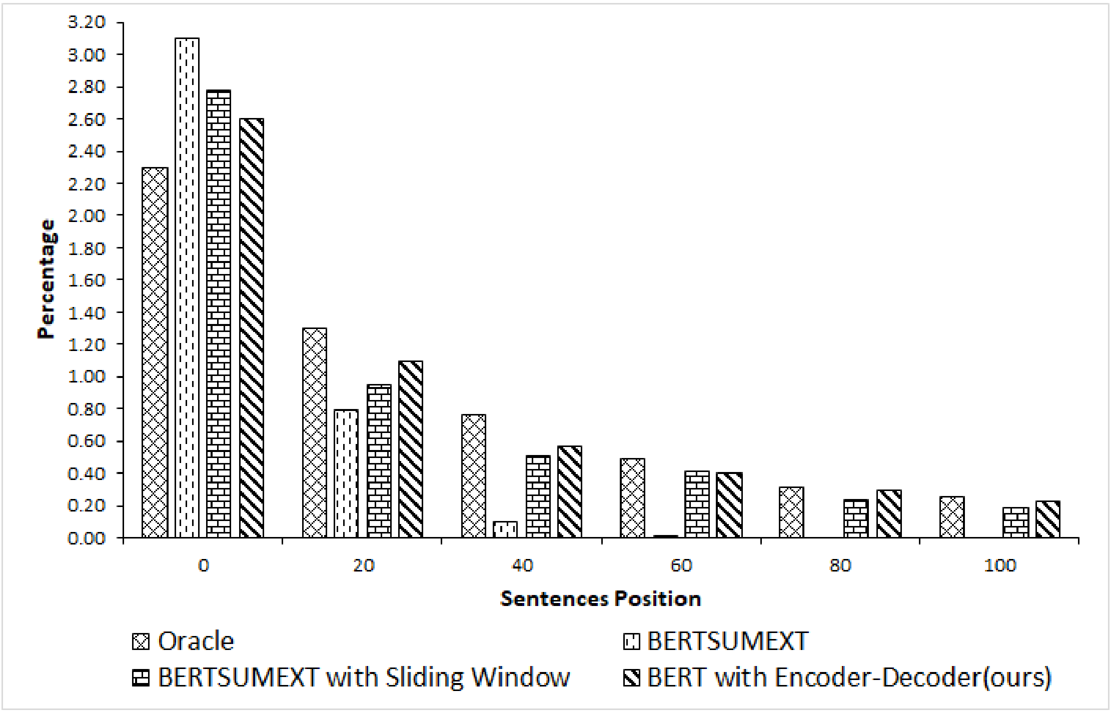


Figure 5.1: Distribution of extracted sentences in the pubMed test dataset based on their position in document. The x-axis represents the linear sentence indices while the y-axis displays the percentage of extracted sentences

Our findings from Figure 5.1 suggest that the most relevant sentences are typically found at the beginning of a document. However, we also observed that the Oracle summary included relevant sentences further down the document, indicating that there is valuable information throughout the entire text. Notably, our proposed model exhibited behavior that was closest to the Oracle, suggesting that it is more effective in capturing the relevant sentences throughout the text.

In contrast, the baseline models tended to prioritize sentences from the document's beginning, while giving less attention to sentences appearing later in the document. This suggests that these models may not be as effective at capturing the full range of relevant information available in the text.

In natural language processing, the performance of text summarization models is often evaluated based on the quality of the summaries they generate, which can be measured using metrics such as Rouge-1. To assess the performance of our proposed models, we conducted experiments on the ArXiv test dataset and compared the results to those of baseline models.

In Figure 5.2, we present the Rouge-1 scores of the different models, with a focus on the word count of the source documents. As the size of the source texts increases, it becomes increasingly difficult for summarization models to accurately capture the most important information and generate high-quality summaries. However, our proposed model demonstrates superior performance, outperforming both BERTSUMEXT and BERTSUMEXT with Sliding Window as the word count of the source document increases.

