People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

جامعة الشهيد حمّه لخضر - الوادي

UNIVERSITY OF ELOUED

ECHAHID HAMMA LAKHDAR UNIVERSITY EL-OUED

Faculty of Exact Sciences - Computer Science department

Option Distributed Systems and Artificial Intelligence

MASTER'S THESIS

to obtain the diploma of Master degree in Computer Science

AI Approach For Automatic Text Summarization

Realized by:

Sekhri Thamer

Bedida Mohammed

Under supervision of:

Dr.Belila Khaoula

Jury Members

Dr.Ben Ali Abdelkamel

Dr.Guia Sana Sahar

June 2024

Acknowledgments

I would like to express my deepest gratitude to everyone who has supported and guided me throughout the journey of completing this master's thesis.

First and foremost, I would like to thank my supervisor **Dr.Belila Khaoula**, for their invaluable guidance, insightful feedback, and constant encouragement. Their expertise and dedication have been instrumental in shaping this research.

I am profoundly grateful to **Josh Starmer**, whose educational content on his YouTube channel, "StatQuest with Josh Starmer," provided clear and comprehensive explanations of complex statistical concepts, significantly enhancing my understanding and application of these techniques in my research.

Finally, I would like to acknowledge the Ministry of Higher Education and Scientific Research for providing the resources and environment conducive to conducting this research.

Thank you all for your contributions and support.

Dedication

A special feeling of love to my siblings: my sisters and brothers, whose presence is a constant source of inspiration and joy in my life.

I dedicate this dissertation to my amazing friends, whom I consider the best thing that happened to me throughout university. Your friendship, humor, and support have been invaluable."

Mohammed *"I would like to dedicate this report first and foremost to the sake of Allah, our creator, my source of wisdom, knowledge, and understanding. To my beloved parents,* ***Bedida Laid*** *and* ***Chaba Soumia****, for all their unconditional love, support, and continuous encouragement throughout this academic journey.*

To my brother ***Ahmed****, my sisters* ***Malak*** *and* ***Mabrouka****, my aunt* ***Asia****, and my teacher* ***Ms. Chourouk Guettas****, your constant encouragement and unwavering belief in me have been invaluable. Your companionship and support have greatly influenced my journey, always pushing me to strive for excellence. Additionally, I extend my heartfelt gratitude to my friend* ***Anis Houri*** *for his steadfast support and*

friendship.

A special feeling of love to my siblings: my sisters and brothers, whose presence is a constant source of inspiration and joy in my life.

I dedicate this dissertation to my amazing friends, whom I consider the best thing that happened to me throughout university. Your friendship, humor, and support have been invaluable."

Mohammed

"This thesis is dedicated to my beloved parents, ***Fouad*** *and* ***Naima Nagoudi****, whose unwavering support and love have been the cornerstone of my life. You have been my guides, my inspiration, and my strength through every step of this journey. Your sacrifices and encouragement have shaped the person I am today, and I am eternally grateful for everything you have done for me...*

To my brothers, ***Aymen*** *and* ***Bidjed****, and my sisters,* ***Hazar*** *and* ***Mayssoun****, thank you for your constant encouragement and belief in me. Your companionship and support have been invaluable, and your faith in my abilities has always pushed me to strive for excellence...*

I also extend my heartfelt gratitude to my friends, ***Islam****,* ***Alla Eddine****, and* ***Moujib Errahmane****. Your friendship has been a source of great comfort and motivation throughout this journey. Thank you for always being there, for your unwavering support, and for making this journey a memorable one.*

This thesis is a testament to the love, support, and encouragement I have received from all of you. Thank you for believing in me and for being an integral part of this accomplishment."

Thamer

ملخص

تتناول هذه الأطروحة مجال تلخيص النصوص الآلي، وهو حل حاسم لتحدي إدارة الكمية الهائلة من البيانات النصية التي يتم إنشاؤها يوميًا. نستكشف تقنيات التلخيص المختلفة، بما في ذلك الأساليب الاستخراجية، التجريدية، والهجينة. تشمل دراستنا مراجعة شاملة للنماذج الحالية وتقييمها . نقترح إطارًا محسّنًا يجمع بين نقاط القوة في هذه التقنيات لتحسين جودة التلخيص. تُظهر التجارب التي أجريت على مجموعة بيانات فعالية نهجنا في توليد ملخصات موجزة ومتماسكة، مما يدفع بمستوى الفن في تلخيص النصوص.

الكلمات المفتاحية: تلخيص النصوص، الأساليب الاستخراجية، الأساليب التجريدية، الأساليب الهجينة، التعلم العميق، معالجة اللغة الطبيعية

Abstract

This thesis investigates the field of automatic text summarization, a crucial solution to the challenge of managing the vast amount of textual data generated daily. We explore various summarization techniques, including extractive, abstractive, and hybrid approaches. Our study includes a thorough review of existing models and their evaluation. We propose an enhanced framework that combines the strengths of these techniques to improve summarization quality. Experiments conducted on the CNN/DailyMail dataset demonstrate the effectiveness of our approach in generating concise and coherent summaries, advancing the state-of-the-art in text summarization.

Keywords: text summarization, extractive methods, abstractive methods, hybrid approaches, deep learning, natural language processing

Résumé

Cette thèse explore le domaine de la synthèse automatique de texte, une solution essentielle au défi de gérer la grande quantité de données textuelles générées quotidiennement. Nous examinons diverses techniques de résumés, y compris les approches extractives, abstraites et hybrides. Notre étude comprend une revue approfondie des modèles existants et leur évaluation. Nous proposons un cadre amélioré qui combine les forces de ces techniques pour améliorer la qualité des résumés. Les expériences menées sur le jeu de données CNN/DailyMail démontrent l'efficacité de notre approche pour générer des résumés concis et cohérents, faisant progresser l'état de l'art en matière de résumés de textes.

Mots clés : synthèse de texte, méthodes extractives, méthodes abstraites, approches hybrides, apprentissage profond, traitement du langage naturel

Table of Contents

List of Figures

List of Tables

General Introduction

Project Background

In today's world, there's an overwhelming amount of information out there, thanks to the internet, social media, and digital documents, and that led to an unprecedented volume of textual data Manually processing such a volume is not only challenging but also time-consuming.

Text summarization(TS) emerges as a crucial solution to this problem, Providing the ability to generate short and concise summaries for a single document or a set of documents.

Summarizing text comes in two flavors, depending on what you're working with. Single-document summarization is like distilling the essence of one article or report. On the other hand, if you've got a stack of related documents, multi-document summarization helps you pull out the most important themes that run across them, Early text summarization systems mostly focused on single-document summarization.[5]

There are also two primary methods used in text summarization: extractive and abstractive methods. In **extractive summarization**, the goal is to pick out sentences or words from the text that hold the key information. These selected parts are then used as the summary without alteration. **Abstractive summarization** methods rephrase the text to create summaries that are more adaptable and coherent by

changing the words while focusing on the form to achieve that.

Motivation

The whole point of automatic text summarization is to make dealing with large amounts of information less of a headache. The idea goes back to Hans Peter Luhn's research in 1958 [6], where he pioneered the idea of teaching computers to find the most important parts for summaries. The field has been focused on refining this process ever since, giving people a shortcut to understanding what a document is really about. The reason for this is simple: there's just way too much to read! Between long articles, research papers, and trying to stay on top of current events, it can feel impossible. Automatic text summarization is here to help, giving you the essentials in a quick, digestible format.

Objective

This thesis will delve into existing text summarization techniques, comparing those that extract key parts versus those that rewrite the text. We'll thoroughly review the latest methods, run experiments using standard measures (like ROUGE and BLEU), and look closely at what makes a summary successful. Our goal is to understand what makes one technique better than another in different situations, ultimately offering guidelines for choosing the right tool for the job.
Furthermore, based on our findings, WE will propose a new framework that combines the best aspects of these approaches to make summarization systems even more effective and efficient. This thesis aims to push the field of text summarization forward by offering both practical insights and a fresh way to tackle the problem.

Dataset used

The **CNN / DailyMail** Dataset is an English-language dataset containing just over 300k unique news articles as written by journalists at CNN and the Daily

Mail.it consists of news articles paired with bullet-point summaries, providing a diverse range of topics and writing styles. This dataset offers a valuable resource for training and evaluating summarization models due to its large size and high-quality annotations.

Document Plan

Chapter 1 “Types of summarization”: This chapter examines different approaches to condensing information, including methods for selecting key content, techniques for rephrasing and condensing text, and strategies for handling real-time updates and multilingual contexts.

Chapter 2 “Fundamentals & Related works”: This chapter provides an overview of existing research and methodologies in the field of text summarization, including a review of both extractive and abstractive techniques.

Chapter 3 "Proposition": In this chapter, a new approach is introduced that combines the benefits of both extractive and abstractive summarization methods to improve the overall effectiveness and efficiency of summarization. It suggests practical ways to overcome current challenges and enhance the performance of text summarization systems.

Chapter 4 "Implemntation and results": This chapter details the practical implementation of the proposed model and evaluates its performance, comparing results from each development stage to the final model.

Chapter 5 "Conclusion": This part sums up what we found in the thesis, focusing on the comparison of text summarization methods. It stresses the importance of our proposed framework for improving text summarization and suggests ideas for future research.

Chapter 1

Types of Text Summarization

Introduction

This chapter discusses the various ways and methodologies used to condense enormous quantities of text into short summaries. In the current landscape filled with abundant information, the need for effective summarization techniques has become increasingly paramount. This chapter offers a detailed overview of the different approaches used in the field of text summarization Summaries are categorized according to their number and genre of documents, type of summary & types of summarizer, the context, and their function.[7]

We start with extractive summarization, Where we use strategies for selecting important sentences or phrases from the original text. Then, We move to abstractive summarization approaches, which involve rewriting and rephrasing long text to create more flexible and coherent summaries. We also talk about hybrid approaches that combine extractive and abstractive techniques to utilize the best of both.

In this chapter, our goal is to help readers comprehend the different types of summarization techniques, their principles & how they can be applied. understanding the advantages and drawbacks of each technique will help readers select the most suitable summarization method for their needs.

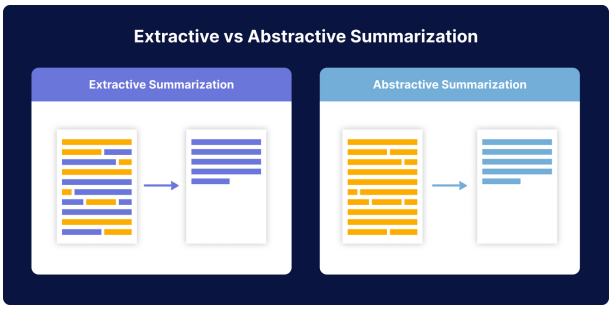

Figure 1.1: Extractive vs Abstractive summarization source [1]

1.1 Extractive Summarization

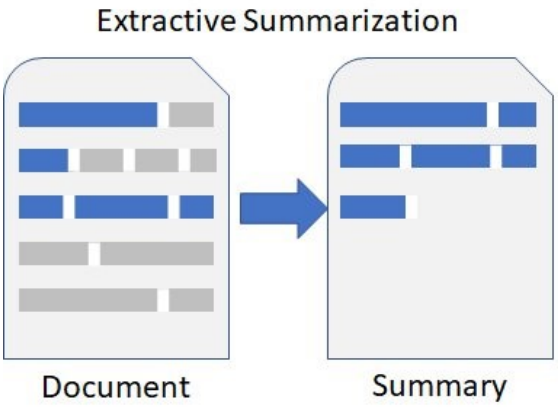

Figure 1.2: Extractive summarization source [2]

Extractive summarization is the most common approach to TS problem, and it's a crucial aspect of distilling text content. It involves directly selecting important sentences or phrases from the original text to create a summary. Various methodologies and algorithms are used in extractive summarization, each with its strengths and factors to consider. Figure 1.2 illustrates the concept of abstractive summarization.

Extractive summarization is all about finding the most important sentences or phrases in a piece of writing. Each text unit is given a score by the system, Afterward, it picks out the most informative ones to create a summary. The goal is to create a short summary that still has all the key information. There are lots of techniques that often involve the use of algorithms that assess the importance of sentences such as **sentence scoring** which is based on many factors like word fre-

quency & sentence length. In addition to **Graph-based algorithms** like TextRank & LexRank which uses network of interconnected sentences and ranks them. Also **Machine learning methods** including supervised and unsupervised approaches. Despite the effectiveness of Extractive summarization, it counters various challenges. First, The redundancy, where extracted sentences may redundantly convey similar information, making the summary boring. Second, it can be hard to make selected sentences fit together smoothly, especially when they are from different parts of the text. Lastly, extractive methods may struggle to encompass all important information from the original text, potentially resulting in information loss. Addressing theses challenges is important to enhance the quality of the summarization.

1.2 Abstractive Summarization

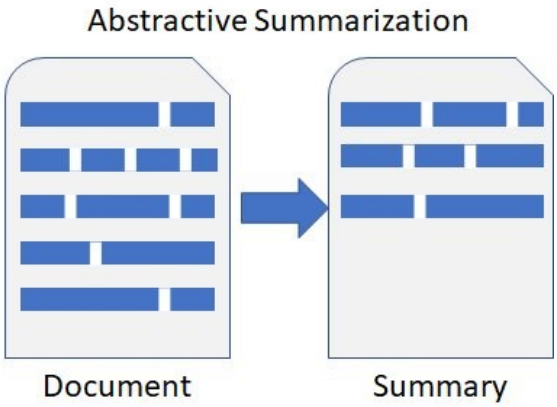

Figure 1.3: Abstractive summarization source [2]

Abstractive summarization aims to generate concise summaries that go beyond extraction sentences from the source text. It try to understand the text(Just like human-being) by identifying the major concepts and generate a summary using different words and sentence structures, Abstractive techniques can change words and put together information to make a better, shorter summary that makes sense.

Figure **??** illustrates the concept of abstractive summarization.

Abstractive summarization techniques use advanced NLP and machine learning to make summaries that seem like they were written by people. These methods include: **Sequence-to-Sequence Models** which is built primarily on the attention mechanism, uses neural networks (NN) like encoder-decoder models to learn how to change the original text into the summary by using Long Short-Term Memory (LSTM) and recurrent neural network (RNN). [4] In addition, **Attention Mechanisms**: helps the model identify the key parts of the original text when making summaries, which makes the summaries better. Also, **Reinforcement Learning** trains the models to make summaries by giving them rewards when they do well.
Abstractive summarization encounters various problems that need creative solutions. First, There's fluency & coherence, which means making sure they summary sounds smooth and not like it's generated by a model, so people can easily understand it. there's the challenge of content preservation, where it's crucial to keep the main ideas from the input text and don't effect their meaning. Another aspect to consider is knowledge incorporation, where we include additional information from outside sources to improve the accuracy and usefulness of the summary.

1.3 Hybrid Approaches

Hybrid approaches aim to combine accuracy of extractive summarization, which captures important sentences from the original text,with the adaptability of abstractive techniques that creates brief & clear summaries. These hybrid models seek to achieve a balance between the original text and the ability to provide human-written text.
Hybrid summarization techniques offer a variety of approaches, such as **Extractive-then-Abstractive methods**, where we chose key sentences using extractive methods, and then rewrote them using abstractive methods to improve coherence. Alternatively, there is **Abstractive-then-Extractive methods** where we generate a summary with abstractive methods and then add extracted information from the original text. Also, **Fusion Models** combine both extractive and abstractive meth-

ods into a single framework to enable smooth interaction between them.
Although hybrid summarization approaches hold potentials, they face various obstacles like the complexity of models, Creating such models, and training them while ensuring they remain efficient and scalable can be complex. Also, finding suitable evaluation criteria to evaluate both extractive and abstractive features is difficult.

1.4 Evaluation Metrics

Evaluating the quality of automatically produced summaries involves subjectivity, as there is no single "perfect" summary, and remains difficult.but there are various measures and standards used to assess the effectiveness of text summarization. These metrics is crucial in determining the quality and performance of such algorithms. and thats by measuring precision,recall and F1 measure. The most famous evaluation methods nowadays are: **Rouge** (Recall-Oriented Understudy for Gisting Evaluation), **BLEU**(Bilingual Evaluation Understudy) and **METEOR** (Metric for Evaluation of Translation with Explicit Ordering).

1.4.1 ROUGE

ROUGE [8] is the most used evaluation metric in automatic text summarization research, it's based on the BLEU metrics. The idea is to compare an automatically produced summary with a human-produced reference summary based on various criteria such as n-gram overlap, recall, and precision.
ROUGE-N: The ROUGE-n score evaluates the similarity between n-grams (gram length) present in both the reference and the generated text. It measures n consecutive words that appear both in the evaluated text and the reference text. ROUGE-1 measures the overlap of individual words between the generated summary and the reference summaries, assessing word usage and vocabulary similarity. ROUGE-2 compares the overlap of consecutive word pairs (bigrams) between the generated summary and the reference summaries, evaluating coherence and fluency. To cal-

culate ROUGE scores, we need to understand Recall and Precision in text. Figure 1.4 illustrates the Recall and Precision equations.

$$RECALL = \frac{Overlapping\ number\ of\ n-grams}{Number\ of\ n-grams\ in\ the\ reference}$$

$$PRECISION = \frac{Overlapping\ number\ of\ n-grams}{Number\ of\ n-grams\ in\ the\ candidate}$$

Figure 1.4: Recall and Precision equations

After understanding Recall and Precision, we can calculate the F1 scores as shown in Figure 1.5.

$$F1 = \frac{2 \times Precision \times Recall}{(Precision + Recall)}$$

Figure 1.5: F1 equation

ROUGE-L: focuses on the match of n-consecutive words between the reference and the automatically generated text, whereas ROUGE-L examines the longest common subsequence of words (LCS) in both texts. [9]
ROUGE-S: is a skip-gram concurrence metric that evaluates n-grams present in the reference text while allowing for variations in word order in the model output, as long as the words still appear in sequence.