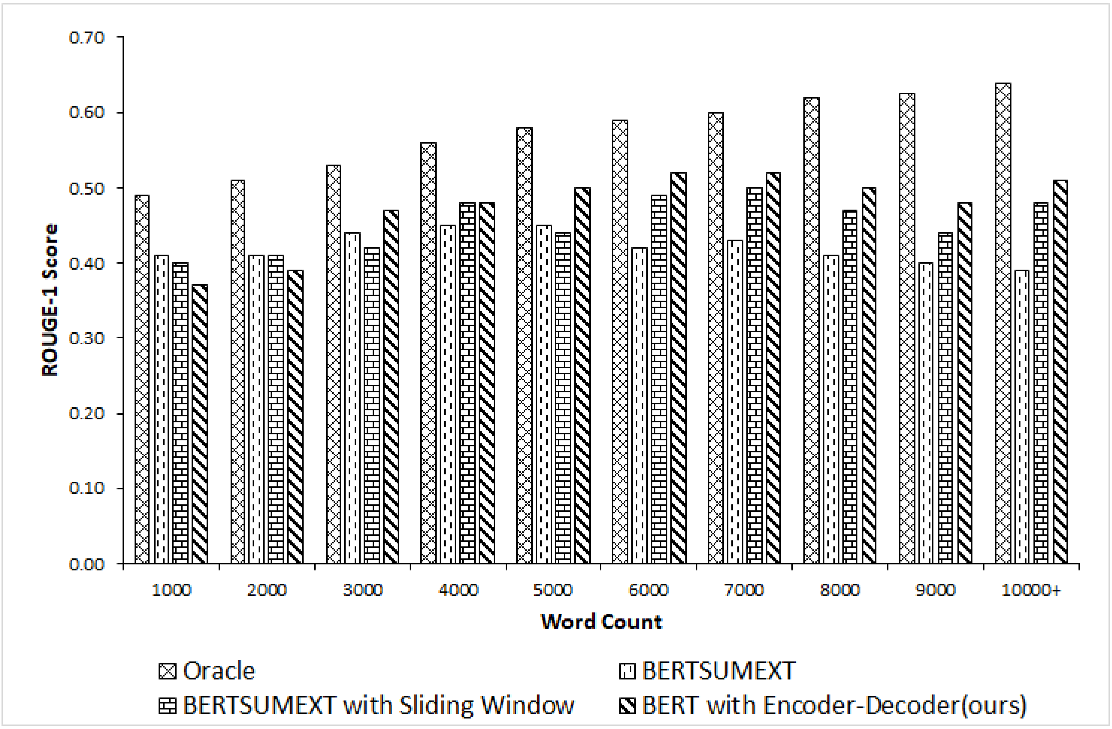


Figure 5.2: The performance evaluation of model on arXiv test set by computing the average ROUGE-F1 score based on the word count of the input document. The document length is represented on the x-axis while the ROUGE Score is displayed on the y-axis.

The results depicted in Figure 5.2 provide strong evidence that our proposed model is capable of effectively and efficiently handling larger texts, consistently generating higher Rouge-1 scores as the word count of the source document increases. This is a significant finding, as it indicates that our model is capable of handling real-world scenarios where the source texts are often lengthy and complex. Overall, our proposed model shows a significant improvement for the extractive summarization of lengthy scholarly documents.

The recall score is an important metric in text summarization evaluation as it measures the proportion of relevant information that the system is able to extract from the original text. In order to obtain a comprehensive evaluation of the system's performance, it is crucial to consider both recall and precision because ROUGE-F measure, which is the harmonic mean of precision and recall, is often used as a fundamental statistic for comparison.

In this context, the proposed model's performance was analyzed based on recall scores by selecting a random sample of three documents from the PubMed dataset, as shown in Table 5.2. The results indicated that the recall scores of the proposed model were substantially higher than expected. This high recall score demonstrated the efficacy of the proposed method in capturing and extracting the most relevant and important information from the original text.

Document	**Rouge-1**			**Rouge-L**		
	Precision	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**
Doc-1	0.6	0.89	0.73	0.5	0.81	0.63
Doc-2	0.5	0.84	0.61	0.5	0.76	0.54
Doc-3	0.5	0.78	0.59	0.4	0.7	0.47

Table 5.2: Rouge-1 and Rouge-L scores for Proposed Model on 3 randomly selected documents from pubMed dataset.

Document	**Rouge-1**			**Rouge-L**		
	Precision	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**
Doc-1	0.45	0.78	0.57	0.38	0.64	0.46
Doc-2	0.47	0.72	0.49	0.38	0.61	0.37
Doc-3	0.42	0.57	0.43	0.31	0.46	0.31

Table 5.3: Rouge-1 and Rouge-L scores for BERTSUMEXT on 3 randomly selected documents from pubMed dataset.

Document	**Rouge-1**			**Rouge-L**		
	Precision	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**
Doc-1	0.51	0.83	0.62	0.42	0.74	0.55
Doc-2	0.47	0.76	0.54	0.41	0.66	0.45
Doc-3	0.43	0.79	0.6	0.34	0.61	0.46

Table 5.4: Rouge-1 and Rouge-L scores for BERTSUMEXT with Sliding Window on 3 randomly selected documents from pubMed dataset.

For comparison, baseline models were also used, as presented in Tables 5.3 and 5.4, which showed that the recall scores of the baseline models were lower than those of the proposed model. The proposed model's higher recall score led to a higher F-measure, indicating superior performance in all aspects. Overall, the analysis of recall scores provided a comprehensive evaluation of the system's ability to extract relevant information from the original text and demonstrated the proposed model's effectiveness in this task.

5.2 Summary

In this chapter, we showcase the results of our proposed model and conduct a thorough comparison with the state-of-the-art model. The obtained results are presented in detail, accompanied by a comprehensive discussion. We analyze the performance metric, such as ROUGE Score, to evaluate the effectiveness and efficiency of our proposed model. This chapter also highlights any significant findings or trends observed during the analysis of the results. Through this in-depth exploration and comprehensive discussion, readers gain valuable insights into the performance and capabilities of our proposed model and gain a nuanced understanding of how it compares to existing state-of-the-art approaches. The upcoming chapter concludes the work done in this thesis and provides future directions for readers to expand upon this research.

Chapter 6

Conclusion and Future Work Directions

6.1 Conclusion

The purpose of this thesis was to develop a model capable of identifying the most essential sentences in long documents. This goal has been met to some extent. In this section, we will summarize our findings and, eventually, respond to the primary study question.

1. **How might ArXiv and PubMed abstracts be used to generate labelled data for a supervised model?**
   A Greedy Sentence Selection technique is used to generate labelled data from these summaries. This technique generates labels by maximizing the similarity between the summaries and their source documents using ROUGE score. This algorithm was chosen because of its computational efficiency and shown ability to provide high-quality labels in similar projects.

2. **How should the model's performance be assessed and evaluated?**
   It is critical to evaluate and examine the performance of a summarization model in order to establish its value and dependability. One of the most typical approaches for evaluating the performance of a summarization model is comparison with human-written summaries. By comparing the generated summaries of the model with those created by humans, we can gain insights into how closely the model's output approximates an ideal summary. ROUGE is an automatic met-

ric that can be used for this comparison. ROUGE compares n-grams and word sequences to determine the similarity between produced and human-written summaries. The ROUGE score gives a quantifiable assessment of the model's performance, which can then be compared to the effectiveness of other summarization models.

3. **How may BERT be utilized for extractive text summarization of long documents?**
   BERT, an OpenAI pre-trained transformer model, can be used in a novel method for extracting text summarization of large texts. The method entails breaking the document into smaller sections and then applying BERT to each section individually. This method outperforms truncating the document or utilizing a sliding window approach. The proposed method of combining BERT with an Encoder-Decoder architecture has proven to be a successful model for creating relevant summaries of large documents while taking the global context into account. This demonstrates that BERT has a lot of potential for use in tasks where global context is critical, such text summarization.

4. **What are the shortcomings of proposed approach?**
   The analysis conducted indicates that the proposed model is well-suited for processing long documents. However, for short documents, it is recommended to use BERTSUMEXT, a state-of-the-art summarization model. This is due to the fact that the proposed model is computationally expensive, and BERTSUMEXT outperforms it on small documents in terms of ROUGE-F1 score.