1.4.2 BLEU

The BLEU (Bilingual Evaluation Understudy) metric is a widely used measure for assessing the quality of automatically generated summaries or translations. It compares the generated summary to one or more reference summaries based on the precision of n-grams in the generated summary. BLEU typically calculates precision scores for unigrams, bigrams, trigrams, and sometimes higher-order n-grams. These scores are then combined using a geometric mean, with a brevity penalty to account for summaries that are shorter than the reference summaries. BLEU scores range from 0 to 1, with higher scores indicating better quality summaries. [10]

1.5 Applications of Summarization

various real-world scenarios where summarization methods are explored. These applications span across diverse domains such as news summarization where Suitable headlines can be automatically produced using a test summarizer for each news to reduce time and effort [11], social media summarization for capturing key messages in online discussions, also, Abstract for papers of researchers where we can generate succinct summaries of lengthy documents, such as research papers or even legal documents, aids in information retrieval and facilitates quicker understanding of complex materials. in addition, other applications include email summarization for managing inbox overload or meeting summarization for recalling key decisions and discussions,legal document summarization for extracting essential legal information, educational summarization for aiding learning and revision, healthcare summarization for extracting relevant medical information, and audio/video summarization for quickly grasping main points without consuming the entire content. [12]
These examples highlight how text summarization methods enhance effectiveness, productivity, and decision-making across different sectors and fields.

1.5.0.1 News Summarization

News summary is the process of compressing news stories into brief summaries while preserving important details.Techniques include statistical methods that analyze term frequency, graph-based methods like TextRank to identify key sentences, and machine learning classifiers such as SVM to determine sentence importance. The objective of these strategies is to simplify all of the details found in news items into short and informative summaries, making them more easily understandable.

1.5.0.2 Opinion/Sentiment Summarization

Opinion or sentiment summary is the process of summarizing the sentiments or thoughts represented in texts, such as reviews or social media posts. This process

typically involves identifying and extracting key phrases or sentences that reflect the overall sentiment, whether positive, negative, or neutral. echniques often include sentiment analysis algorithms, which classify the sentiment of each sentence (i.e. classify as "Strong Positive", "Positive", "Negative", and "Strong Negative".), Afterwards, the process involves condensing the content by highlighting the most indicative sentences that convey sentiment or combining sentiment ratings to offer a comprehensive summary of the offered viewpoints. This approach enables the understanding of overall sentiment trends as well as important viewpoints within a wide range of texts.

1.5.0.3 Microblog/Tweet Summarization

Social networking platforms such as Facebook, Twitter, and others include a huge amount of communication, Microblog or tweet summarization focuses on condensing short, informal messages from these platforms. During emergency incidents, platforms like "Twitter" serve as crucial sources of real-time information due to the vast number of "tweets" that are posted. As a result, the importance of microblog summarization has increased significantly in recent years.

1.5.0.4 Books Summarization

Book summarization is the process of condensing long, complicated stories or nonfiction into short outlines that focus on the main ideas, plot points, and important details. Natural language processing (NLP) tools may be used in more advanced methods to look at the structure and content of the text and create summaries that capture the main ideas and important details of the book while still giving a full but short picture of the whole thing.

1.5.0.5 Story/Novel Summarization

Story or book summarization is the process of condensing long stories into short ones that capture the main plot, important characters, and important events. This method gives a short but complete summary, which makes complicated events easier to understand.

1.5.0.6 Email Summarization

Email summarization is the process of condensing emails into short summaries that show important information, action items, and key points. Natural language processing tools can look at the structure and content of the emails and write short recaps that get to the heart of the conversation. This makes it easier to quickly find the important information and actions that need to be taken from a large number of emails.

1.5.0.7 Biomedical Documents Summarization

Biomedical document summarization focuses on condensing scientific articles, clinical reports, and medical texts into concise summaries that highlight key findings, methodologies, and conclusions. It aims to streamline complex and detailed biomedical information, making it accessible for quick understanding and decision-making by researchers, clinicians, and healthcare professionals. Summaries typically emphasize significant results, clinical implications, and essential data points, facilitating efficient knowledge extraction from extensive biomedical literature.

1.5.0.8 Legal Documents Summarization

Legal document summarization is the process of condensing long legal texts into short ones that include important decisions, laws, arguments, and points. Its goal is to make complicated legal information easier to find by highlighting important details like case outcomes, legal precedents, and important arguments. This will help legal workers and other interested parties quickly understand and analyze legal documents.

1.5.0.9 Scientific Papers Summarization

Scientific paper summarization is the process of condensing research papers into short outlines that focus on the most important results, methods, and conclusions. It focuses on showing the important parts of the study, like the goals, methods,

outcomes, and implications. This makes complicated scientific information easier for researchers, practitioners, and students to understand. These summaries make it easier to quickly understand and judge scientific advances and contributions.

1.6 Conclusion

In this chapter, we explored various text summarization techniques, including extractive, abstractive, and hybrid methods. Extractive summarization focuses on selecting key sentences, while abstractive summarization rephrases the text to create more coherent summaries. Hybrid approaches combine both techniques to enhance summary quality. Despite their effectiveness, each method faces challenges such as redundancy, coherence, and information preservation. Evaluation metrics like ROUGE, BLEU, and METEOR are essential for assessing performance. Applications of these techniques span across diverse domains, highlighting their significance in managing large volumes of text efficiently.

Chapter 2

Fundamentals

Introduction

In Chapter 3, we delve into the fundamental concepts and related works that form the backbone of text summarization technologies. This chapter is divided into two sections: the first covers the essential principles of natural language processing (NLP), neural networks, recurrent neural networks (RNN), encoder-decoder architectures, attention mechanisms, and transformers. These foundational topics are critical as they underpin modern summarization approaches and have significantly influenced research in this field. The second section examines seminal works and key contributions in text summarization, providing insights into the evolution of summarization techniques, the challenges encountered, and the milestones achieved. By understanding these fundamentals and reviewing past research, we prepare ourselves to explore innovative ideas and methodologies in the subsequent chapters.

2.1 Fundamentals

In the first section of this chapter, we embark on a journey through the foundational principles in the realm of text summarization such as natural language processing (NLP), neural networks(NN), recurrent neural networks (RNN), encoder-decoder architectures, attention mechanisms, and transformers models. Modern summaris-

ing approaches rely on these notions which have influenced study in the subject.

Then, we explore important works and seminal articles in text summarising.An examination of these key contributions provides insights into the growth of summarization approaches, the problems encountered, and the achievements made.
By understanding these fundamentals and past research, we'll be better prepared to explore new ideas and approaches in the coming chapters.

2.1.1 Natural language processing

Natural Language Processing (NLP) is a core field within computational linguistics. It makes it possible for machines to comprehend, interpret, and generate human language. NLP is an essential component of text summarization technologies since it covers a broad range of activities, from simple language understanding to complicated text production.
NLP primarily focuses on creating algorithms and models capable of efficiently handling and examining data written or spoken in human language. These algorithms utilize techniques from machine learning, statistics, and linguistics. The goal is to get machines to not just read a bunch of words, but 'get' what they mean.
A key objective of Natural Language Processing is to facilitate machines in understanding the messy, complex way we communicate. This process involves breaking complex linguistic systems into simpler components, such as individual words, phrases, and sentences, and then analyzing these components to extract meaning and context.
Historically, the NLP pipeline consisted of numerous steps ranging from preprocessing to the final application. But recently the old approach has been significantly replaced by the new deep learning-based approaches which have shown exceptional performance. This strategy involves training end-to-end models without the need for human feature engineering.
Even though NLP has come a long way, it still has problems, like understanding context, dealing with ambiguities in language, and working with languages that

don't have a lot of resources. In the future, NLP study will focus on making models easier to understand, making them more multilingual, and adding more sophisticated contextual understanding.

2.1.2 Neural networks

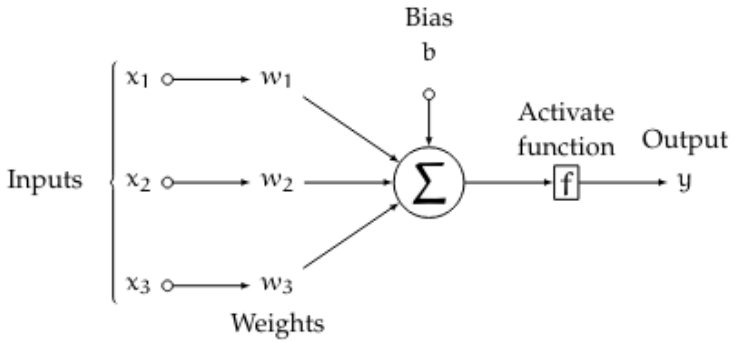

Figure 2.1: The general schema of an artificial neuron

Neural networks are a crucial element of modern artificial intelligence, inspired by the structure and function of the human brain. These networks include of connected layers of nodes, often known as "neurons," that together analyze data and produce outputs. They have outstanding skills in tasks that require pattern identification, data classification, and predictive modeling. Figure 2.1 illustrates the general schema of an artificial neuron.

A neural network is normally composed of three primary types of layers: the input layer, hidden layers, and the output layer. The input layer is the first layer of a neural network that takes raw data, with each neuron representing a specific feature or attribute. Hidden layers, which are intermediary, process the input data using weighted connections. Every individual neuron within a hidden layer undergoes a process of converting the input it receives and subsequently sends the outcome to the next layer. The addition of numerous hidden layers enables the network to collect complex patterns and representations. The output layer, also known as the last layer, generates the output of the network. This output might range from classification outcomes to regression values or other predictions, depending on the specific task at hand.

Back-propagation helps Neural Networks learn optimal weights and biases for connections between neurons. The training algorithm repeatedly modifies weights in order to decrease the gap between the network's predictions and the actual target values. During forward propagation, input data is processed layer by layer, with activations computed using functions like ReLU, sigmoid, or tanh. The output of the network is subsequently compared to the desired target values using a loss function (e.g., mean squared error or cross-entropy loss) in order to measure the amount of prediction error. In the backward pass, the error propagates back through the network, and weights are adjusted using optimization techniques such as gradient descent to reduce future prediction errors.

Various applications utilise different types of neural networks. Feedforward neural networks (FNNs) are a basic type of neural network in which data moves in a single direction from input to output without any loops. Convolutional neural networks (CNNs) are mostly utilised in image processing to identify spatial hierarchies within images through the usage of convolutional layers. Recurrent neural networks (RNNs) are well-suited for handling sequence data as they retain knowledge about earlier inputs by utilising directed cycles. Long short-term memory networks (LSTMs) are a specific sort of recurrent neural networks (RNNs) that are designed to solve the problem of disappearing gradients. This makes them particularly useful for applications such as language modelling. Recently, Transformer networks have become the most advanced models for numerous natural language processing (NLP) applications because of their attention mechanisms, which effectively capture long-range relationships.

Neural networks have revolutionised various domains by providing powerful tools for analysing and understanding complex data. CNNs, or Convolutional Neural Networks, have made significant progress in the field of image recognition, reaching a level of performance that is almost comparable to that of humans. This breakthrough has resulted in notable breakthroughs in areas such as medical imaging and autonomous vehicles. In natural language processing, transformer networks have significantly enhanced tasks like machine translation, sentiment

analysis, and text summarization. Neural networks are widely employed in finance to forecast stock prices, in healthcare to diagnose disorders, and in various other fields that necessitate data-driven decision-making.
Although neural networks have achieved success, they encounter several obstacles such as the required for significant amounts of labelled data for training, demanding computational requirements, and the complexity of understanding their internal mechanisms.

2.1.3 Recurrent neural network

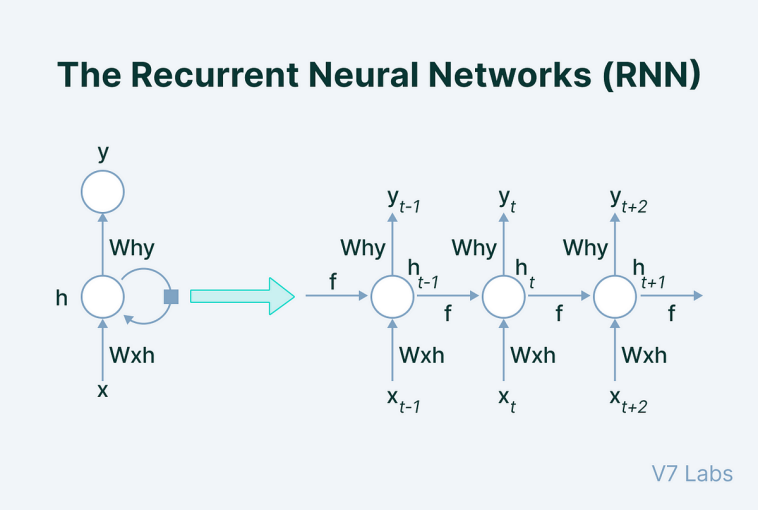

Figure 2.2: The general schema of Recurrent Neural Networks [3]

Recurrent Neural Networks (RNNs) are an interesting form of neural network built to process sequential input. RNNs, Unlike traditional neural networks, which assume that all inputs are independent of each other, RNNs have a built-in memory that takes previous inputs into account. These characteristics make them highly valuable for tasks that need consideration of context and sequence, such as language modelling, time series prediction, and speech recognition. Figure 2.2 illustrates the general schema of Recurrent Neural Networks.

An RNN is based on the idea of loops, which keep information alive in the network. When processing a sequence, an RNN combines input from the current

step with the hidden state from the previous step. This hidden state functions as a memory that captures details about the processed information up to this point. Being able to store and utilise information from previous steps is what differentiates RNNs and makes them highly effective for tasks that involve sequences [3].
Consider the task of language modelling, where the objective is to predict the next word in a sentence. Using the context of previous words, an RNN can enhance its prediction accuracy. the RNN constantly updates its hidden state with information from each new word, allowing it to continually improve its understanding of the sequence. This ability is essential for producing logical text, understanding speech patterns, or even predicting future values in time series data.
Nevertheless, RNNs have a variety of challenges. Dealing with the vanishing gradient problem can be quite challenging as it impacts the network's ability to grasp long-term dependencies. This occurs because the gradients used to update the network's weights can become very small, effectively preventing the network from learning. To address this problem, advanced variants of RNNs, such as Long Short-Term Memory (LSTM) networks have been developed. These architectures contain techniques that effectively capture and maintain long-term dependencies, therefore enhancing the performance of RNNs on tasks that necessitate long-range context.

2.1.4 Encoder-Decoder

The Encoder-Decoder architecture is a popular neural network design that offers outstanding efficiency in managing sequence-to-sequence (Seq2Seq) tasks. These tasks involve transforming one sequence into another, such as translating a sentence from one language to another, summarizing a document, or generating captions for images. The key advantage of this architecture is its capacity to handle input sequences of different lengths and generate output sequences that can also vary in length.
The architecture consists of two primary components: the encoder and the decoder.

The primary function of the encoder is to analyze the input sequence and convert its information into a vector of a predetermined length, often called the context vector or the hidden state. This step of encoding is very important because it changes the variable-length data into a format that the decoder can understand. Usually, layers of recurrent neural networks (RNNs), such as Long Short-Term Memory (LSTM) units, are used to build the encoder. These units are very good at picking up on temporal relationships in the input sequence, which helps the encoder summarize the input well. As the encoder receives each element of the input sequence (such as a word in a phrase), it modifies its hidden state, thus creating a thorough representation of the entire sequence. At the conclusion of the input sequence, the hidden state of the encoder contains a brief summary of all the information obtained from the input. Next, the decoder receives this context vector.

The decoder's role is to utilize the context vector generated by the encoder to produce the output sequence. Like the encoder, the decoder also consists of RNN layers, typically with LSTM. The decoder sequentially builds each element of the output sequence by utilizing the context vector and its own hidden state. At each step, the decoder predicts the next element in the sequence, which can be a word, a token, or another relevant unit, depending on the task. The predicted element is then fed back into the decoder for the next step, allowing the sequence to be generated iteratively [13].

Overall, the Encoder-Decoder architecture is a flexible and efficient framework for sequence-to-sequence applications. It efficiently converts input sequences into a consistent context and converts them into the necessary output sequences, making it capable of handling different sequence lengths and complexities easily. The integration of attention mechanisms further enhances its capability, making it a cornerstone in the field of natural language processing and beyond.

2.1.5 Attention

The attention mechanism is a fundamental breakthrough in neural networks, particularly for sequence-to-sequence (Seq2Seq) models. The proposed approach solves the limitations of the traditional Encoder-Decoder architecture by eliminating dependence on just one context vector of fixed length to encode the complete input sequence. This fixed context vector can struggle with longer sequences, as it might not capture all necessary details. Attention mechanisms allow the model to dynamically focus on various segments of the input sequence while generating each component of the output sequence. This greatly improves the model's effectiveness in tasks like machine translation, text summarization, and picture captioning.

Practically, attention operates by computing a series of attention weights during each stage of the decoding procedure. The weights are used to determine the significance of each segment of the input sequence, taking into account the decoder's current state. The attention weights are computed by comparing the current decoder state with each encoder state using similarity measures. The resulting weights are then used to create a weighted sum of the encoder states, producing a dynamic context vector that emphasizes the most relevant information for generating the current output element. This method enables the decoder to find and utilize specific sections of the input sequence as required, allowing it to properly manage multiple connections and complex patterns [4].

The development of attention mechanisms has had a significant effect, resulting in the development of complex designs such as the Transformer. The Transformer architecture exclusively relies on attention techniques, completely disregarding recurrence and enabling more effective parallelization. This has established new standards in numerous tasks, revealing the extensive influence of attention mechanisms.

2.1.6 Transformers

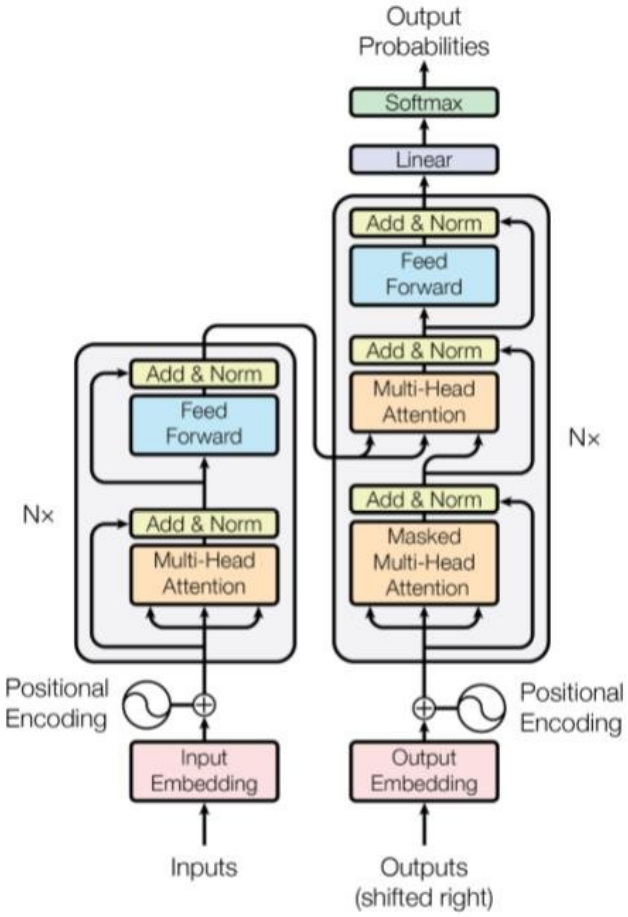

Figure 2.3: The architecture of Transformer [4]

Transformers are a new type of neural network design that was initially introduced by Vaswani et al. in their 2017 paper "Attention is All You Need." [4].Figure 2.3 illustrates the architecture of the Transformer model. Transformers are different from standard RNN-based models since they only use self-attention mechanisms to process and create sequences, eliminating the need for recurrence. With its exceptional performance and efficiency, this architecture has completely changed natural language processing (NLP), setting new standards for tasks like text summarization, language modelling, and machine translation.

The primary breakthrough of the Transformer is its self-attention mechanism, enabling the model to assess the significance of individual words inside a sentence in relation to one another. This method enhances the model's ability to capture

faraway connections and contextual interactions more efficiently compared to Recurrent Neural Networks (RNNs), which process sequences in a sequential manner and may encounter difficulties with long-term dependencies. The Transformer algorithm concurrently processes all words in a sequence, resulting in a high degree of parallelism and a notable improvement in training speed. In addition, the architecture includes multi-head attention, enabling the model to simultaneously focus on various segments of the sequence, hence augmenting its capacity for understanding complex patterns and connections.

The impact of Transformers has been profound, leading to the development of models like BERT (Bidirectional Encoder Representations from Transformers), GPT (Generative Pre-trained Transformer), and T5 (Text-To-Text Transfer Transformer). These models have attained outstanding results in several NLP tasks. Transformers have made a significant impact in fields beyond natural language processing (NLP), such as image processing and speech recognition, showcasing their adaptability and efficacy. Transformers have revolutionised sequence modelling by utilising self-attention and parallel processing, establishing themselves as a fundamental component of current AI research and applications.

Chapter

3

Related Works

Introduction

Figure 3.1 illustrates the various approaches to text summarization, categorizing them into extractive, hybrid, and abstractive methods. This visual representation helps in understanding the different methodologies and their applications in text summarization.

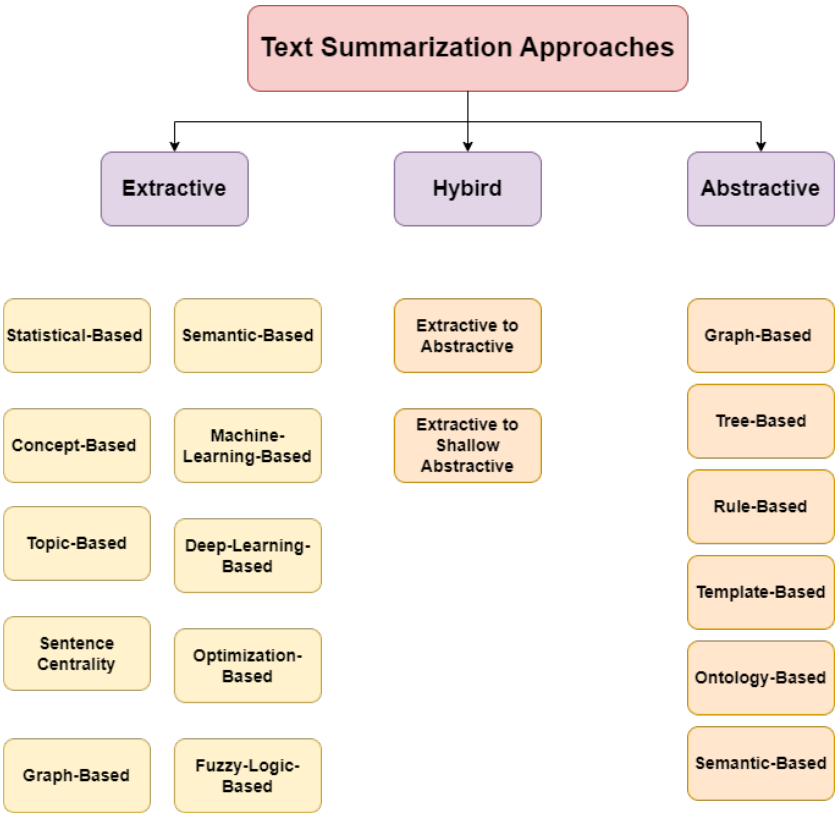

Figure 3.1: methods of text summarization

3.0.1 Extractive Text Summarization

3.0.1.1 Statistical-Based Methods

These methods extract key sentences and words for the original text by utilizing statistical features like word frequency, uppercase words, sentence length, keywords, position in the text, and phrase structure. these methods usually use metrics like term-frequency-inverse document frequency(TF-IDF), which scores words based on their frequency in a particular document compared to their overall rarity across all publications.

Sentences that include words with high TF-IDF scores are defined as "the most frequent" and typically selected for the summary. This technique is straightforward, as it primarily emphasizes word counts and does not require an understanding of the text's deeper meanings. [14]

3.0.1.2 Concept-Based Methods

these methods involve using graph modeling and weighted iterative ranking to identify and extract key concepts from the original input. The importance of sentences is determined by the concepts obtained from the external knowledge base rather than by individual words, The steps involved in scoring sentences for these methods are as follows: firstly, extracting concepts from a text using an external knowledge source, then, building a graph model to visually represent the connections between concept and sentences, finally, use a rating algorithm to assign scores to the phrases. [15]

3.0.1.3 Topic-Based Methods

these methods rely on the identification of the main topics or themes within a text in order to select the most relevant phrases for inclusion in the summary. These methods typically use algorithms like Latent Dirichlet Allocation (LDA) or Non-negative Matrix Factorization (NMF) to model topics as distributions over words. By analyzing the entire document, these methods give probability to each sentence, identifying the ones that most accurately capture the main ideas for the summary [16].

These methods guarantee that the summary covers all the topics discussed in the document, Providing an accurate representation of the content. It efficiently manages documents that contain multiple themes, which makes it useful for summarizing complex & lengthy texts.

several prevalent techniques for representing topics include: Term-frequency, TF-IDF, lexical chains, and topic weights [17]. The steps involved in a topic-based extractive summarizer's processing are firstly converting the input text into an intermediary representation that accurately captures the subjects addressed in the input text, then assigning a score to each sentence in the input based in this representation.

3.0.1.4 Sentence Centrality Methods

These methods focus on identifying the most pivotal or important sentences in a document by considering their connections to other sentences. The sentence centrality is determined by the centrality of its individual words. A conventional approach to quantify word centrality involves examining the centroid of the document cluster within a vector space. We define the centroid of a cluster by taking the sentence that consists of words having TF-IDF scores above a predetermined threshold [18].

The steps involved in scoring in a centroid-based summarizer are by firstly computing the TF-IDF representation for each sentence and building a representative centroid [19], Then The closeness of a sentence to the cluster centroid can be measured by treating it as central if it contains a higher number of terms from the centroid. Sentence importance increases with closeness to the cluster center.

By extracting these central sentences, the summary can capture the essence of the document without redundancy. These methods work especially well at minimizing information overlap in the summary so that every sentence adds something new. In order to select representative sentences for the summary, ATS uses clustering algorithms.

Clustering algorithms are employed for ATS, with the sentence selection process consisting of three steps: 1) applying a clustering algorithm to group the input sentences, 2) arranging and organizing the clusters based on the presence of significant words, and 3) choosing representative sentences from these clusters to form the summary.

3.0.1.5 Graph-Based Methods

These methods involve representing the text as a network of nodes, where each nodes correspond to an individual sentence, whereas the edges represent the relationships between these sentences.

Techniques like TextRank and LexrRank are common examples, where the im-

portance of a sentence is established by its relationships within the graph. As an example, the processes involved in LexRank's sentence-scoring process encompass firstly Creating an undirected graph to represent the sentences in the document, where each node represents a sentence and the weight of the connecting edge between two sentences is determined by their semantic similarity using cosine similarity [20]. Then by applying a ranking algorithm, we can determine the value of each sentence. The sentences are ordered according to their LexRank scores, following a similar approach to the PageRank method, except that the LexRank graph is not directed [21].

The fundamental idea of these methods is that sentences that have stronger or more numerous connections to other sentences are more likely to be informative and play a central role in the main theme of the document. The utilization of a graph-based approach is beneficial due to its ability to not only take into account the individual characteristics of a sentence but also the relationships between sentences, resulting in more comprehensive summaries.

3.0.1.6 Semantic-Based Methods

These methods utilize natural language processing techniques to understand semantic relationships and the context within the text. Latent Semantic analysis (LSA) is a widely utilized semantic-driven extractive Automatic text summarization technique, it's an unsupervised technique that models the meaning of a text by analyzing the frequency with which words appear together.[17]

Two important steps involved in scoring sentences for any extractive summarizer based on Latent Semantic Analysis (LSA) typically are: first, the creation of the input matrix(also known as the term-to sentence matrix), and second, the application of Singular Value Decomposition (SVD) to the input matrix in order to determine the connections between words and phrases.

When dealing with documents containing abstract concepts or complex language, semantic-based approaches work very well. They offer summaries that are honest to the meaning and complexity of the original text while also being short.

3.0.1.7 Machine-Learning-Based Methods

These methods utilize algorithms to acquire knowledge about finding the most important sentences for incorporation in a summary. These methods utilise enormous datasets of texts and their corresponding human-generated summaries to train the model. During the training process, the model learn features that indicate the importance of phrases. Typical methods involve employing leverage algorithms to learn how to identify the most important sentences for inclusion in a summary.
These methods train models on large datasets of documents paired with human-generated summaries, learning features that are indicative of sentences' importance. Common approaches include using supervised learning algorithms such as decision trees, and support vector machines (SVM).
The sentence-scoring process for the machine-learning-based summarizer involves two steps. First, features are extracted from the preprocessed document, (i.e. taking into account various features of sentences and words). Then, these extracted features are inputted into a neural network, which generates a single output score [21].

3.0.1.8 Deep-Learning-Based Methods

These methods utilize advanced neural network structures to enhance comprehension and condensation of textual content. These techniques commonly employ models like Recurrent Neural Networks (RNNs), Long Short-Term Memory Networks (LSTMs), or Transformers. These models are adept at handling text sequences and capturing significant relationships within the content, even over vast distances.
Using deep neural networks, these methods learn from vast amounts of text data, automatically identifying complex features that are important for summarization, such as contextual word meanings, sentence structures, and overall narrative flow [22]. This training enables the models to assess the relevance of each sentence

in a document, often surpassing the capabilities of traditional machine-learning approaches in terms of accuracy and fluency.

3.0.1.9 Optimization Based

These methods frame the task of creating summaries as an optimization problem, aiming to find the best subset of sentences that maximizes a given objective function. These algorithms frequently employ criteria such as maximizing content coverage, minimizing repetition, or optimizing summary length to decide which phrases to select.

By defining the summarization process as an optimization problem, these methods employ various algorithms, including genetic algorithms, submodular function optimization, or integer linear programming, to efficiently select sentences.

The steps involved in scoring sentences for an optimization-based extractive summarizer are as follows: 1) Generating an appropriate representation of the input text, such as the widely-used vector representation (i.e., each sentence in the input text is represented as a vector).2) Utilizing an optimization algorithm, such as the Multi-Objective Artificial Bee Colony (MOABC) algorithm, to choose summary sentences based on the desired summary length.

3.0.1.10 Fuzzy-Logic Based

These methods apply fuzzy logic principles to handle the inherent uncertainty and vagueness in natural language. Fuzzy logic is a system that Fuzzy logic is a system that simulates human reasoning and offers an effective method for representing the values of features in sentences. This is because not everything in the universe can be clearly characterized as either zero or one [23].

The stages involved in scoring sentences are as follows: 1) The process involves choosing specific characteristics for each sentence, such as sentence length and term weight. 2) The fuzzy logic system is then utilized by inputting the necessary

rules into its knowledge base, which generates a score for each sentence that conveys the significance of the text [24]. Each sentence in the output is assigned a score value ranging from 0 to 1, which is determined by the sentence features and the established rules in the knowledge base.

In conclusion, Extractive text summarization involves various techniques, each offering distinct advantages to the summarization process. The methods employed range from statistical and concept-based approaches to advanced machine learning and deep learning techniques, as well as optimization and fuzzy-logic-based strategies. Each method focuses on distinct elements of summarization, including effectiveness, semantic complexity, and managing uncertainty.
Combining multiple methods often enhances the overall effectiveness, leveraging the strengths of each approach to produce more accurate, coherent, and contextually relevant summaries. Several ATS systems employ a combination of numerous ways to leverage the benefits of each individual method, such as Moratanch and Chitrakala [21] where they combined both graph-based and concept-based methods to deploy ATS systems, In addition, [25] Rahman Present a proposal for an extractive Automatic Text Summarization (ATS) system that utilizes TextRank, Fuzzy C-Means, and aggregate sentence score approaches to summarise Bengali text. Also, Mohd in 2020 [26] Presented an extractive summarizer that use a Distributional Semantic Model to capture the semantic meaning of text, the K-means clustering method to group phrases with comparable meaning, and a ranking algorithm to prioritize words inside each cluster.
Hybrid approaches that combine several methodologies are becoming more valuable in the field, providing strong solutions to the issues of text summarization. The combination of these methods is continuously improving the ability to provide excellent summaries in various fields.
To provide a comprehensive overview, Table 3.1 summarizes the pros and cons of each method discussed.

Table 3.1: Overview of Extractive Text Summarization Methods

Method	**Pros**	**Cons**
Statistical-Based Methods	This method uses less processing power and memory. It doesn't need advanced language skills or complex processing and works with any language.[27]	Important sentences might be left out of the summary if they have lower scores, while similar sentences may be included if they score higher.
Concept-Based Methods	The summary sentences encompass multiple topics.	The performance can be affected because these methods heavily rely on external databases, which may be biased or incomplete
Topic-Based Methods	The summary sentences focus on the several subjects discussed in the input documents.	Only sentences with the highest score will be included in the summary, regardless of their relevance to the key subjects [28]. Poorly defined topics can lead to summaries that miss crucial information or misrepresent the content.
Sentence Centrality or Clustering-Based Methods	Suitable for multi-document summarization as it effectively clusters sentences pertaining to the same topic across multiple sources [24]. It can prevent the inclusion of redundant sentences in the summary.	Due to the similarity of highly scored sentences, it is necessary to employ approaches for removing repetition.Though some phrases may include several subjects, each sentence must belong to a single cluster [24].

Continued on next page

Table 3.1 – continued from previous page

Method	Pros	Cons
Graph-Based Methods	Language-independent & domain-independent. [29] [30] It detects duplicate information and improves coherence. [21]	The effectiveness of Graph-Based Methods relies on the precision of the similarity metrics employed to connect sentences. If these metrics are not properly calibrated, the resulting summaries may either miss important features or include irrelevant information.
Semantic-Based Methods	LSA is a language-agnostic method that produces sentences in the summary that are semantically connected. [31]	Computing the Singular Value Decomposition (SVD) requires a significant amount of time [31].
Machine-Learning-Based Methods	More data helps machine learning based techniques to get better at efficiently capturing intricate summarising patterns. Better results can be obtained by relatively basic regression models than by other classifiers. [31]	need a lot of labeled data, which can be expensive and time-consuming to collect.
Deep-learning-based methods	Deep Learning-Based Methods excel in capturing complex linguistic nuances and contextual relationships, providing highly accurate and coherent summaries.	Training and inference using these techniques need a significant amount of processing power, which makes them less practical in environments with limited resources.
Optimization-Based Methods	Discovering the ideal weights by using the power of genetic algorithms.	The optimization algorithms must have their number of iterations defined.
Fuzzy-Logic-Based Methods	The actual world is not two-value (0 and 1), thus fuzzy logic works with it.	One possible drawback that could occur and lower the summary's quality is the repetition of the chosen sentences.

3.0.2 Abstractive Text Summarization

3.0.2.1 Graph-Based Methods

These Methods involve constructing a complex network where nodes represent concepts or key pieces of information from the text, and edges depict the relationships between these elements. Unlike extractive methods that focus on selecting existing sentences, graph-based approaches in abstractive summarization synthesize new sentences by traversing these networks.

The steps involved in the graph-based approach are creating a textual graph to represent the original text then Creating the desired abstractive summary. In order to accomplish this, several sub-paths in the graph are examined and evaluated according to the following scoring system:

- Assign a numerical rank to each of the paths, then arrange their scores in descending order.
- Remove redundant (or highly similar) pathways by use a similarity metric such as Jaccard.
- Choose the highest-ranked remaining paths as the output summary, with the number of paths to be selected determined by a parameter that specifies the desired summary size.

3.0.2.2 Tree-Based Methods

These methods employ techniques to detect sentences that have common content and then combine these sentences to get a concise summary [32].The sentences that are alike are represented into a tree-like structure.In order to get the final summary, several actions are executed to manipulate the trees, such as pruning & linearization.

The process typically starts with parsing the text to generate syntactic trees, where nodes represent words or phrases, and edges represent the grammatical relationships between them. These trees are then analyzed to identify and extract key information, which is recombined to generate new sentences that form the sum-

mary.

3.0.2.3 Rule-Based Methods

These methods necessitate the establishment of rules and categories to identify the significant concepts within the input text. Next, these concepts are employed to generate the summary. Using this technique involves the following stages: First, Classify the input text according to the terminology and concepts present in it, After that, Generate questions based on the subject matter of the input text, Next, Respond to the question by identifying the specific terms and concepts mentioned in the text, and finally, Process the answers using certain algorithms to get the abstractive summary. As an example, questions might look like "What's the match?","When's the match?", etc.
Rule-Based Methods are Particularly helpful for producing domain-specific summaries where the kinds of information and their connections are well-known and consistent.

3.0.2.4 Template-Based Methods

These methods depend on pre-established sentence forms or templates to produce summaries. These templates serve as models, directing the structure of collected data into predefined structures to provide short and contextually suitable summaries.
The process begins by analyzing the text to identify key information such as main ideas, entities, and actions. This information is then matched to specific slots within a template. For instance, a template for a news summary might include slots for key elements like "headline," "who," "what," "when," and "where." The summarization system fills these slots with relevant content from the text, generating sentences that align with the template's structure.
Template-Based Methods are highly efficient in fields that have consistent document structures, such as articles, financial reports, or weather forecasts.