6.2 Future Work

The approach proposed for leveraging BERT in extractive text summary of large documents has various benefits, including:

1. The method provides a general way processing long documents using BERT so it can be used for a variety of NLP tasks such as document-level machine translation and question answering, making it a useful tool for a wide range of NLP applications other than text summarization.

2. Because the approach is easily expanded to other NLP task so one can use it to generate abstractive text summary as well, it becomes a versatile tool for summarizing text in a variety of ways.

Bibliography

[1] Chin-Yew Lin. “Rouge: A package for automatic evaluation of summaries”. In: *Text summarization branches out*. 2004, pp. 74–81.

[2] Minakshi Tomer and Manoj Kumar. “Multi-document extractive text summarization based on firefly algorithm”. In: *Journal of King Saud University - Computer and Information Sciences* 34.8, Part B (2022), pp. 6057–6065. ISSN: 1319-1578. DOI: https://doi.org/10.1016/j.jksuci.2021.04.004.

[3] Adhika Pramita Widyassari et al. “Review of automatic text summarization techniques & methods”. In: *Journal of King Saud University - Computer and Information Sciences* 34.4 (2022), pp. 1029–1046. ISSN: 1319-1578. DOI: https://doi.org/10.1016/j.jksuci.2020.05.006.

[4] Rada Mihalcea and Paul Tarau. “Textrank: Bringing order into text”. In: *Proceedings of the 2004 conference on empirical methods in natural language processing*. 2004, pp. 404–411.

[5] Hans Peter Luhn. “A statistical approach to mechanized encoding and searching of literary information”. In: *IBM Journal of research and development* 1.4 (1957), pp. 309–317. DOI: 10.1147/rd.14.0309.

[6] Karen Sparck Jones. “A statistical interpretation of term specificity and its application in retrieval”. In: *Journal of documentation* 28.1 (1972), pp. 11–21. DOI: https://doi.org/10.1108/eb026526.

[7] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018). DOI: https://doi.org/10.48550/arXiv.1810.04805.

[8] Ming Ding et al. “Cogltx: Applying bert to long texts”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 12792–12804.

[9] Arash Afkanpour et al. "BERT for Long Documents: A Case Study of Automated ICD Coding". In: *arXiv preprint arXiv:2211.02519* (2022). DOI: https://doi.org/10.48550/arXiv.2211.02519.

[10] Zhiguo Wang et al. "Multi-passage bert: A globally normalized bert model for open-domain question answering". In: *arXiv preprint arXiv:1908.08167* (2019). DOI: https://doi.org/10.48550/arXiv.1908.08167.

[11] Thomas Wolf et al. "Huggingface's transformers: State-of-the-art natural language processing". In: *arXiv preprint arXiv:1910.03771* (2019). DOI: https://doi.org/10.48550/arXiv.1910.03771.

[12] Mandar Joshi et al. "BERT for coreference resolution: Baselines and analysis". In: *arXiv preprint arXiv:1908.09091* (2019). DOI: https://doi.org/10.48550/arXiv.1908.09091.

[13] Jeffrey L Elman. "Finding structure in time". In: *Cognitive science* 14.2 (1990), pp. 179–211.

[14] Tomas Mikolov et al. "Recurrent neural network based language model." In: *Interspeech*. Vol. 2. 3. Makuhari. 2010, pp. 1045–1048.

[15] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[16] Christopher Olah. "Understanding lstm networks". In: (2015).

[17] Infolks Group. *Recurrent neural network and long term dependencies*. Mar. 2020. URL: https://medium.com/tech-break/recurrent-neural-network-and-long-term-dependencies-e21773defd92.

[18] Kyunghyun Cho et al. "On the properties of neural machine translation: Encoder-decoder approaches". In: *arXiv preprint arXiv:1409.1259* (2014).

[19] Junyoung Chung et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555* (2014).

[20] Ye Zhang and Byron Wallace. "A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification". In: *arXiv preprint arXiv:1510.03820* (2015).

[21] Kyunghyun Cho et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *arXiv preprint arXiv:1406.1078* (2014).

[22] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[23] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[24] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[25] Yonghui Wu et al. "Google's neural machine translation system: Bridging the gap between human and machine translation". In: *arXiv preprint arXiv:1609.08144* (2016).