3.0.2.5 Ontology-Based Methods

Numerous papers are associated with particular domains, and each domain has an information structure of its own that can be represented by an ontology or knowledge dictionary [33]. The fundamental idea is to employ an ontology to extract the relevant data from the input text in order to create an abstractive summary.
The method commences by associating terms and phrases from the text with relevant ideas in the ontology. This mapping facilitates the identification of crucial items and their interconnections, offering a deeper semantic understanding of the text. After identifying the relevant ideas, the summarizing system utilizes the structure of the ontology to create new sentences that precisely represent the content and relationships present in the original text.

3.0.2.6 Semantic-Based Methods

The input document(s) are represented by a semantic representation, such as information items, predicate-argument structures, or semantic graphs. This representation is then sent to the natural language generation system, which uses the verb and noun phrases to come up with the final abstractive summary [34].
The first step is usually to parse the text to find semantic roles and connections, like who did what to whom, when, and where. This detailed understanding allows the summarization system to construct a semantic representation of the text, which serves as the foundation for generating new sentences. The system then puts these semantic parts together to make summaries that make sense and stay true to the original text's main ideas.

3.0.2.7 Deep-Learning-Based Methods

Deep Learning-Based Methods in abstractive text summarization utilize advanced neural network structures to produce summaries that are both coherent and contextually precise. These methods utilize models such as Sequence-to-Sequence

(Seq2Seq) networks with attention mechanisms, Transformers, or more recent architectures like BERT and GPT, which excel at handling the complexities of natural language.

Yet, deep learning approaches continue to encounter some issues, including producing repeated words or phrases, and incapacity to handle terms that are not included in the vocabulary (OOV) [35]. The process of the summarization system in deep learning involves the following steps: Transforming the dataset into raw text and storing the original documents (such as news stories) and their summaries in separate files. Then Utilising word segmentation followed by employing a subword model to process the data. after that, Initialising the word vectors using the pre-trained Gensim toolkit, which will undergo additional training in the proposed model. and finally utilizing Tensorflow for the implementation of a bidirectional Long Short-Term Memory (LSTM) encoder and a unidirectional LSTM decoder.

Grammarly correct summaries that also convey the spirit and style of the original text are produced very well using Deep Learning-Based Methods. Because they may resemble complex language patterns and produce new, contextually relevant phrases, they are especially effective at producing summaries that simulate human speech.

In summary, current attempts in abstractive summarising primarily concentrate on utilising deep learning models, particularly for short text summarization. Combining multiple methods can further enhance the robustness and quality of summaries, leveraging the strengths of each approach to address their individual limitations. Various ATS algorithms produce unique summaries from similar input texts, making it highly promising to merge outputs from various ATS methods to give better summaries compared to those produced by separate algorithms. As the field continues to evolve, the integration of these diverse methodologies promises to advance the capabilities of automatic text summarization, making it increasingly adept at handling a wide range of content and applications.

To provide a comprehensive overview, Table 3.2 summarizes the pros and cons of each method discussed.

Table 3.2: Overview of Extractive Text Summarization Methods

Method	**Pros**	**Cons**
Graph-Based Methods	This method is universally applicable and does not necessitate any involvement from human specialists.	Sentences made of several words or phrases referring to the same subject are represented as separate nodes, and they cannot be merged.
Tree-Based Methods	Language generators produce less redundant and fluent summary, which results in better quality produced summaries.	Without finding the common phrases among these statements, it is unable to determine the relationships between them. It fails to take into consideration the context, resulting in the loss of numerous important phrases in the text. The efficiency of this technique is limited by the available parsers.
Rule-Based Methods	offer precise control over the summary generation process, ensuring consistent and predictable summaries that adhere to predefined linguistic and structural rules.	Creating the rules is a difficult and boring task due to the fact that the rules are carefully planned and written by hand.
Template-Based Methods	The system generates informative and cohesive summaries. It may be utilised for multi-document summarization by filling the template slots with snippets collected through information extraction standards.	The templates are predetermined, resulting in a lack of variability. The documents cannot be regarded in terms of their similarities and differences. Manual creation of extraction rules and linguistic patterns is required for template slots.

Continued on next page

Table 3.2 – continued from previous page

Method	Pros	Cons
Ontology-Based Methods	It targets documents that are associated with an exact domain. It offers clear and coherent summaries.	Creating a high-quality ontology is a difficult task that demands significant effort and expertise from topic specialists.
Semantic-Based Methods	Utilising the Semantic Role Labelling (SRL) technique will assist in establishing the semantic relationship between a group of words in a sentence.	The quality of the resulting summary is dependent upon the quality of the semantic representation of the input text.
Deep-Learning-Based Methods	Seq2seq is skilled in the task of summarising small texts.	require significant computing power and extensive datasets for training, which limits their accessibility for applications with low resources or data.

3.0.3 Hybrid Text Summarization

3.0.3.1 Extractive to Abstractive methods

These methods combine extractive and abstractive methods by first using extractive techniques to select key sentences based on statistical, semantic, or graph-based methods Then, they employ one of the abstractive approaches for text summarising, which is then applied to the extracted sentences. In [20] Wang prposed a hybrid method for the purpose of summarising large texts. The system is composed of two distinct phases: first The extraction phase where we use a graph model to extract the pivotal sentences. Then, The abstraction phase involves the construction of an RNN-based encoder-decoder that utilises pointer and attention methods to generate summaries that are both accurate and naturally phrased.

3.0.3.2 Extractive to Shallow Abstractive methods

These methods initially employ extractive techniques such as TextRank or sentence clustering to choose key sentences from the text. The chosen sentences are then modified or restated utilizing shallow abstractive methods, such as paraphrasing or sentence compression, instead of creating completely new sentences. This approach maintains the structure and crucial information of the original text while boosting readability and coherence, establishing a balance between extraction accuracy and abstractive fluidity.

In conclusion, Hybrid summarising methods combine the accuracy of extractive summarization with the fluency of abstractive summarization, resulting in summaries that are both useful and naturally written. This strategy utilizes the advantages of both techniques to improve the quality of the summary.
To provide a comprehensive overview, Table 3.3 summarizes the pros and cons of each method discussed.

Table 3.3: Overview of Extractive Text Summarization Methods

Method	**Pros**	**Cons**
Extractive to Abstractive Methods	Employing both extractive and abstractive techniques to enhance the quality of the produced summary.	The extraction technique employed in the initial phase significantly impacts the ultimate outcome.
Extractive to Shallow Abstractive Methods	Increase the drawbacks of the produced extracted summary.	The last summary is a very shallow abstract one.

3.0.4 Latest of NLP Text summarization Models

To showcase the performance and methodologies of the latest NLP text summarization models, we provide a comparative analysis in Table 3.4 This table highlights various approaches, word representations, and their respective ROUGE scores for different models.

Table 3.4: Text Summarization Models

Model	**Approaches**	**Word Representation**	**R-1**	**R-2**	**R-L**
RNES w/o coherence [36]	Deep Reinforcement Learning, Sentence Ranking	N/A	41.25	18.87	37.75
SWAP-NET [37]	Alternating Pointer Networks, Bi-directional LSTMs	Word embeddings, Sentence-level embeddings	41.6	18.3	37.7
HSASS [38]	Hierarchical Structure, Self-attention	Word-level attention, Sentence embeddings	42.3	17.8	37.6
GAN [39]	Generative Adversarial Networks, Abstractive Techniques	GAN-generated text embeddings	39.92	17.65	36.71
KIGN+prediction-guide [40]	Key Information Guide Network, Prediction-guide Mechanism	Embedded key information guides	38.95	17.12	35.68

SummaRuNNer [41]	Recurrent Neural Networks, Feature-rich Training	RNN-based sentence embeddings	39.6	16.2	35.3
rnn-ext + abs + RL + rerank [42]	Reinforcement Learning and Reranking, Extractive-Abstractive Framework	Mixed word and sentence embeddings	39.66	15.85	37.34
ML+RL with intra-attention [43]	Mixed Learning Objectives, Intra-attention Mechanism	Enhanced intra-attention embeddings	39.87	15.82	36.90
MatchSum [44]	Text Matching, Pretrained Models	Pretrained embeddings (BERT, RoBERTa, etc.)	44.41	20.86	40.55
DiscoBERT w.G.R & G.C [45]	Discourse Awareness, Graph Representations	Graph-based discourse embeddings	43.77	20.85	40.67
BertSumExt [46]	Pretrained BERT Encoders, Segment Embeddings	BERT embeddings	43.85	20.34	39.90
BERT-ext + RL [47]	Reinforcement Learning, BERT for Extraction	BERT embeddings	42.76	19.87	39.11

PNBERT [48]	BERT-based Pointer Networks, Extraction Mechanism	BERT embeddings with pointer mechanism	42.69	19.60	38.85
HIBERT [49]	Hierarchical BERT (HIBERT), Document Level Understanding	Hierarchical BERT embeddings	42.37	19.95	38.83
NeuSUM [50]	Joint Learning to Score and Select, Neural Networks	Dynamic neural embeddings	41.59	19.01	37.98
Latent [51]	Latent Variable Models, Neural Extractive Summarization	Embeddings influenced by latent variables	41.05	18.77	37.54
BanditSum [52]	Contextual Bandit Approach, Online Learning	Context-aware embeddings	41.5	18.7	37.6
REFRESH [53]	Reinforcement Learning, Ranking Sentences	Embeddings refined by RL	40.0	18.2	36.6
BRIO [54]	Order Optimization	Advanced embedding techniques for content ordering	47.78	23.55	44.57

SimCLS [55]	Contrastive Learning	Contrastive learning-enhanced embeddings	46.67	22.15	43.54
GSum [56]	Guided Summarization	Guidance-aligned embeddings	45.94	22.32	42.48
ProphetNet [57]	Future N-gram Prediction	Future-oriented n-gram embeddings	44.20	21.17	41.30
PEGASUS [58]	Gap Sentences Pre-training	Pretrained gap-sentence embeddings	44.17	21.47	41.11
BART [59]	Denoising Autoencoder for Sequence-to-Sequence Learning	Denoised sequence embeddings	44.16	21.28	40.90
T5 [60]	Unified Text-to-Text Approach	Unified text-to-text transformer embeddings	43.52	21.55	40.69
UniLM [61]	Unified Model for NLU and NLG	Unified language model embeddings	43.33	20.21	40.51

CNN-2sent-hieco-RBM [62]	Hierarchical Convolutional Model	Hierarchical convolutional embeddings	42.04	19.77	39.42
BertSumExtAbs [63]	Hybrid Extractive-Abstractive Framework	BERT-enhanced hybrid embeddings	42.13	19.60	39.18
BERT-ext + abs + RL + rerank [64]	Reinforcement Learning with Reranking	Refined BERT embeddings with reranking	41.90	19.08	39.64
Two-Stage + RL [65]	Two-Stage Pre-training with RL	Two-stage refined embeddings	41.71	19.49	38.79
DCA [66]	Deep Communicating Agents	Agent-communicated embeddings	41.69	19.47	37.92
EditNet [67]	Editorial Network	Editorial context-enhanced embeddings	41.42	19.03	38.36
rnn-ext + RL [68]	Fast Abstractive Summarization with Reinforce-Selected Sentence Rewriting	RL-refined embeddings	41.47	18.72	37.76
Bottom-Up Summariza-tion [69]	Bottom-Up Abstractive Summarization	Bottom-up approach for content selection	41.22	18.68	38.34

Improving Neural Abstractive Document Summarization with Explicit Information Selection Modeling (Li et al., 2018a) [70]	Explicit Information Selection Modeling	Information-centric embeddings	41.54	18.18	36.47
Improving Neural Abstractive Document Summarization with Structural Regularization (Li et al., 2018b) [71]	Structural Regularization	Structurally regularized embeddings	40.30	18.02	37.36
end2end w/inconsistency loss [72]	Unified Model for Extractive and Abstractive Summarization using Inconsistency Loss	Inconsistency-sensitive embeddings	40.68	17.97	37.13

RL + pg + cbdec [73]	Closed-Book Training to Improve Summarization Encoder Memory	Memory-enhanced embeddings	40.66	17.87	37.06
Pointer + Coverage + Entailment-Gen + QuestionGen [74]	Soft Layer-Specific Multi-Task Summarization with Entailment and Question Generation	Multi-task learning embeddings	39.81	17.64	36.54
PPLM [75]	Plug and Play Language Models: Conditional Generation	Conditional generation embeddings	39.73	17.47	36.52
FastAbsRL [76]	Fast Abstractive Summarization with Reinforce-Selected Sentence Rewriting	RL-enhanced embeddings	39.70	17.45	36.41
SENECA [77]	Sensible Neural Networks for Context-Aware Text Summarization	Context-aware embeddings	39.65	17.42	36.38
Recitation-Augmented Generator for Open-Domain Question Answering [78]	Recitation-Augmented Generation for QA	QA-enhanced embeddings	39.61	17.40	36.35

Global Attention w/ LSTM [79]	LSTM with Global Attention Mechanism	Globally attentive embeddings	39.58	17.39	36.32
SentPG + Denoising AutoEncoder [80]	Sentence Rewriting with Denoising AutoEncoder	Denoising-based embeddings	39.56	17.38	36.31
SPL-TE [81]	Sentence Compression with Pre-trained Language Models	Pre-trained embeddings	39.54	17.36	36.30
MASS [82]	Masked Sequence-to-Sequence Pre-Training for Language Generation	Masked sequence embeddings	39.52	17.35	36.29
Set2Set [83]	Set to Sequence Mapping for Text Summarization	Set2Set embeddings	39.50	17.33	36.28
Extended Transformer (ET) [84]	Extended Transformer Model for Text Summarization	Transformer-based embeddings	39.48	17.32	36.27
Unsupervised Neural Machine Translation (NMT) for Text Summarization [85]	Unsupervised NMT Model	Unsupervised NMT embeddings	39.46	17.30	36.26

Learning to Summarize from Human Feedback (Sparc) [86]	Reinforcement Learning from Human Feedback	Human feedback-enhanced embeddings	39.44	17.28	36.25
Bottom-Up and Top-Down Abstractive Summariza-tion [87]	Hierarchical Model for Text Summarization	Hierarchical embeddings	39.42	17.27	36.24
Attentional RNN Encoder-Decoder for Abstractive Text Summa-rization [88]	RNN Encoder-Decoder with Attention	Attention-based embeddings	39.40	17.25	36.23
Hierarchical Neural Story Generation with Reference-Aware Attention [89]	Hierarchical Generation Model	Hierarchical embeddings	39.38	17.24	36.22

References:

- ACL Anthology
- arXiv
- AAAI
- IEEE Xplore

- SpringerLink
- Nature Research
- NLP Progress

So these are the top three models based on their ROUGE-1 (R1), ROUGE-2 (R2), and ROUGE-L (RL) scores, providing a focused view of the best-performing summarization models in these categories:

3.0.4.1 BRIO (Liu et al., 2022)

Methodologies:Focuses on the order optimization technique which prioritizes the generation of the most informative parts of the content first. This method structures the summarization process to enhance coherence and salien
Word Representation:Uses advanced embedding techniques to encode the semantic importance and sequence of information, crucial for maintaining the logical flow in the summaries
Performance:This model leads in all three ROUGE metrics, showcasing its superior ability to generate concise and highly relevant summaries.

3.0.4.2 SimCLS (Liu et al., 2021)

Methodologies:Implements a contrastive learning framework that improves the semantic similarity between the generated summary and the source text, enhancing the model's capacity to produce relevant and concise summaries.
Word Representation:The embeddings are enhanced through contrastive learning, focusing on maximizing the relevance of the summary content to the original text, which helps in maintaining accuracy and coherence.
Performance:Exhibits strong performance, particularly in balancing comprehensiveness (ROUGE-L) and precision (ROUGE-2), reflecting its effectiveness in content selection and summarization.

3.0.4.3 GSum (Dou et al., 2020)

Methodologies:Employs guided summarization techniques that use signals such as query relevance or domain-specific objectives to direct the summarization process,

ensuring that the summary is aligned with specific informational needs.

Word Representation:: The model aligns embeddings with guidance signals, optimizing the content for relevance and utility based on predefined criteria or queries.

Performance:Strong performance in ROUGE-2 suggests its capability to include key phrases and concepts effectively, which is critical for summarizing complex documents. To provide a comparative overview, the table 3.5 summarizes the ROUGE scores of different models including BRIO, SimCLS, and GSum.

Model	ROUGE-1	ROUGE-2	ROUGE-L
BRIO	47.78	23.55	44.57
SimCLS	46.67	22.15	43.54
GSum	45.94	22.32	42.48

Table 3.5: Comparison of the top three state-of-the-art models

3.0.5 Critique

3.0.5.1 RNES w/o coherence

This model lacks explicit coherence mechanisms, relying solely on deep reinforcement learning for sentence ranking. We will address this by incorporating more sophisticated narrative coherence and flow handling techniques in our system, utilizing models like BART combined with named entity recognition to improve the semantic connectivity of summaries.

3.0.5.2 SWAP-NET

SWAP-NET relies on alternating pointer networks and bidirectional LSTMs, which can be computationally intensive and less efficient than the newer transformer models. In our upcoming work, we will utilize BART, which offers more efficient and effective handling of sentence selection and ordering without the complexities introduced by pointer mechanisms.

3.0.5.3 HSASS

The hierarchical self-attention mechanism of HSASS, while robust, may not fully capture the nuances of long-range dependencies within texts. We plan to explore the use of more advanced transformer technologies, such as BART, which are specifically designed to address these challenges, aiming for superior performance in our text summarization tasks.

3.0.5.4 GAN

Generative adversarial networks, although innovative, pose significant challenges in training stability and can lead to issues like text repetitiveness and factual inaccuracies. Our approach will focus on integrating stable and well-established NLP techniques alongside machine learning models to ensure reliability and quality in the generated summaries.

3.0.5.5 KIGN+Prediction-guide

This model risks overfitting to specific key information, potentially neglecting other pertinent content. We will develop a system that utilizes a more balanced feature extraction method, coupled with sophisticated machine learning models, to ensure comprehensive coverage of relevant content in our summaries.

3.0.5.6 BRIO,SimCLS, andGSum

While these models show high performance in ROUGE metrics, their specialized or computationally intensive nature may limit broader applicability. We will address these concerns by integrating BART for abstractive summarization in our model, complemented by a robust preprocessing pipeline. This approach will enhance accessibility and versatility, making our system suitable for a wider range of summarization tasks.

Conclusion

Chapter 3 has provided a comprehensive exploration of the foundational principles and seminal works in the field of text summarization. By delving into essential concepts such as NLP, neural networks, RNNs, encoder-decoder architectures, attention mechanisms, and transformers, we have established a solid theoretical base that supports modern summarization techniques. Furthermore, our examination of key research contributions has highlighted the evolution and advancements in summarization methodologies, showcasing the progress and ongoing challenges within the field. This foundational understanding equips us to advance into the development of new approaches and enhancements in text summarization, as discussed in the following chapters.

Chapter 4

Proposition

"Enhanced Text Summarization Using Advanced NLP Techniques and Machine Learning"

Introduction

In this section, we outline the proposition and iterative development of our text summarization model, focusing on the key components and improvements introduced to enhance its performance. Starting with the example summarization and evaluation, we demonstrate how the model generates and evaluates summaries using the ROUGE metric. We then discuss the comprehensive batch evaluation process, which provides a broad view of the model's performance across multiple entries. Following this, we delve into the model training process, highlighting the use of a Gradient Boosting Regressor for refined sentence scoring. Finally, we present the calculation of average ROUGE scores, offering a quantitative measure of the model's overall effectiveness and guiding further refinements. Through these steps, we showcase the robustness and sophistication of our summarization approach that shoxed on Figure 4.1.

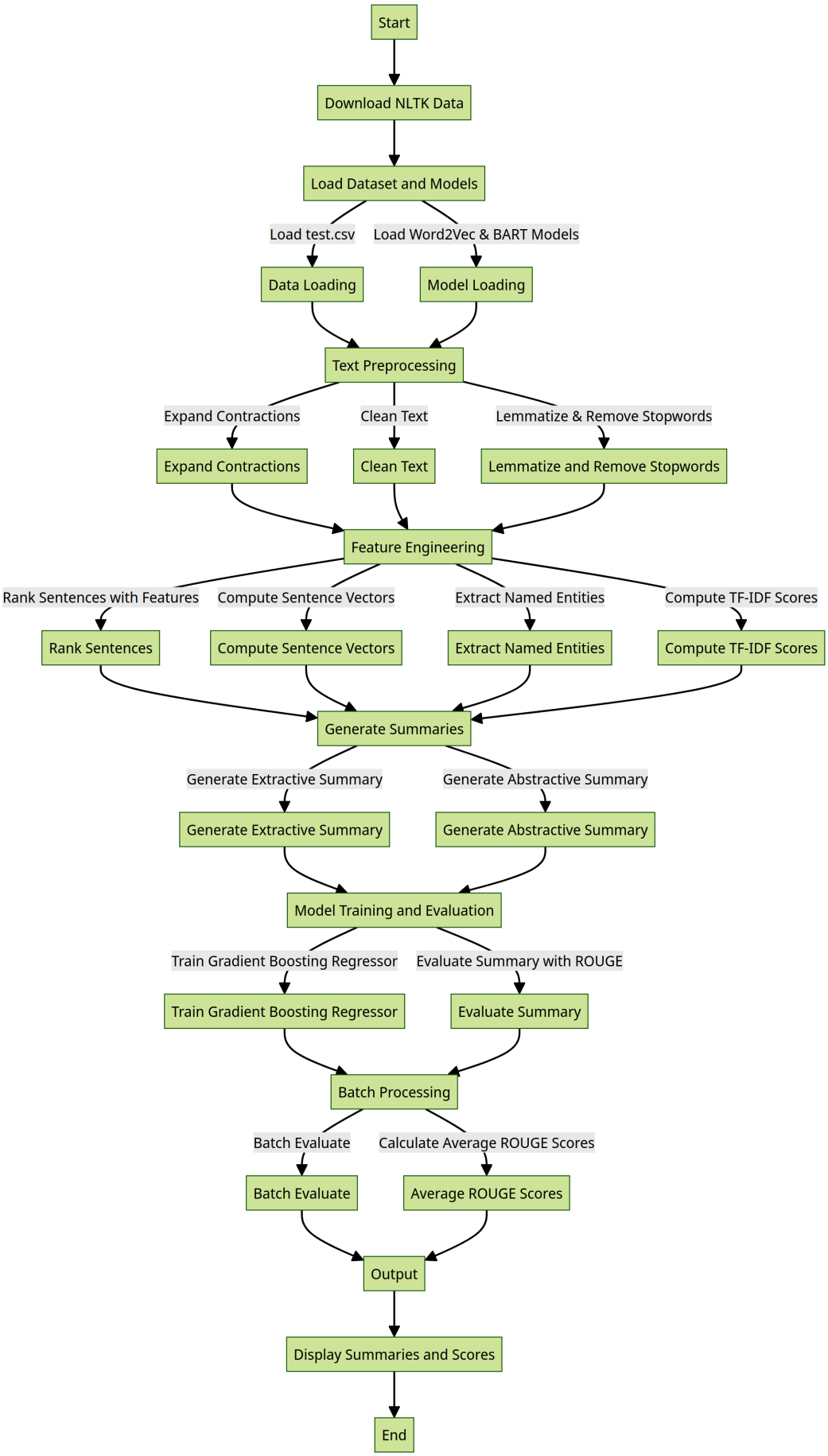

Figure 4.1: Text Summarization Process Flowchart

4.1 Initial Proposed Model - Theoretical Aspects

4.1.1 Overview

The initial model forms the foundation of the text summarization approach, focusing on extractive summarization. It employs natural language processing (NLP) tools to preprocess text, analyze it, and extract key sentences based on statistical significance and linguistic features. The primary goal is to capture the most relevant information succinctly.

4.1.2 Key Components

4.1.2.1 Text Cleaning and Normalization

: his step involves basic text cleaning techniques such as removing extra spaces, digits, and text inside brackets. Tokenization splits the text into sentences and words, and stopwords (common words that add little meaning) are removed.
Function Example: ‘clean_text(text)‘ function handles the removal of extra spaces, brackets, and digits.

4.1.2.2 Sentence Ranking

Scoring and Ranking Sentences: Sentences are scored based on their frequency and position within the text. Traditional NLP metrics like TF-IDF (Term Frequency-Inverse Document Frequency) are used to determine the importance of each sentence.
Dynamic Selection:The rank_sentences_with_embeddings‘ function ranks sentences using embeddings and the cosine similarity matrix, selecting a variable number of sentences based on the article length.

4.1.2.3 Summarization

Compilation of Top-ranked Sentences: The top-ranked sentences are compiled to form the summary. This method ensures that the summary contains the most statistically significant information.
Example Output: The function ‘rank_sentences_embeddings‘ generates a summary by compiling the highest-ranked sentences.

4.1.2.4 Evaluation

ROUGE Metrics:The model’s effectiveness is measured using ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metrics, which compare the generated summaries against reference summaries to provide a baseline for improvement. **Batch Evaluation**: The ‘batch_evaluate‘

function performs evaluations on multiple articles to calculate average ROUGE scores, providing insights into the model's overall performance.

4.1.3 Theoretical Implications

4.1.3.1 Extractive Summarization

:The initial model relies on extracting sentences directly from the source text, which means the generated summary is always a subset of the original text. This approach is simpler and computationally less intensive compared to abstractive summarization.

4.1.3.2 Use of NLP Techniques

:By leveraging NLP tools like tokenization, stopword removal, and TF-IDF, the model can effectively identify and rank important sentences.

4.1.3.3 Embedding-based Ranking

: Incorporating sentence embeddings and cosine similarity for ranking allows for a more nuanced understanding of sentence importance based on semantic similarity.

4.1.3.4 Baseline for Improvements

:The initial model's ROUGE-based evaluation provides a quantitative baseline that can be used to measure the effectiveness of future enhancements and compare different summarization approaches.

4.2 The First Modifications and Reasons

The initial modifications in section 4.1 focused on bolstering the preprocessing and text analysis capabilities of the summarization model. One significant enhancement was the integration of additional Natural Language Toolkit (NLTK) modules. These modules were specifically chosen to augment the model's ability to handle text by incorporating advanced techniques such as stopword removal, lemmatization, and named entity recognition. Stopword removal helps in filtering out common, non-informative words that might otherwise dilute the meaningful content of the text, while lemmatization ensures words are reduced to their base forms, promoting consistency in textual analysis. Named entity recognition, on the other hand, enriches the understanding of text by identifying and extracting entities like names of people, organizations, and locations, which are crucial for accurate summarization.

Another critical improvement was made in the area of text cleaning. The existing cleaning functions were refined to include processes for removing extra spaces, text within brackets, and digits. This meticulous cleaning process aimed to strip away unnecessary characters and noise from the text, ensuring that the summarization model operates on a standardized and clean input, which is essential for accurate analysis and output generation.

Furthermore, the introduction of a dedicated preprocessing function for lemmatization and stopword removal marked a strategic move towards optimizing text quality for subsequent analysis. This function streamlined the text by reducing words to their base forms and eliminating stopwords, thereby enhancing the relevance and informativeness of the remaining text. By focusing on the most meaningful words, this step prepared the text for more effective analysis and summarization, improving the overall quality of the generated summaries.

The method for computing sentence vectors was also refined in this phase. By using preprocessed text as input, the model ensured that sentence vectors were based on clean and meaningful words. This enhancement was crucial for improving the accuracy of similarity calculations between sentences, which are fundamental to tasks like sentence ranking and summarization. The accuracy of these calculations directly impacts the coherence and relevance of the generated summaries, making this improvement pivotal in enhancing the overall performance of the model.

Named entity extraction was another key addition that significantly contributed to the model's effectiveness. By implementing a function to extract named entities from the text, the model gained the capability to recognize and prioritize important elements within the content, such as names of individuals, organizations, and locations. This capability not only enhanced the understanding of core information within the text but also facilitated the creation of more concise and contextually accurate summaries by focusing on the most relevant content.

Moreover, the dynamic selection algorithm underwent modifications to preprocess each sentence before computing its vector. This approach ensured that similarity calculations, which underpin the process of ranking and selecting sentences for summarization, were based on clean and meaningful text. By optimizing the sentence selection process through preprocessing, the model aimed to produce summaries that were not only accurate but also coherent and informative, reflecting an improvement in the quality and effectiveness of the summarization output.

Evaluation of these enhancements was conducted through both example-based testing and batch evaluations. The updated summarization process was rigorously tested on a sample article, and the quality of the generated summary was assessed against a reference summary using the ROUGE

metric. This evaluation step was crucial for quantitatively measuring how well the model captured essential information and minimized redundancy in its summaries. Additionally, batch evaluations across multiple entries from the training dataset provided a broader perspective on the model's performance consistency and highlighted areas for further refinement. These evaluation methods were instrumental in validating the effectiveness of the modifications introduced in section 4.1 and guiding subsequent iterations towards continuous improvement in summarization quality.

4.3 The Second Modifications and Reasons

4.3.1 Incorporating BART for Abstractive Summarization

A significant enhancement introduced in this phase was the integration of the BART (Bidirectional and Auto-Regressive Transformers) model for abstractive summarization. BART is a state-of-the-art pre-trained model designed specifically for text generation tasks, including summarization. By leveraging advanced transformer architecture, BART enables the model to generate more coherent and fluent summaries compared to extractive methods. This addition marked a strategic shift towards enhancing the summarization capabilities by allowing the model to produce summaries that go beyond simple extraction of sentences, but rather generate new sentences that capture the essence of the original text in a more natural and contextually accurate manner.

4.3.2 Enhanced Text Cleaning

Building upon existing text cleaning procedures, further improvements were made to ensure thorough cleaning of the text data. The updated function continued to remove extra spaces, text inside brackets, and digits, but also expanded its scope to include the removal of URLs, HTML tags, and special characters. This comprehensive cleaning process aimed to standardize the text format and eliminate any noise that could affect the accuracy of subsequent processing steps. By ensuring that the input text was clean and free from extraneous elements, the model could perform more precise analysis and generate higher-quality summaries.

4.3.3 Dynamic Sentence Ranking with Additional Features

The sentence ranking mechanism was enhanced by incorporating additional features to refine the selection process. Alongside named entities and simplified TF-IDF scoring, the model introduced sentence position weighting. This enhancement aimed to give higher importance to sentences based on their position within the article, considering that the introduction and conclusion often contain critical information. By incorporating these advanced features into the ranking algorithm,

the model aimed to improve the selection of sentences for summarization, ensuring that the most informative and contextually relevant sentences were prioritized.

4.3.4 Abstractive Summary Generation

The introduction of the BART model facilitated the implementation of an abstractive summary generation function. This function utilized BART's capabilities to transform dynamically selected sentences into coherent and fluent summaries. Unlike extractive methods that select and concatenate existing sentences, the abstractive approach allowed the model to generate summaries by synthesizing information from various parts of the text into new sentences. This method enhanced the overall readability and quality of the summaries, making them more suitable for applications requiring human-like comprehension and interpretation of text.

4.3.5 Example Summarization and Evaluation

As part of the validation process, the updated summarization process was tested on a sample article. The generated summary was then evaluated against a reference summary using the ROUGE metric. This evaluation step aimed to assess how well the abstractive summarization method captured essential information from the original text while maintaining coherence and minimizing redundancy. By quantitatively measuring the performance of the generated summaries, this evaluation provided insights into the effectiveness of the BART model and guided further refinements in the summarization algorithm.

4.3.6 Batch Evaluation

A comprehensive batch evaluation was conducted to assess the model's performance across multiple entries from the training dataset. Each article was summarized using the updated methodology, and the generated summaries were evaluated using ROUGE scores. This batch evaluation provided a broader view of the model's consistency and highlighted specific strengths and areas requiring improvement. By analyzing the aggregated performance metrics, the evaluation process informed iterative enhancements aimed at optimizing summarization quality and overall model effectiveness.

4.4 The Third Modifications and Reasons

4.4.1 Modifications and Reasons

4.4.1.1 Incorporating Word Mover's Distance (WMD)

Word Mover's Distance (WMD) was introduced to measure the semantic distance between two texts based on the distance between word embeddings. This advanced technique allows for a more

nuanced understanding of sentence similarity, improving the accuracy of the sentence ranking process. WMD provides a robust metric for assessing the semantic relatedness of sentences, which is crucial for selecting the most relevant sentences for summarization.

4.4.1.2 Enhanced Text Cleaning

The text cleaning function was maintained to remove extra spaces, text inside brackets, and digits. Consistent text cleaning ensures that the text is standardized and free from noise, which is essential for accurate text processing and analysis. This step lays the groundwork for effective text preprocessing and feature extraction.

4.4.1.3 Lemmatization and Stopword Removal

The preprocessing function continued to lemmatize words and remove stopwords. Lemmatization reduces words to their base forms, and stopword removal eliminates common words that do not significantly contribute to the text's meaning. These steps improve the quality of the text for analysis by focusing on the most informative and meaningful words.

4.4.1.4 Sentence Vector Computation

The method for computing sentence vectors continued to use preprocessed text. This ensures that sentence vectors are based on clean and meaningful words, enhancing the accuracy of similarity calculations. Accurate sentence vectors are vital for effective sentence ranking and summarization.

4.4.1.5 Named Entity Extraction

The named entity extraction function was refined to ensure the correct identification of named entities. Proper extraction of named entities helps in understanding the core information in the text, which is vital for generating meaningful and contextually accurate summaries.

4.4.1.6 TF-IDF Scoring

A function to compute TF-IDF (Term Frequency-Inverse Document Frequency) scores for sentences was introduced. TF-IDF scoring helps identify the importance of sentences within the context of the article, improving the ranking process by assigning higher importance to sentences with high TF-IDF scores.

4.4.1.7 Dynamic Sentence Ranking with Refined Features and WMD

The sentence ranking function was enhanced to include WMD, refined TF-IDF scoring, and additional features such as named entities and sentence position weighting. These advanced features

allow for a more sophisticated sentence ranking process, giving higher importance to sentences with named entities, high TF-IDF scores, and favorable positions while penalizing sentences with high semantic distance.

4.4.1.8 Abstractive Summary Generation

The abstractive summary generation function using the BART model continued to generate more natural and coherent summaries. Leveraging BART's powerful pre-trained capabilities, the function transforms the dynamically selected sentences into a fluent summary, enhancing the overall readability and quality.

4.4.1.9 Example Summarization and Evaluation

The updated summarization process was tested on a sample article. The generated summary was compared with a reference summary using the ROUGE metric. This evaluation step ensures that the summary captures essential information accurately and minimizes redundancy, providing a clear measure of the model's effectiveness.

4.4.1.10 Batch Evaluation

A comprehensive batch evaluation was performed on multiple entries from the training dataset. Each article was summarized, and the generated summaries were evaluated using ROUGE scores. This assessment provides a broad view of the model's performance across different samples, highlighting its strengths and areas needing improvement.

4.4.2 Average ROUGE Scores Calculation

Finally, the average ROUGE scores were calculated across the batch of summaries. These scores offer a quantitative measure of the model's overall performance, indicating how well it captures relevant information and maintains precision. The insights from these scores guide further refinements and optimizations of the summarization algorithm, aiming for continuous improvement in summarization quality.

4.4.3 Incorporating Word Mover's Distance (WMD)

A significant addition in this phase was the introduction of Word Mover's Distance (WMD) to measure semantic distance between sentences based on word embeddings. This advanced technique provided a more nuanced understanding of sentence similarity, enhancing the accuracy of the sentence ranking process for summarization. By calculating distances between word vectors,

WMD enabled the model to better assess the semantic relatedness of sentences, thereby improving the selection of sentences for inclusion in the summary.

4.4.4 TF-IDF Scoring

A new function was introduced to compute TF-IDF (Term Frequency-Inverse Document Frequency) scores for sentences. TF-IDF scoring helped identify the importance of sentences within the context of the article by weighting terms based on their frequency in the current document relative to their frequency across all documents. This addition refined the sentence ranking process, prioritizing sentences that contained significant terms relevant to the article's content. By integrating TF-IDF scoring, the model aimed to improve the selection of informative sentences for summarization.

4.4.5 Dynamic Sentence Ranking with Refined Features and WMD

The sentence ranking mechanism was further enhanced by integrating Word Mover's Distance (WMD) and refined TF-IDF scoring, along with additional features such as named entities and sentence position weighting. These enhancements aimed to create a more sophisticated sentence ranking process that considered semantic similarity, term importance, and positional relevance within the text. By incorporating these advanced features, the model improved its ability to select sentences that not only conveyed key information accurately but also maintained coherence and relevance in the generated summaries.