[26] Rani Horev. *Bert explained: State of the art language model for NLP*. Nov. 2018. URL: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270.

[27] M Esther Hannah, Saswati Mukherjee, and K Ganesh Kumar. "An extractive text summarization based on multivariate approach". In: *2010 3rd International conference on advanced computer theory and engineering (ICACTE)*. Vol. 3. IEEE. 2010, pp. V3–157.

[28] Sai Teja Polisetty* et al. "Extractive text summarization for sports articles using statistical method". In: *International Journal of Recent Technology and Engineering (IJRTE)* 8.6 (2020), pp. 5622–5627. DOI: 10.35940/ijrte.f9965.038620.

[29] Taner Uçkan and Ali Karcı. "Extractive multi-document text summarization based on graph independent sets". In: *Egyptian Informatics Journal* 21.3 (2020), pp. 145–157.

[30] Ramesh Chandra Belwal, Sawan Rai, and Atul Gupta. "A new graph-based extractive text summarization using keywords or topic modeling". In: *Journal of Ambient Intelligence and Humanized Computing* 12.10 (2021), pp. 8975–8990.

[31] Danqing Wang et al. "Heterogeneous graph neural networks for extractive document summarization". In: *arXiv preprint arXiv:2004.12393* (2020).

[32] Rada Mihalcea. "Graph-based ranking algorithms for sentence extraction, applied to text summarization". In: *Proceedings of the ACL interactive poster and demonstration sessions*. 2004, pp. 170–173.

[33] Yazan Alaya AL-Khassawneh, Naomie Salim, and Obasa Adekunle Isiaka. "Extractive text summarisation using graph triangle counting approach: Proposed method". In: *1 st International Conference of Recent Trends in Information and Communication Technologies in Universiti Teknologi Malaysia, Johor, Malaysia*. 2014, pp. 300–311.

[34] Nikolaos Gialitsis, Nikiforos Pittaras, and Panagiotis Stamatopoulos. "A topic-based sentence representation for extractive text summarization". In: *Proceedings of the Workshop MultiLing 2019: Summarization Across Languages, Genres and Sources*. 2019, pp. 26–34.

[35] Ruby Rani and DK Lobiyal. "An extractive text summarization approach using tagged-LDA based topic modeling". In: *Multimedia tools and applications* 80 (2021), pp. 3275–3305.

[36] Kalliath Abdul Rasheed Issam, Shivam Patel, et al. "Topic modeling based extractive text summarization". In: *arXiv preprint arXiv:2106.15313* (2021).

[37] Prannoy Subba, Susmita Ghosh, and Rahul Roy. "Partitioned-based clustering approaches for single document extractive text summarization". In: *Mining Intelligence and Knowledge Exploration: 5th International Conference, MIKE 2017, Hyderabad, India, December 13–15, 2017, Proceedings 5*. Springer. 2017, pp. 297–307.

[38] Sebastian Suarez Benjumea and Elizabeth León. "Genetic clustering algorithm for extractive text summarization". In: *2015 IEEE Symposium Series on Computational Intelligence*. IEEE. 2015, pp. 949–956.

[39] Pankaj Bhole and AJ Agrawal. "Extractive based single document text summarization using clustering approach". In: *IAES International Journal of Artificial Intelligence (IJ-AI)* 3.2 (2014), pp. 73–78.

[40] M Esther Hannah, TV Geetha, and Saswati Mukherjee. "Automatic extractive text summarization based on fuzzy logic: a sentence oriented approach". In: *Swarm, Evolutionary, and Memetic Computing: Second International Conference, SEMCCO 2011, Visakhapatnam, Andhra Pradesh, India, December 19-21, 2011, Proceedings, Part I 2*. Springer. 2011, pp. 530–538.

[41] Ladda Suanmali, Naomie Salim, and Mohammed Salem Binwahlan. “Fuzzy logic based method for improving text summarization”. In: *arXiv preprint arXiv:0906.4690* (2009).

[42] Mahmoud R Alfarra, Abdalfattah M Alfarra, and Jamal M Alattar. “Graph-based fuzzy logic for extractive text summarization (GFLES)”. In: *2019 International Conference on Promising Electronic Technologies (ICPET)*. IEEE. 2019, pp. 96–101.