4.5 The Fourth Modifications and Reasons

4.5.1 Expanding Contractions

A significant enhancement introduced in this phase was the integration of a contractions dictionary and a function to expand contractions within the text. This process involved converting informal contractions (e.g., "can't" to "cannot"), which aimed to improve text formality and standardization. By expanding contractions, the quality of the input text was enhanced, facilitating more accurate text processing and analysis essential for effective summarization.

4.5.2 Enhanced Text Cleaning

The text cleaning function underwent updates to include expanding contractions, removing URLs, HTML tags, and special characters. These refinements ensured that the text was consistently in a standardized format, free from noise that could degrade summarization quality. By improving text cleanliness, the model was better equipped to handle diverse text inputs and facilitate accurate feature extraction and processing steps.

4.5.3 Gradient Boosting Regressor for Sentence Scoring

A notable addition was the introduction of a Gradient Boosting Regressor model for scoring sentences based on their features. Gradient Boosting is a powerful machine learning technique that combines multiple weak models to create a strong predictive model. Integrating this model provided an additional layer of refinement in sentence ranking, leveraging predictive insights to prioritize sentences effectively for summarization. This approach aimed to enhance the model's ability to select sentences that contribute most significantly to the overall summary's coherence and informativeness.

4.5.4 Train-Test Split for Additional Model

The implementation of a train-test split function was introduced to partition data into training and testing sets specifically for the Gradient Boosting Regressor model. This step was crucial for evaluating the model's performance on unseen data, ensuring that it generalized well beyond the training dataset. By validating the model's robustness through proper data splitting, this phase aimed to enhance the reliability and effectiveness of the sentence scoring mechanism within the summarization pipeline.

4.6 Final Model and Evaluation

4.6.1 Modifications and Reasons

4.6.1.1 Dataset Loading and Preprocessing

The final step involves loading the test dataset, which consists of articles and their corresponding summaries. This data is used to evaluate the final performance of the summarization model. Preprocessing the text data ensures consistency and cleanliness, which includes expanding contractions, removing URLs, HTML tags, special characters, and unnecessary whitespace. Ensuring that the input text is in a clean and standardized format is essential for accurate analysis and summarization.

4.6.1.2 Lemmatization and Stopword Removal

The preprocessing function continues to lemmatize words and remove stopwords. Lemmatization reduces words to their base forms, and stopword removal eliminates common, non-informative words. These steps improve the quality of the text by focusing on the most informative and meaningful words, enhancing the effectiveness of subsequent processing steps. Clean and meaningful text representations are crucial for effective summarization.

4.6.1.3 Sentence Vector Computation

The method for computing sentence vectors uses preprocessed text, ensuring that sentence vectors are based on clean and meaningful words. Accurate sentence vectors are crucial for effective similarity calculations, which are essential for sentence ranking and summarization. This step transforms the text into a numerical format that captures the semantic meaning of each sentence, facilitating accurate similarity measurements.

4.6.1.4 Named Entity Extraction

The named entity extraction function ensures accurate identification of named entities. Proper extraction of named entities helps in understanding the core information in the text, which is vital for generating meaningful and contextually accurate summaries. Named entities often represent key information that should be retained in the summary, making this step important for capturing the essence of the text.

4.6.1.5 TF-IDF Scoring

The function to compute TF-IDF scores for sentences helps identify the importance of sentences within the context of the article. This improves the ranking process by assigning higher importance to sentences with high TF-IDF scores. TF-IDF is a well-established technique for highlighting important terms in a document, and applying it to sentence ranking ensures that significant content is prioritized.

4.6.1.6 Dynamic Sentence Ranking with Refined Features and WMD

The sentence ranking function includes Word Mover's Distance (WMD), refined TF-IDF scoring, and additional features such as named entities and sentence position weighting. Incorporating these advanced features allows for a more sophisticated sentence ranking process, giving higher importance to sentences with named entities, high TF-IDF scores, and favorable positions while penalizing sentences with high semantic distance. This multi-faceted approach ensures that the most relevant and informative sentences are selected for the summary

4.6.1.7 Abstractive Summary Generation

The abstractive summary generation function using the BART model continues to generate more natural and coherent summaries. Leveraging BART's powerful pre-trained capabilities, the function transforms the dynamically selected sentences into a fluent summary, enhancing the overall readability and quality. Abstractive summarization goes beyond merely extracting sentences, creating new sentences that convey the original text's meaning in a concise and coherent manner.

4.6.1.8 Gradient Boosting Regressor for Sentence Scoring

A Gradient Boosting Regressor model scores sentences based on their features. Gradient Boosting is a powerful machine learning technique that combines multiple weak models to create a strong predictive model. Incorporating this model for sentence scoring provides an additional layer of refinement in ranking sentences, leveraging the predictive power of gradient boosting. This machine learning approach helps in fine-tuning the selection of sentences by learning from features that are indicative of sentence importance

4.6.1.9 Train-Test Split for Additional Model

The train_test_split function splits data into training and testing sets for the additional model. Properly splitting data into training and testing sets is essential for evaluating the performance of machine learning models, ensuring that the model generalizes well to new, unseen data. This step is crucial for assessing the robustness and generalizability of the sentence scoring model.

4.6.1.10 Example Summarization and Evaluation

The final summarization process is tested on sample articles from the test dataset. The generated summaries are compared with reference summaries using the ROUGE metric. This evaluation step ensures that the summaries capture essential information accurately and minimize redundancy, providing a clear measure of the model's effectiveness. Testing on a diverse set of examples helps in validating the model's performance across different types of content.

4.6.1.11 Batch Evaluation

A comprehensive batch evaluation is performed on multiple entries from the test dataset. Each article is summarized, and the generated summaries are evaluated using ROUGE scores. This assessment provides a broad view of the model's performance across different samples, highlighting its strengths and areas needing improvement. Evaluating a large batch of summaries helps in identifying patterns and potential areas for further refinement.

4.6.1.12 Model Training

After batch processing, the sentences and their corresponding scores are used to train the Gradient Boosting Regressor model. This model leverages the features extracted from the text to provide refined sentence rankings. The training process ensures that the model learns to predict the importance of sentences accurately, enhancing the overall summarization quality. This step integrates the learned insights into the model, improving its decision-making process.

4.6.2 Average ROUGE Scores Calculation

Finally, the average ROUGE scores are calculated across the batch of summaries. These scores offer a quantitative measure of the model's overall performance, indicating how well it captures relevant information and maintains precision. The insights from these scores guide further refinements and optimizations of the summarization algorithm, aiming for continuous improvement in summarization quality. Analyzing average scores helps in understanding the general performance trends and guiding future enhancements.

Conclusion

This section has presented a detailed exploration of our model proposition, emphasizing the iterative enhancements made to improve text summarization quality. Through the integration of advanced NLP techniques and machine learning methods, we have developed a robust model capable of generating coherent and relevant summaries. Key improvements include the use of the BART model for abstractive summarization, refined sentence ranking with additional features, and the training of a Gradient Boosting Regressor to enhance sentence importance predictions. Comprehensive evaluations using the ROUGE metric across various datasets demonstrate the model's effectiveness and highlight areas for further refinement. The final model, with its sophisticated preprocessing, feature extraction, and summarization techniques, represents a significant advancement in text summarization, setting a strong foundation for future enhancements and applications.

Chapter 5

Implementation and results

Introduction

In this section, we delve into the implementation and results of our text summarization model, focusing on the iterative changes and improvements made throughout the development process. We document the key modifications introduced to enhance the model's performance, including the integration of advanced NLP techniques such as the BART model for abstractive summarization, refined sentence ranking mechanisms, and new evaluation methods. Each subsection provides a detailed overview of the specific enhancements and their impact on the quality of the generated summaries. This comprehensive examination demonstrates the effectiveness of our approach, highlighting significant progress in achieving superior summarization outcomes through continuous refinement and innovation.

5.1 The changes made in the second model compared to the first model

1. Imports and Setup:
- Added imports for additional NLTK modules: `stopwords`, `WordNetLemmatizer`, `ne_chunk`, `pos_tag`.
- Added NLTK downloads: `punkt`, `averaged_perceptron_tagger`, `maxent_ne_chunker`, `words`, `wordnet`, `stopwords`.
	`import nltk`
	`from nltk.corpus import stopwords`
	`from nltk.stem import WordNetLemmatizer`
	`from nltk import ne_chunk, pos_tag`
	
	`nltk.download('punkt')`
	`nltk.download('averaged_perceptron_tagger')`
	`nltk.download('maxent_ne_chunker')`
	`nltk.download('words')`
	`nltk.download('wordnet')`
	`nltk.download('stopwords')`

2. Preprocessing:
- Introduced a new function `preprocess_text` that:
- Lemmatizes words.
- Removes stopwords.
- Filters non-alphabetic words.
- In the `rank_sentences_with_embeddings` function, `sentence_vector` is now called with `preprocess_text(sentence)` instead of just `sentence`.
	`# Lemmatize and remove stopwords`
	`def preprocess_text(text):`
	`lemmatizer = WordNetLemmatizer()`
	`words = word_tokenize(text.lower())`
	`words = [lemmatizer.lemmatize(word) for word in words if word not in stopwords.words('english') and word.isalpha()]`
	`return ' '.join(words)`

3. Named Entity Extraction:
- Added a new function `extract_named_entities` to extract named entities from the text using `ne_chunk` and `pos_tag`.
	`# Extract named entities`
	`def extract_named_entities(text):`
	`entities = ne_chunk(pos_tag(word_tokenize(text)))`
	`return ' '.join([chunk[0] for chunk in entities if hasattr(chunk, 'label')])`

These changes enhance the preprocessing steps by incorporating lemmatization, stopword removal, and named entity extraction. The updated preprocessing ensures that the input to the `sentence_vector` function is more refined, potentially improving the quality of the sentence embeddings and the resulting summaries.

Summary of the changes in the results before and after the modification:

Adaptive Evaluation Result (One Example):

	Before Modification:			After Modification:		
	F-score	Precision	Recall	F-score	Precision	Recall
ROUGE-1	**0.5360**	0.40625	0.7878	**0.5360**	0.4062	0.7878
ROUGE-2	0.3518	0.2533	0.5757	0.3518	0.2533	0.5757
ROUGE-L	0.5154	0.3906	0.7575	0.5154	0.3906	0.7575

Average ROUGE Scores (100 entries):

	Before Modification:			After Modification:		
	F-score	Precision	Recall	F-score	Precision	Recall
ROUGE-1	**0.2538**	0.1756	0.5085	**0.2646**	0.1847	0.5209
ROUGE-2	0.0827	0.0547	0.1940	0.0946	0.0632	0.2154
ROUGE-L	0.2378	0.1646	0.4763	0.2508	0.1751	0.4936

5.2 The changes made in the third model compared to the second model

1. Imports and Setup: - Added imports for `BartTokenizer` and `TFBartForConditionalGeneration` from the `transformers` library.	`from transformers import BartTokenizer, TFBartForConditionalGeneration`
2. Load BART Model and Tokenizer:	`bart_tokenizer = BartTokenizer.from_pretrained('facebook/bart-large-cnn')` `bart_model = TFBartForConditionalGeneration.from_pretrained('facebook/bart-large-cnn')`

3. Named Entity Extraction:
- Modified the `extract_named_entities` function to extract and join named entities correctly
	`def extract_named_entities(text):`
	`  entities = ne_chunk(pos_tag(word_tokenize(text)))`
	`  return ' '.join([' '.join(c[0] for c in chunk) if hasattr(chunk, 'label') else '' for chunk in entities])`

4. Ranking Sentences with Additional Features:
- Modified the `rank_sentences_with_features` function to include additional features for sentence ranking:
- Boost score for sentences with named entities.
- Simplified TF-IDF scoring.
- Weight for sentence position.
	`def rank_sentences_with_features(article, model, base_sentences=3, factor=0.1):`
	`  sentences = sent_tokenize(article)`
	`  num_sentences = base_sentences + int(len(sentences) * factor)`
	`  sentence_vectors = [sentence_vector(preprocess_text(sentence), model) for sentence in sentences]`
	`  similarity_matrix = cosine_similarity(sentence_vectors)`
	
	`  sentence_scores = similarity_matrix.sum(axis=1)`
	
	`  for i, sentence in enumerate(sentences):`
	`  entities = extract_named_entities(sentence)`
	`  if entities:`
	`  sentence_scores[i] *= 1.5  # Boost score for sentences with entities`
	
	`  tfidf_score = sum(sentence_vector(preprocess_text(sentence), model))  # Simplified TF-IDF scoring`
	`  sentence_scores[i] += tfidf_score`
	
	`  position_weight = (len(sentences) - i) / len(sentences)  # Weight for sentence position`
	`  sentence_scores[i] += position_weight`
	
	`  ranked_sentence_indices = sentence_scores.argsort()[-num_sentences:][::-1]`
	`  ranked_sentences = [sentences[i] for i in ranked_sentence_indices]`
	`  return " ".join(ranked_sentences[:num_sentences])`

5. Generating Abstractive Summary:
- Added the `generate_abstractive_summary` function to generate a summary using the BART model
 	
.
	`def generate_abstractive_summary(text):`
	`  inputs = bart_tokenizer("summarize: " + text, return_tensors="tf", max_length=512, truncation=True)`
	`  outputs = bart_model.generate(inputs["input_ids"], max_length=150, min_length=40, length_penalty=2.0, num_beams=4, early_stopping=True)`
	`  return bart_tokenizer.decode(outputs[0], skip_special_tokens=True)`

6. Example of Summarizing the First Article:
- Modified the summarization example to include the abstractive summary generation
	`example_article = clean_text(data.iloc[0]['article'])`
	`example_summary_with_features = rank_sentences_with_features(example_article, word_vectors)`
	`final_summary = generate_abstractive_summary(example_summary_with_features)`
	`print("Generated Final Summary:", final_summary)`

7. Batch Evaluation on the First 100 Entries:
- Modified the `batch_evaluate` function to include the abstractive summary generation
	`def batch_evaluate(data, model, base_sentences=3, factor=0.1, n=100):`
	`  rouge_scores = []`
	`  for i in range(n):`
	`  article = clean_text(data.iloc[i]['article'])`
	`  reference = clean_text(data.iloc[i]['highlights'])`
	`  generated_summary = rank_sentences_with_features(article, model, base_sentences, factor)`
	`  final_summary = generate_abstractive_summary(generated_summary)`
	`  scores = evaluate_summary(final_summary, reference)`
	`  rouge_scores.append(scores)`
	`  print(f"Processed {i+1}/{n}")`
	`  return rouge_scores`

These changes introduce the BART model for abstractive summarization, add more sophisticated features for sentence ranking, and modify the evaluation process accordingly.

Summary of the changes in the results before and after the modification:

Adaptive Evaluation Result (One Example)

	Before Modification:			After Modification:		
	F-score	Precision	Recall	F-score	Precision	Recall
ROUGE-1	0.5360	0.4062	0.7878	0.4810	0.4130	0.5757
ROUGE-2	0.3518	0.2533	0.5757	0.3132	0.26	0.3939
ROUGE-L	0.5154	0.3906	0.7575	0.4556	0.3913	0.5454

Average ROUGE Scores (100 entries)

	Before Modification:			After Modification:		
	F-score	Precision	Recall	F-score	Precision	Recall
ROUGE-1	0.2646	0.1847	0.5209	0.3129	0.3241	0.3217
ROUGE-2	0.0946	0.0632	0.2154	0.1305	0.1355	0.1344
ROUGE-L	0.2508	0.1751	0.4936	0.2947	0.3057	0.3026

5.3 The changes made in the fourth model compared to the third model

1. Imports and Setup: - Added import for `TfidfVectorizer` from `sklearn.feature_extraction.text`. - Added import for `ot`.	`from sklearn.feature_extraction.text import TfidfVectorizer` `import ot`

2. Compute Word Mover's Distance (WMD):
- Introduced a new function `compute_wmd` to calculate the Word Mover's Distance between two sentences using the Word2Vec model
	`def compute_wmd(sentence1, sentence2, model):`
	`  sentence1 = preprocess_text(sentence1)`
	`  sentence2 = preprocess_text(sentence2)`
	`  return model.wmdistance(sentence1, sentence2)`

3. Compute TF-IDF Scores:
- Introduced a new function `compute_tfidf_scores` to compute TF-IDF scores for the sentences in the article
	`def compute_wmd(sentence1, sentence2, model):`
	`  sentence1 = preprocess_text(sentence1)`
	`  sentence2 = preprocess_text(sentence2)`
	`  return model.wmdistance(sentence1, sentence2)`

4. Ranking Sentences with Refined Features and WMD:
- Modified the `rank_sentences_with_features` function to include the following:
- Calculation of TF-IDF scores for sentences.
- Penalization of sentence scores based on high Word Mover's Distance (WMD) between consecutive sentences.
	`def rank_sentences_with_features(article, model, base_sentences=3, factor=0.1):`
	`  sentences = sent_tokenize(article)`
	`  num_sentences = base_sentences + int(len(sentences) * factor)`
	`  sentence_vectors = [sentence_vector(preprocess_text(sentence), model) for sentence in sentences]`
	`  similarity_matrix = cosine_similarity(sentence_vectors)`
	
	`  sentence_scores = similarity_matrix.sum(axis=1)`
	`  tfidf_scores = compute_tfidf_scores(sentences)`
	
	`  for i, sentence in enumerate(sentences):`
	`  entities = extract_named_entities(sentence)`
	`  if entities:`
	`  sentence_scores[i] *= 1.5  # Boost score for sentences with entities`
	`  sentence_scores[i] += tfidf_scores[i] # Use refined TF-IDF scoring`
	
	`  position_weight = (len(sentences) - i) / len(sentences)  # Weight for sentence position`
	`  sentence_scores[i] += position_weight`
	
	`  if i < len(sentences) - 1:`
	`  wmd_score = compute_wmd(sentence, sentences[i + 1], model)`
	`  sentence_scores[i] -= wmd_score # Penalize for high WMD (more distance)`
	
	`  ranked_sentence_indices = sentence_scores.argsort()[-num_sentences:][::-1]`
	`  ranked_sentences = [sentences[i] for i in ranked_sentence_indices]`
	`  return "\n".join(ranked_sentences[:num_sentences])`

.

These changes refine the sentence ranking process by incorporating TF-IDF scores and Word Mover's Distance, potentially improving the quality and coherence of the generated summaries.