[43] Jyoti Yadav and Yogesh Kumar Meena. “Use of fuzzy logic and wordnet for improving performance of extractive automatic text summarization”. In: *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE. 2016, pp. 2071–2077.

[44] Rucha S Dixit and SS Apte. “Improvement of text summarization using fuzzy logic based method”. In: *IOSR Journal of Computer Engineering (IOSRJCE)* 5.6 (2012), pp. 5–10.

[45] Niladri Chatterjee, Gautam Jain, and Gurkirat Singh Bajwa. “Single document extractive text summarization using neural networks and genetic algorithm”. In: *Intelligent Computing: Proceedings of the 2018 Computing Conference, Volume 1*. Springer. 2019, pp. 338–358.

[46] Peng Cui, Le Hu, and Yuanchao Liu. “Enhancing extractive text summarization with topic-aware graph neural networks”. In: *arXiv preprint arXiv:2010.06253* (2020).

[47] Balaanand Muthu et al. “A framework for extractive text summarization based on deep learning modified neural network classifier”. In: *Transactions on Asian and Low-Resource Language Information Processing* 20.3 (2021), pp. 1–20.

[48] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. “Summarunner: A recurrent neural network based sequence model for extractive summarization of documents”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 31. 1. 2017.

[49] Yang Liu. “Fine-tune BERT for extractive summarization”. In: *arXiv preprint arXiv:1903.10318* (2019).

[50] S Sudha Lakshmi, Tirupati SPMVV, and M Usha Rani. “Hybrid Approach for Multi-Document Text Summarization by N-gram and Deep Learning Models”. In: ().

[51] Paulus Setiawan Suryadjaja and Rila Mandala. “Improving the Performance of the Extractive Text Summarization by a Novel Topic Modeling and Sentence Embedding Technique using SBERT”. In: *2021 8th International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA)*. IEEE. 2021, pp. 1–6.

[52] Derek Miller. “Leveraging BERT for extractive text summarization on lectures”. In: *arXiv preprint arXiv:1906.04165* (2019).

[53] Salima Lamsiyah et al. “Unsupervised extractive multi-document summarization method based on transfer learning from BERT multi-task fine-tuning”. In: *Journal of Information Science* (2021), p. 0165551521990616.

[54] Vishal Gupta and Gurpreet Singh Lehal. “A survey of text summarization extractive techniques”. In: *Journal of emerging technologies in web intelligence* 2.3 (2010), pp. 258–268.

[55] Yihong Gong and Xin Liu. “Generic text summarization using relevance measure and latent semantic analysis”. In: *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval.* 2001, pp. 19–25.

[56] Dragomir R Radev et al. “Centroid-based summarization of multiple documents”. In: *Information Processing & Management* 40.6 (2004), pp. 919–938.

[57] Ayush Agrawal and Utsav Gupta. “Extraction based approach for text summarization using k-means clustering”. In: *International Journal of Scientific and Research Publications* 4.11 (2014), pp. 1–4.

[58] AKSHI Kumar and ADITI Sharma. “Systematic literature review of fuzzy logic based text summarization”. In: *Iranian journal of fuzzy systems* 16.5 (2019), pp. 45–59.

[59] AKSHI Kumar and ADITI Sharma. “Systematic literature review of fuzzy logic based text summarization”. In: *Iranian journal of fuzzy systems* 16.5 (2019), pp. 45–59.

[60] Arman Cohan et al. “A discourse-aware attention model for abstractive summarization of long documents”. In: *arXiv preprint arXiv:1804.05685* (2018).

[61] Qiuli Qin, Shuang Zhao, and Chunmei Liu. “A BERT-BiGRU-CRF model for entity recognition of Chinese electronic medical records”. In: *Complexity* 2021 (2021), pp. 1–11.

[62] Jin Dai and Chong Chen. “Text classification system of academic papers based on hybrid Bert-BiGRU model”. In: *2020 12th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*. Vol. 2. IEEE. 2020, pp. 40–44.