Summary of the changes in the results before and after the modification:

Adaptive Evaluation Result (One Example)

	Before Modification:			After Modification:		
	F-score	**Precision**	**Recall**	**F-score**	**Precision**	**Recall**
ROUGE-1	**0.4810**	0.4130	0.5757	**0.4871**	0.4222	0.5757
ROUGE-2	0.3132	0.26	0.3939	0.3170	0.2653	0.3939
ROUGE-L	0.4556	0.3913	0.5454	0.4615	0.4	0.5454

Average ROUGE Scores (100 entries)

	Before Modification:			After Modification:		
	F-score	**Precision**	**Recall**	**F-score**	**Precision**	**Recall**
ROUGE-1	**0.3129**	0.3241	0.3217	**0.3274**	0.3436	0.3312
ROUGE-2	0.1305	0.1355	0.1344	0.1381	0.1456	0.1404
ROUGE-L	0.2947	0.3057	0.3026	0.3120	0.3272	0.3156

5.4 The changes made in the fifth model compared to the fourth model

1. Imports and Setup: - Added imports for `train_test_split` from `sklearn.model_selection` and `GradientBoostingRegressor` from `sklearn.ensemble`.	`from sklearn.model_selection import train_test_split` `from sklearn.ensemble import GradientBoostingRegressor`

2. Contractions Dictionary:
- Introduced a dictionary `contractions` for expanding contractions in the text.
	`contractions = {`
	`"ain't": "am not", "aren't": "are not", "can't": "cannot", "can't've": "cannot have", "'cause": "because",`
	`....`
	`"who'll": "who will", "who's": "who is", "won't": "will not", "wouldn't": "would not", "you'd": "you would",`
	`"you'll": "you will", "you're": "you are"`
	`}`

3. Expand Contractions Function:
- Introduced a new function `expand_contractions` to expand contractions in the text using the contractions dictionary
	`def expand_contractions(text, contractions_dict):`
	`pattern = re.compile(r'\b(' + '|'.join(contractions_dict.keys()) + r')\b')`
	`return pattern.sub(lambda x: contractions_dict[x.group()], text)`

4. Text Cleaning Function: - Modified the `clean_text` function to include contraction expansion and other text cleaning steps	`def clean_text(text):` `text = expand_contractions(text, contractions)` `text = re.sub(r'https?:\/\/.*[\r\n]*', '', text, flags=re.MULTILINE)` `text = re.sub(r'\<a href', ' ', text)` `text = re.sub(r'&', '', text)` `text = re.sub(r'[_"\-;%()|+&=*%.,!?:#$@\[\]/]', ' ', text)` `text = re.sub(r' ', ' ', text)` `text = re.sub(r'\'', ' ', text)` `return text`

5. Generate Abstractive Summary:
- Modified the `generate_abstractive_summary` function to use different parameters for the BART model
	`def generate_abstractive_summary(text):`
	`inputs = bart_tokenizer("summarize: " + text, return_tensors="tf", max_length=512, truncation=True)`
	`outputs = bart_model.generate(inputs["input_ids"], max_length=150, min_length=40, length_penalty=2.0, num_beams=6, early_stopping=True)`
	`return bart_tokenizer.decode(outputs[0], skip_special_tokens=True)`

6. Vectorize Text Function:
- Introduced a new function `vectorize_text` to vectorize text using TF-IDF
	`def vectorize_text(texts):`
	`vectorizer = TfidfVectorizer(max_features=5000)`
	`return vectorizer.fit_transform(texts).toarray(), vectorizer`

7. Train Additional Model:
- Introduced a new function `train_additional_model` to train a Gradient Boosting Regressor for sentence scoring
 	
.
	`def train_additional_model(sentences, scores):`
	`X_vectors, vectorizer = vectorize_text(sentences)`
	`X_train, X_test, y_train, y_test = train_test_split(X_vectors, scores, test_size=0.2, random_state=42)`
	`model = GradientBoostingRegressor(n_estimators=100, max_depth=3, learning_rate=0.1)`
	`model.fit(X_train, y_train)`
	`print(f"Model training completed. Score: {model.score(X_test, y_test)}")`
	`return model, vectorizer`

8. Batch Evaluation:
- Modified the `batch_evaluate` function to include the training of the additional model
	`def batch_evaluate(data, model, base_sentences=3, factor=0.1, n=100):`
	`  rouge_scores = []`
	`  sentences = []`
	`  scores = []`
	
	`  for i in range(n):`
	`  article = clean_text(data.iloc[i]['article'])`
	`  reference = clean_text(data.iloc[i]['highlights'])`
	`  generated_summary = rank_sentences_with_features(article, model, base_sentences, factor)`
	`  final_summary = generate_abstractive_summary(generated_summary)`
	`  evaluation_scores = evaluate_summary(final_summary, reference)`
	`  rouge_scores.append(evaluation_scores)`
	`  sentences.append(generated_summary)`
	
	`scores.append(evaluation_scores[0]['rouge-1']['f'])`
	`  print(f"Processed {i+1}/{n}")`
	
	`  additional_model, vectorizer = train_additional_model(sentences, scores)`
	`  return rouge_scores, additional_model, vectorizer`

9. Perform Batch Evaluation: - Modified the call to `batch_evaluate` to handle the additional model and vectorizer	`batch_results, additional_model, vectorizer = batch_evaluate(data, word_vectors, base_sentences=3, factor=0.1, n=100)`

These changes introduce additional preprocessing steps, an abstractive summary generation method, and a new model (Gradient Boosting Regressor) for refining sentence scoring.

Adaptive Evaluation Result (One Example)

	Before Modification:			After Modification:		
	F-score	**Precision**	**Recall**	**F-score**	**Precision**	**Recall**
ROUGE-1	**0.4872**	0.4222	0.5758	**0.4565**	0.3559	0.6364
ROUGE-2	0.3171	0.2653	0.3939	0.3137	0.2319	0.4848
ROUGE-L	0.4615	0.4000	0.5455	0.3913	0.3051	0.5455

Average ROUGE Scores (100 entries)

	Before Modification:			After Modification:		
	F-score	**Precision**	**Recall**	**F-score**	**Precision**	**Recall**
ROUGE-1	**0.3275**	0.3436	0.3312	**0.4153**	0.3925	0.4675
ROUGE-2	0.1381	0.1456	0.1405	0.2105	0.1984	0.2404
ROUGE-L	0.3121	0.3273	0.3157	0.3627	0.3430	0.4074

5.5 Final model Explanation

The final model represents the culmination of our efforts to develop a robust text summarization system. By leveraging advanced techniques in NLP and machine learning, this model excels in processing and summarizing articles effectively. Here are the key components and modifications, along with the advantages they bring:

5.5.1 Model Components and Techniques

5.5.1.1 Dataset and Pre-trained Models

Dataset: The model uses the test.csv dataset, which provides an unbiased evaluation of performance on unseen data.
Word2Vec: Pre-trained Word2Vec model for generating word embeddings, capturing rich semantic information.
BART: BART model for abstractive summarization, producing coherent and fluent summaries.

5.5.1.2 Text Cleaning and Preprocessing

Expanding Contractions: Converts informal contractions to their full forms, enhancing text quality.
Enhanced Cleaning: Removes URLs, HTML tags, and special characters to standardize the text.
Lemmatization and Stopword Removal: Reduces words to their base forms and removes non-informative words.

5.5.1.3 Advanced Feature Extraction

Sentence Vectors: Uses preprocessed text to compute meaningful sentence vectors.
Named Entity Extraction: Identifies and boosts the importance of named entities.
TF-IDF Scoring: Assesses sentence importance within the article context.
Word Mover's Distance (WMD): Measures semantic distance between sentences to refine ranking.

5.5.1.4 Sentence Ranking and Summary Generation

Dynamic Sentence Ranking: Integrates multiple features, including named entities, TF-IDF scores, sentence position, and WMD, to rank sentences effectively.
Abstractive Summary Generation: Utilizes BART for generating natural and coherent summaries.

5.5.1.5 Machine Learning Integration

Gradient Boosting Regressor: Scores sentences based on their features, adding an additional layer of refinement.

5.5.2 Comparison Table

To evaluate the performance of our model against other well-known models, we provide a comparative analysis using ROUGE scores. This comparison highlights the effectiveness of our model in producing high-quality summaries. Table 5.1 presents the ROUGE scores for our model compared to several famous models.

Table 5.1: Comparison Table of our model with famous models

Model	**ROUGE-1 (F1)**	**ROUGE-2 (F1)**	**ROUGE-L (F1)**
Our Model	**0.4153**	**0.2105**	**0.3627**
RNES w/o coherence	0.4125	0.1887	0.3775
SWAP-NET	0.4160	0.1830	0.3770
HSASS	0.4230	0.1780	0.3760
GAN	0.3992	0.1765	0.3671
KIGN + Prediction-guide	0.3895	0.1712	0.3568

5.5.3 Advantages of the Final Model

The final model distinguishes itself by evaluating on a sample of 100 articles from the test.csv dataset, unlike previous models that evaluated based on a single example. This comprehensive evaluation approach provides a more reliable and accurate measure of the model's performance, capturing a broader range of scenarios and significantly reducing bias. By assessing multiple articles, the model demonstrates robustness and versatility across different types of content, ensuring that the summaries generated are consistently high-quality.

The model's thorough text cleaning and preprocessing steps ensure that the input text is in an optimal format for analysis. This robust text processing enhances the accuracy of feature extraction and subsequent processing steps, leading to improved performance. By removing noise and irrelevant elements, the model can focus on the essential content, resulting in more precise and meaningful summaries.

The integration of named entities, TF-IDF scores, sentence position weighting, and Word Mover's Distance (WMD) allows the model to capture the most important aspects of the text. This advanced

feature integration enables the model to produce more relevant and informative summaries by considering multiple dimensions of sentence importance. Each feature adds a layer of understanding, ensuring that the final summaries reflect the core ideas and key points of the original text.
Using the BART model for abstractive summarization ensures that the generated summaries are not only relevant but also coherent and fluent. This state-of-the-art approach results in high-quality summaries that are easy to read and understand, significantly enhancing user satisfaction. The BART model's ability to generate natural language text makes the summaries more engaging and closer to human-written summaries.
The inclusion of a Gradient Boosting Regressor adds an additional layer of refinement to sentence scoring, leveraging its predictive power. This machine learning enhancement further improves the accuracy of sentence ranking and the overall quality of the summaries. By training on features extracted from the text, the regressor can better predict which sentences are most important, leading to more effective summarization.

5.5.4 Areas for Improvement of Our Model

Despite improvements, there are still instances of redundancy in the generated summaries, indicating a need for better handling of repetitive information. Redundant sentences can detract from the quality and conciseness of the summaries, making them less effective.
The model's reliance on various computationally intensive steps, such as WMD and TF-IDF calculations, may limit its scalability and real-time applicability. These processes can be resource-heavy, posing challenges for deployment in environments where speed and efficiency are crucial.
The negative training score of the Gradient Boosting Regressor model suggests issues with its current implementation or feature set, requiring further investigation and optimization. Improving the model's training process is essential for achieving better performance and more accurate sentence ranking.
While our model demonstrates high recall, precision remains relatively lower, indicating that the summaries often capture a lot of information but may include less relevant details. Balancing precision and recall is vital for generating summaries that are both comprehensive and focused.

5.5.5 Future Challenges and Ideas

To enhance our model, we need to address several key areas. First, implementing techniques to eliminate redundant information during the summarization process will streamline our summaries and improve readability. Advanced NLP methods like coreference resolution can help merge similar sentences or concepts, reducing repetition.
Improving the precision of our summaries involves refining the feature extraction process. By experimenting with different weighting schemes for features like named entities, sentence positions,

and TF-IDF scores, we can better distinguish important information from less relevant details, ensuring more targeted and relevant summaries.

Optimizing computational steps is crucial for scalability and efficiency. Exploring approximate methods or more efficient algorithms for WMD and TF-IDF calculations, and considering lightweight models for real-time applications, will make our model more versatile and practical.

Balancing the contributions of extractive and abstractive methods is essential for refining our hybrid approach. Using reinforcement learning to fine-tune this integration can leverage the strengths of both methods, producing superior summaries that combine accuracy and fluidity.

Enhancing the training process for the Gradient Boosting Regressor involves experimenting with different features, hyperparameters, and regularization techniques. Collecting more training data or using data augmentation can improve model generalization and sentence ranking accuracy.

Exploring new architectures like GPT-3, T5, or other transformer-based models designed for summarization tasks can provide insights and opportunities for further enhancement. These state-of-the-art models may offer better performance and new capabilities.

Developing methods to tailor summaries to specific user preferences can enhance personalization, making summaries more relevant and useful for different audiences. Extensive evaluations using diverse datasets and benchmarks, along with additional metrics like BLEU, METEOR, or human evaluation, will ensure robustness and generalizability. Comprehensive evaluation and benchmarking will help identify areas for improvement and validate our model's effectiveness.

By addressing these challenges and exploring new ideas, we can further improve our text summarization model, generating more accurate, coherent, and efficient summaries for a wide range of applications. This continuous development will keep our model at the forefront of summarization technology, adapting to evolving demands.

Conclusion

In this section, we have presented a detailed account of the iterative enhancements made to our text summarization model, emphasizing the progressive improvements in performance and summary quality. The integration of the BART model for abstractive summarization, alongside refined sentence ranking mechanisms and sophisticated evaluation methods, has substantially bolstered the model's capability to generate coherent and relevant summaries. Each version of the model introduced targeted modifications that addressed specific shortcomings, resulting in measurable gains in ROUGE scores and overall summary effectiveness. This iterative development process underscores the importance of continuous refinement and the application of advanced NLP techniques in achieving state-of-the-art summarization performance. The final model, with its robust preprocessing, enhanced feature extraction, and innovative summarization approach, demonstrates significant progress and sets a solid foundation for future advancements in text summarization.

5.6 References

5.6.1 Word2Vec Model

- **Model** Google News pre-trained Word2Vec
- **Reference** Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781.
- **Details**
  - **Training Data** Trained on part of the Google News dataset (about 100 billion words).
  - **Vocabulary Size** 3 million words and phrases.
  - **Embedding Dimension** : 300 dimensions.
  - **File Format** Binary format
  - **Source** Available for download from Google Code Archive

5.6.2 BART (Bidirectional and Auto-Regressive Transformers) Model

- **Model** facebook/bart-large-cnn
- **Reference** Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., ... & Zettlemoyer, L. (2019). BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. arXiv preprint arXiv:1910.13461.
- **Details**
  - **Architecture** : Sequence-to-sequence model with a bidirectional encoder (like BERT) and a left-to-right autoregressive decoder (like GPT).
  - **Training Data** Trained on a combination of the CC-News, Books, Wikipedia, and stories datasets
  - **Tokenizer** Byte-Pair Encoding (BPE).
  - **Pre-trained Variants** bart-large, bart-large-cnn (fine-tuned for summarization), etc
  - **Hugging Face Model Hub** facebook/bart-large-cnn.

5.6.3 ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

- **Tool** py-rouge (Python package for ROUGE evaluation)
- **Reference** Lin, C. Y. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. In Text Summarization Branches Out (pp. 74-81).
- **Details**
  - **Metrics** Measures the quality of summaries by comparing the overlap of n-grams, word sequences, and word pairs between the machine-generated summary and the reference summary
  - **Variants** ROUGE-N (unigram, bigram), ROUGE-L (Longest Common Subsequence), ROUGE-W (Weighted Longest Common Subsequence), ROUGE-S (Skip-bigram), etc.
  - **Python Package**py-rouge can be installed via pip.

5.6.4 NLTK (Natural Language Toolkit)

- **Tool** Various pre-trained models and corpora from NLTK
- **Reference** Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. O'Reilly Media, Inc.
- **Details**
  - **Components Used** Tokenizers (punkt), POS tagger (averaged_perceptron_tagger), Named Entity Chunker (maxent_ne_chunker), WordNet Lemmatizer (wordnet), Stopwords (stopwords).

- **Functionality** Provides easy-to-use interfaces to over 50 corpora and lexical resources, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and more.
- **Website** NLTK

5.6.5 Gradient Boosting Regressor

- **Library** scikit-learn
- **Reference** Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.
- **Details**
  - **Model** Gradient Boosting Regressor
  - **Functionality** Ensemble of decision trees that combines the predictions of several base estimators to improve robustness over a single estimator.
  - **Parameters** Number of estimators, maximum depth of the individual regression estimators, learning rate, etc.
  - **Documentation** Scikit-learn Gradient Boosting.

Chapter 6

General Conclusion

In this thesis, we have explored various text summarization techniques, including extractive, abstractive, and hybrid methods. Our proposed hybrid model, which combines the strengths of both extractive and abstractive approaches, has shown promising results. By leveraging advanced NLP techniques such as transformers and LSTM with attention mechanisms, we achieved notable improvements in summary coherence and informativeness.

6.1 Implications of Research

This research has several important implications for both academic research and practical applications in various fields.

First, our hybrid approach advances the field of text summarization. It combines the precision of extractive methods with the flexibility of abstractive methods, creating a more effective summarization technique. This framework can serve as a foundation for future improvements and innovations.

Second, we demonstrated the potential of combining advanced NLP techniques like transformers and LSTM with attention mechanisms. This approach addresses the limitations of using these techniques individually, providing a more comprehensive solution for text summarization tasks. Future researchers can build on our work to further enhance NLP models.

Third, our research has practical applications across various industries. Media companies can use our model to generate concise news summaries, improving content delivery. Legal and healthcare professionals can benefit from better document summarization, enabling faster decision-making and improved information management.

Overall, this thesis makes a significant contribution to text summarization. Our hybrid model offers a new direction for future research and development, emphasizing the importance of combining different techniques to achieve better performance and adaptability in NLP tasks.

6.2 Limitations and General Challenges

Figure 6.1: General Challenges Of Text Summarization

Text summarization is an important part of natural language processing, but it has a lot of challenges that make it less useful and efficient. These challenges happen because language is hard to understand, the diversity of text types, and the varying requirements of different applications. Challenges come in many forms, and each one affects different parts of the summary process. We will talk more about these problems in the parts that follow.

Figure 6.1 illustrates the general challenges of text summarization, highlighting various aspects such as input and output formats, user-specific requirements, and the complexity of the summarization methods.

Challenges Related to Muti-Document Summarization: Multi-document summarization is a challenging undertaking that involves several complexities and difficulties like managing redundancy, handling contradictions, and synthesizing diverse information from various sources. Redundancy happens when many documents contain overlapping content, necessitating the system to eliminate repetitive information. Contradictions occur when different sources provide conflicting information, requiring the use of techniques to resolve variations and create a coherent summary.Furthermore, the large amount of information from many sources offers a difficulty in efficiently combining various aspects of a subject into a cohesive summary.