[63] Qing Yu, Ziyin Wang, and Kaiwen Jiang. “Research on text classification based on bert-bigru model”. In: *Journal of Physics: Conference Series*. Vol. 1746. 1. IOP Publishing. 2021, p. 012019.

[64] Na-Na Zhang and Yinan Xing. “Questions and answers on legal texts based on BERT-BiGRU”. In: *Journal of Physics: Conference Series*. Vol. 1828. 1. IOP Publishing. 2021, p. 012035.

[65] Kyunghyun Cho et al. “On the properties of neural machine translation: Encoder-decoder approaches”. In: *arXiv preprint arXiv:1409.1259* (2014).

[66] Chris Kedzie, Kathleen McKeown, and Hal Daume III. “Content selection in deep learning models of summarization”. In: *arXiv preprint arXiv:1810.12343* (2018).

[67] Yang Liu and Mirella Lapata. “Text summarization with pretrained encoders”. In: *arXiv preprint arXiv:1908.08345* (2019).

[68] Thomas Wolf et al. “Huggingface’s transformers: State-of-the-art natural language processing”. In: *arXiv preprint arXiv:1910.03771* (2019).

[69] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

Publications

1. S. Bano and S. Khalid, "BERT-based Extractive Text Summarization of Scholarly Articles: A Novel Architecture," 2022 International Conference on Artificial Intelligence of Things (ICAIoT), Istanbul, Turkey, 2022, pp. 1-5, doi: 10.1109/ICAIoT57170.2022.10121826. **{Published as First Author}.**

2. "Summarization of Scholarly Articles Using BERT and BiGRU: Deep Learning-Based Extractive Approach," Journal of King Saud University Computer and Information Sciences. **{Under Review as First Author}.**

Appendix A: Formulas used, and Evaluation Metrics.

This appendix provides a comprehensive overview of the mathematical formulations utilized in the research study. Specifically, we delve into the equations pertaining to BiGRU hidden states and the attention mechanism. Furthermore, we elaborate on the evaluation metric adopted, namely the ROUGE score. We elucidate the formulas encompassing ROUGE recall, precision, and F-measure, thereby offering a detailed understanding of the evaluation process.

1. BiGRU Hidden States

The BiGRU (Bidirectional Gated Recurrent Unit) model is a type of recurrent neural network architecture that incorporates both forward and backward information flow. The hidden states in BiGRU are computed using the following formulas:

$$\begin{aligned}
\text{Forward GRU:} \quad & \overrightarrow{h}_t = EGRU(\overrightarrow{c}_t, \overrightarrow{h}_{t-1}) \\
\text{Backward GRU:} \quad & \overleftarrow{h}_t = EGRU(\overleftarrow{c}_t, \overleftarrow{h}_{t-1}) \\
\text{BiGRU Hidden State:} \quad & z = [\overrightarrow{h}_T, \overleftarrow{h}_T]
\end{aligned}$$

2. Attention Mechanism

The attention mechanism allows the model to focus on specific parts of the input sequence when generating predictions. The attention vector is computed using the following formulas:

$$\begin{aligned}
\text{Energy:} \quad & E_t = tanh(attn(d_{t-1}, H)) \\
\text{Attention Vector:} \quad & \hat{a}_t = uE_t
\end{aligned}$$

3. ROUGE Score Formulas

The ROUGE score is a commonly used evaluation metric for text summarization tasks. It assesses the overlap between the generated summary and the reference summary. The key formulas for ROUGE recall (R), precision (P), and F-measure (F) are as follows:

$$\begin{aligned}
\text{ROUGE Recall } (R)\text{:} \quad & R = \frac{\text{Number of overlapping n-grams in generated summary}}{\text{Total number of n-grams in reference summary}} \\
\text{ROUGE Precision } (P)\text{:} \quad & P = \frac{\text{Number of overlapping n-grams in generated summary}}{\text{Total number of n-grams in generated summary}} \\
\text{ROUGE F-measure } (F)\text{:} \quad & F = \frac{2 \times P \times R}{P + R}
\end{aligned}$$

These formulas provide a quantitative measure of how well the generated summary aligns with the reference summary, enabling a comprehensive evaluation of the summarization model's performance.