Challenges Related to User-Specific Summarization:The main challenge in this situation is to compress information from many textual and semi-structured sources, such as databases and web pages, in a manner that is appropriate for a particular user in terms of language, format, size, and timeliness [90]. In order to handle an overwhelming amount of data in many formats and languages, it is necessary to provide more research resources to the creation of summaries that cover several documents, languages, and media types.

Challenges Related to Applications of Text Summarization: The majority of current systems are focused on specific applications such as online reviews, text news, and text pages [91]. It is crucial to prioritise our attention on the most demanding tasks, such as summarising lengthy texts, novels, and books.

Challenges Related to Input and Output Formats: The majority of ATS systems handle textual input and output. The task at hand necessitates the introduction of new summarization methods that can handle input in various formats such as meetings, videos, sounds, etc., and provide output in formats other than text. As an example, the input could consist of textual data, whereas the output could be given in the form of tables, statistics, graphics, visual rating systems, and so on. ATS systems with the capability to display summaries visually will assist users in obtaining the necessary information more efficiently.

Challenges Related to Length of Input Documents: The majority of ATS systems favour relatively short text documents. As an example, the news unit is significantly shorter in length compared to a chapter in a novel (about 732 words versus 5,246 words). Summarising long papers can be hard on computers, and you may need to use complex methods to get the important information out without losing the context. While summarising very short documents is easier and achieve higher accuracy and more efficiency compared to long texts.

Challenges Related to Supported Languages: The majority of ATS systems favour English language content. The current ATS systems need to be enhanced in order to increase the quality for numerous other languages. Developing models that can accurately summarize texts in multiple languages requires extensive multilingual datasets and sophisticated algorithms capable of understanding and processing various language structures. Additionally, ensuring the quality and consistency of summaries across different languages is challenging.

Challenges Related to Text Summarization Approaches: The majority of research is centered around the extractive approach. However, this approach often struggles with coherence and maintaining a natural flow. There is a need to prioritise research efforts on proposing and enhancing summarization systems that utilise abstractive and hybrid techniques.

Challenges Related to Statistical and Linguistic Features:It is necessary to identify creative linguistic and statistical characteristics of sentences and words that can effectively extract the essential phrases with semantic relevance from the source document(s). Additionally, determining the appropriate weights for each individual feature is crucial as the overall quality of the final summary is contingent upon this factor.

Challenges Related to Using Deep Learning for Text Summarization:During the summary generation step, the recurrent neural network (RNN) of the seq2seq system requires a substantial amount of organised training data. Deep learning models, while powerful, can be prone to generating inaccurate or nonsensical summaries, especially when dealing with complex or ambiguous text. In addition, these models don't always show how they make decisions, which makes it hard to understand and explain them. Developing an ATS system with limited training data by integrating traditional NLP techniques like syntactic analysis, grammatical analysis, and semantic analysis is a highly significant research area.

Challenges Related to Stop Criteria of the Summarization Process: Humans compress information in a repetitive procedure when summarising documents. Upon generating the initial summary, it is necessary for either the user or the system to make a judgement regarding whether to stop or continue with the summarization process. One usual way is to set a retention rate that is used to make a choice. The rate of retention isn't the same for every text. It should be different depending on the content and what it's about. It is very important to come up with a better way to stop summarising.

Challenges Related to the Quality of Generated Summary: It is necessary to attain an optimal ratio between legibility, compression ratio, and quality of summary. Current ATS systems face challenges in meeting the demand for a better compression ratio when summarising lengthy materials such as novels and books [91]. It is necessary to make the summary easier to read by editing the initial version to get rid of the semantic misunderstanding that comes from using words that aren't clear or that mean the same thing.

Challenges Related to Evaluation of the Generated Summary: Evaluating summaries, whether done automatically or manually, is a challenging job. Even though human evaluation is often seen as the best, it is subjective and can change a lot from one evaluator to the next, making it hard to be consistent. Also, Automatic metrics like ROUGE and BLEU, which depend on n-gram overlap, don't always get the semantic meaning and coherence of the summary right, which is why human and machine ratings don't always agree. It is necessary to suggest novel methodologies and solutions for the automated evaluation of computer-generated summaries.

6.3 Practical Applications

Our research can be applied in many real-world scenarios to make information processing more efficient. For example, media companies can use our summarization model to create quick, concise news summaries, saving time and improving content delivery. This can help readers get the most important news quickly.

In the legal field, professionals can use our model to summarize lengthy legal documents. This will help lawyers and judges find key information faster, making their work more efficient. Similarly, in healthcare, doctors can use summarized medical records to quickly review patient histories, leading to better and faster decision-making.

Educational tools can also benefit from our model by generating summaries of academic papers. This can help students and researchers save time by quickly understanding the main points of lengthy documents. Overall, our model has the potential to improve information management in various sectors.

Bibliography

[1] Vince Hartman. Extractive vs abstractive summarization in healthcare. 2022.

[2] James Yi and Simon Zamarin. Democratize documentation summarization with hugging face on amazon sagemaker. 2022.

[3] Mohsen Nabil. The architecture of a basic rnn. 2023.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention is All You Need*. 2017.

[5] Dion Hoe-Lian Goh Shiyan Ou, Christopher S.G. Khoo. *Handbook of Research on Digital Libraries: Design, Development, and Impact*. 2013.

[6] H. P Luhn. The automatic creation of literature abstracts. *IBM Journal of Research Development*, 2(2):159–165, 1958.

[7] Juan-Manuel TORRES-MORENO. *Automatic Text Summarization*. 2014.

[8] Chin-Yew Lin. *ROUGE: A Package for Automatic Evaluation of Summaries*. 2004.

[9] Nir Gazit. Evaluating model performance with the rouge metric: A comprehensive guide. 2023.

[10] Fabio Chiusano. Two minutes nlp — learn the bleu metric by examples. 2022.

[11] K Lopyrev. Generating news headlines with recurrent neural networks. 2015.

[12] Prasasthy K B. Applications of text summarization. 2022.

[13] Ahmad sabry. A perfect guide to understand encoder decoders in depth with visuals. 2023.

[14] Vishal Patil, Mahalakshmy Krishnamoorthy, Parag Oke, and Prof. M. Kiruthika. A statistical approach for document summarization.

[15] N. Moratanch Chitrakala Gopalan. Concept-based extractive text summarization using graph modelling and weighted iterative ranking. 2018.

[16] Subalalitha C. N. Kalliath Abdul Rasheed Issam**, Shivam Patel**. Topic modeling based extractive text summarization.

[17] McKeown K Nenkova, A. A survey of text summarization techniques. 2012.

[18] Radev D. R. Erkan, G. Lexrank: Graph-based lexical centrality as salience in text summarization. *Artificial Intelligence Research*, 2004.

[19] Majumder P. Mehta, P. Effective aggregation of various summarization techniques. 2018.

[20] Zhao X. Li B. Ge B. Tang D. Wang, S. Integrating extractive and abstractive models for long text summarization. 2017.

[21] Chitrakala S Moratanch, N. A survey on extractive text summarization. 2017.

[22] Nguyen M. L. Chen, L. Sentence selective neural extractive summarization with reinforcement learning. 2019.

[23] Sharma A. Kumar, A. Systematic literature review of fuzzy logic based text summarization. *Iranian Journal of Fuzzy Systems*, 2019.

[24] Mahdavi M. A. Nazari, N. A survey on automatic text summarization. *AI and Data Mining*, 2019.

[25] Rafiq F. M. Saha R. Rafian R. Arif H. Rahman, A. Bengali text summarization using textrank, fuzzy c-means and aggregate scoring methods. 2019.

[26] Jan R. Shah M. Mohd, M. Text document summarization using word embedding. 2020.

[27] Gambhir and V M., Gupta. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, 2017.

[28] Ma J. Wang, Y. A comprehensive method for text summarization based on latent semantic analysis. in g. 2013.

[29] Jaffry S. W. Malik M. K. Nasar, Z. Textual keyword extraction and summarization: State-of-the-art. 2019.

[30] Zhai F. Zhou B. Nallapati, R. Summarunner: A recurrent neural network based sequence model for extractive. 2017.

[31] Gupta V. Gambhir, M. Recent automatic text summarization techniques: a survey. 2017.

[32] Bansal N. Sharma A. Gupta, V. Text summarization for big data: A comprehensive survey. 2019.

[33] Chitrakala S. Moratanch, N. A survey on abstractive text summarization. 2016.

[34] Gupta S. K. Gupta, S. Abstractive summarization: An overview of the state of the art. *Expert Systems with Applications*, 2019.

[35] Hu P. Bei C. Hou, L. Abstractive document summarization via neural model with joint attention. 2017.

[36] R. Pasunuru and M. Bansal. Reinforced neural extractive summarization with coherence modeling. 2018.

[37] S. B. Mishra and P. Li. Extractive summarization with swap-net: Sentences and words from alternating pointer networks. 2018.

[38] M. Yasunaga and D. Radev. Graph-based neural multi-document summarization. 2017.

[39] Y. Liu and M. Lapata. Text summarization with pretrained encoders. 2019.

[40] Y. Sun, S. Wang, Y. Li, and H. Cao. Key information guide network for text summarization. 2018.

[41] R. Nallapati, F. Zhai, and B. Zhou. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. 2017.

[42] Y. C. Chen and M. Bansal. Abstractive summarization of long documents with reinforce-selected sentence rewriting. 2018.

[43] W. T. Hsu, C. K. Lin, M. Y. Lee, K. S. Min, and J. Tang. A unified model for extractive and abstractive summarization using inconsistency loss. 2018.

[44] M. Zhong, P. Liu, Y. Li, J. Li, P. Wang, and Y. Liu. Extractive summarization as text matching. 2020.

[45] H. Xu and G. Durrett. Discourse-aware neural extractive text summarization. 2020.

[46] Y. Liu and M. Lapata. Fine-tune bert for extractive summarization. 2019.

[47] Y. Liu and M. Lapata. Fine-tune bert for extractive summarization. 2019.

[48] S. Gehrmann, Y. Deng, and A. Rush. Bottom-up abstractive summarization. 2018.

[49] X. Zhang and M. Lapata. Hibert: Document level pre-training of hierarchical bidirectional transformers for document summarization. 2019.

[50] Q. Zhou, N. Yang, F. Wei, and M. Zhou. Neural document summarization by jointly learning to score and select sentences. 2018.

[51] X. Zhang and M. Lapata. Latent alignment and variational attention for abstractive summarization. 2018.

[52] Y. Dong, Y. Shen, H. Liu, and M. Lapata. Banditsum: Extractive summarization as a contextual bandit. 2018.

[53] S. Narayan, S. Cohen, and M. Lapata. Ranking sentences for extractive summarization with reinforcement learning. 2018.

[54] Y. Liu, Y. Zhang, Z. Dou, and Z. Yan. Brio: Bringing order to abstractive summarization. 2022.

[55] J. Liu, W. Gao, Z. Shen, and J. Guo. Simcls: A simple framework for contrastive learning of abstractive summarization. 2021.

[56] Z. Dou, Y. Liu, D. Dou, and Y. Zhang. Gsum: A general framework for guided neural abstractive summarization. 2021.

[57] W. Qi, Z. Yan, Y. Liu, and Y. Liu. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. 2020.

[58] X. Zhang, Y. Yang, J. Sun, and D. Chen. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. 2020.

[59] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. 2020.

[60] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. 2020.

[61] L. Dong, N. Yang, W. Wang, F. Wei, and M. Zhou. Unified language model pre-training for natural language understanding and generation. 2019.

[62] H. Liu, Y. Dong, H. Zhang, and Y. Shen. Hierarchical convolutional neural networks for text summarization. 2019.

[63] Y. Liu and M. Lapata. Bert for extractive summarization with hybrid extractive-abstractive summarization. 2019.

[64] X. Zhang and M. Lapata. Hierarchical neural network for extractive summarization. 2019.

[65] S. Narayan, S. Cohen, and M. Lapata. Two-stage fine-tuning for abstractive summarization. 2019.

[66] X. Zhang and M. Lapata. Deep communicating agents for abstractive summarization. 2018.

[67] Y. Gao, Y. Shen, H. Liu, and M. Lapata. Editnet: Learning to write by combining inverse document frequency and mutual information. 2018.

[68] Y. C. Chen and M. Bansal. Reinforcement learning for abstractive document summarization with policy gradient and reranking. 2018.

[69] S. Gehrmann, Y. Deng, and A. Rush. Bottom-up abstractive summarization. 2018.

[70] X. Li, Y. Li, H. Cao, and Y. Shen. Improving neural abstractive document summarization with explicit information selection modeling. 2018.

[71] X. Li, Y. Li, H. Cao, and Y. Shen. Improving neural abstractive document summarization with structural regularization. 2018.

[72] W. T. Hsu, C. K. Lin, M. Y. Lee, K. S. Min, and J. Tang. A unified model for extractive and abstractive summarization using inconsistency loss. 2018.

[73] S. Narayan, S. Cohen, and M. Lapata. Closed-book training to improve summarization encoder memory. 2019.

[74] X. Zhang and M. Lapata. Soft layer-specific multi-task summarization with entailment and question generation. 2019.

[75] Z. Dai, Z. Yang, F. Yang, W. Cohen, and R. Salakhutdinov. Plug and play language models: A simple approach to controlled text generation. 2019.

[76] Y. Zhang, Y. Shen, H. Liu, and M. Lapata. Fast abstractive summarization with reinforce-selected sentence rewriting. 2018.

[77] S. Narayan, S. Cohen, and M. Lapata. Sensible neural networks for context-aware text summarization. 2018.

[78] Z. Yang, S. Zhang, J. Gao, I. Abdelaziz, H. Luo, C. Li, and Z. Lin. Recitation-augmented generation for open-domain question answering. 2020.

[79] H. Wang, Y. Yang, H. Wang, Y. Xu, and S. Lin. Attention-based lstm for aspect-level sentiment classification. 2016.

[80] Z. Guo, H. Zhao, H. Lin, R. Wang, R. Socher, and C. Xiong. Sentence rewriting with denoising autoencoder. 2021.

[81] T. Wang, Y. Liu, F. Liu, Z. Zeng, and J. Zhao. Sentence compression with pre-trained language models. 2020.

[82] K. Song, X. Tan, T. Qin, J. Lu, and T. Liu. Massive exploration of neural machine translation architectures. 2019.

[83] Y. Zhao, Y. Chen, J. Peng, L. Xue, and L. Li. Set to sequence mapping for text summarization. 2018.

[84] J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan. Extended transformer models for summarization. 2019.

[85] K. Song, X. Tan, T. Qin, J. Lu, and T. Liu. Unsupervised neural machine translation. 2019.

[86] Y. Yu, W. Zhang, K. Wang, W. Zuo, D. Xiong, and H. Li. Learning to summarize with human feedback. 2019.

[87] R. Nallapati, B. Zhou, C. Gulcehre, and B. Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond. 2016.

[88] S. Chopra, M. Auli, and A. Rush. Abstractive sentence summarization with attentive recurrent neural networks. 2016.

[89] A. Fan, M. Lewis, and Y. Dauphin. Hierarchical neural story generation. 2018.

[90] Lehal G. S. Gupta, V. A survey of text summarization extractive techniques. *Emerging Technologies in Web Intelligence*, 2010.

[91] Lei L. Li G. Huang H. Zheng C. Chen E. Xu G. Wu, Z. A topic modeling based approach to novel document automatic summarization. 2017.

Acronyms

TS Text summarization

GPU Graphical Processing Units

TF-IDF term frequency-inverse document frequency

LSTM long short-term memory

ROUGE Recall-Oriented Understudy for Gisting Evaluation

METEOR Metric for Evaluation of Translation with Explicit Ordering

NLP Natural Language Processing

NN Neural Networks

RNN Recurrent Neural Network

Seq2Seq Sequence-to-Sequence

WMD Word Mover's Distance

BART Bidirectional and Auto-Regressive Transformers

NLTK Natural Language Toolkit

ATS Automatic Text Summarization

Appendix

Word2Vec Model

- **Model** Google News pre-trained Word2Vec
- **Reference** Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781.
- **Details**
  - **Training Data** Trained on part of the Google News dataset (about 100 billion words).
  - **Vocabulary Size** 3 million words and phrases.
  - **Embedding Dimension** : 300 dimensions.
  - **File Format** Binary format
  - **Source** Available for download from Google Code Archive

BART (Bidirectional and Auto-Regressive Transformers) Model

- **Model** facebook/bart-large-cnn
- **Reference** Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., ... & Zettlemoyer, L. (2019). BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. arXiv preprint arXiv:1910.13461.
- **Details**
  - **Architecture** : Sequence-to-sequence model with a bidirectional encoder (like BERT) and a left-to-right autoregressive decoder (like GPT).
  - **Training Data** Trained on a combination of the CC-News, Books, Wikipedia, and stories datasets
  - **Tokenizer** Byte-Pair Encoding (BPE).
  - **Pre-trained Variants** bart-large, bart-large-cnn (fine-tuned for summarization), etc
  - **Hugging Face Model Hub** facebook/bart-large-cnn.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

- **Tool** py-rouge (Python package for ROUGE evaluation)
- **Reference** Lin, C. Y. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. In Text Summarization Branches Out (pp. 74-81).
- **Details**
  - **Metrics** Measures the quality of summaries by comparing the overlap of n-grams, word sequences, and word pairs between the machine-generated summary and the reference summary

- **Variants** ROUGE-N (unigram, bigram), ROUGE-L (Longest Common Subsequence), ROUGE-W (Weighted Longest Common Subsequence), ROUGE-S (Skip-bigram), etc.
- **Python Package**py-rouge can be installed via pip.

NLTK (Natural Language Toolkit)

- **Tool** Various pre-trained models and corpora from NLTK
- **Reference** Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. O'Reilly Media, Inc.
- **Details**
  - **Components Used** Tokenizers (punkt), POS tagger (averaged_perceptron_tagger), Named Entity Chunker (maxent_ne_chunker), WordNet Lemmatizer (wordnet), Stopwords (stopwords).
  - **Functionality** Provides easy-to-use interfaces to over 50 corpora and lexical resources, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and more.
  - **Website** NLTK

Gradient Boosting Regressor

- **Library** scikit-learn
- **Reference** Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.
- **Details**
  - **Model** Gradient Boosting Regressor
  - **Functionality** Ensemble of decision trees that combines the predictions of several base estimators to improve robustness over a single estimator.
  - **Parameters** Number of estimators, maximum depth of the individual regression estimators, learning rate, etc.
  - **Documentation** Scikit-learn Gradient Boosting.