Some Techniques for Improving Extractive and Abstractive Automatic Text Summarization

Thesis submitted by

Sohini Roy Chowdhury

DOCTOR OF PHILOSOPHY (Engineering)

Department of Computer Science and Engineering,
Faculty Council of Engineering & Technology,
Jadavpur University
Kolkata, India
2024

JADAVPUR UNIVERSITY

KOLKATA-700032, INDIA

INDEX NO. 208/17/E

1. Title of the Thesis:

   Some Techniques for Improving Extractive and Abstractive Automatic Text Summarization

2. Name, Designation & Institution of the Supervisor/s:

   Prof. Kamal Sarkar

   Professor

   Department of Computer Science and Engineering

   Jadavpur University, Kolkata-700032, India

List of Publications

Papers in Journals

1. Sohini Roy Chowdhury, Kamal Sarkar (2023), A New Method for Extractive Text Summarization Using Neural Networks. SN COMPUT. SCI. Vol.4(4), DOI: https://doi.org/10.1007/s42979-023-01806-0, Springer(Scopus indexed, cite score 4.3)

2. Sohini Roy Chowdhury, Kamal Sarkar (2023), Sentence Fusion using Deep Learning. EAI Endorsed Transactions on Internet of Things. Vol. 10, DOI: 10.4108/eetiot.4605. European Alliance for Innovation (EAI), (Scopus indexed)

3. Santanu Dam, Kamal Sarkar. Sohini Roy Chowdhury (2018), A Study on Effect of Positional Information in Single Document Text Summarization. International Journal of Computing and Applications(IJCA). Vol. 16(1),pp 29-37, ISSN: 0973-5704.

Papers in conference procedings

1. Kamal Sarkar,Sohini Roy Chowdhury (2024). Improving Salience-based Multi-document Summarization Performance Using a Hybrid Sentence Similarity Measure, In: 4th International Conference on NLP & Text Mining (NLTM 2024), Copenhagen, Denmark,pp.15-29. DOI: 10.5121/csit.2024.140202.

2. Sohini Roy Chowdhury and Kamal Sarkar (2023), Abstractive Multi-Document Summarization Using Sentence Fusion, In: International Conference on Information Technology (ICIT), Amman, Jordan,pp. 734-741, DOI: 10.1109/ICIT58056.2023.10225941.

3. Sohini Roy Chowdhury, Kamal Sarkar, and Arka Maji (2022). Unsupervised Bengali Text Summarization Using Sentence Embedding and Spectral Clustering. In: Proceedings of the 19th International Conference on Natural Language Processing (ICON),Association for Computational Linguistics(ACL), New Delhi, India, pp 337–346.

4. Kamal Sarkar, Sohini Roy Chowdhury, (2021). A Novel Sentence Scoring Method for Extractive Text Summarization. In: Proceedings of International Conference on Frontiers in Computing and Systems(COMSYS), Advances in Intelligent Systems and Computing,(AISC,volume 1255), ISBN: 978-981-15-7834-2,Springer, Jalpaiguri, West Bengal, India, pp. 169–179. DOI: https://doi.org/10.1007/978-981-15-7834-2-16

5. Sohini Roy Chowdhury, Kamal Sarkar and Santanu Dam (2017), An Approach to Generic Bengali Text Summarization Using Latent Semantic Analysis,International Conference on Information Technology (ICIT), IEEE, Bhubaneswar, India, pp. 11-16, DOI: 10.1109/ICIT.2017.12.

List of Presentations in International Conference:

1. Kamal Sarkar,Sohini Roy Chowdhury (2024). Improving Salience-based Multi-document Summarization Performance Using a Hybrid Sentence Similarity Measure, In: 4th International Conference on NLP & Text Mining (NLTM 2024), Copenhagen, Denmark,pp.15-29. DOI: 10.5121/csit.2024.140202.

2. Sohini Roy Chowdhury and Kamal Sarkar (2023), Abstractive Multi-Document Summarization Using Sentence Fusion, In: International Conference on Information Technology (ICIT), Amman, Jordan,pp. 734-741, DOI: 10.1109/ICIT58056.2023.10225941.

3. Sohini Roy Chowdhury, Kamal Sarkar, and Arka Maji (2022). Unsupervised Bengali Text Summarization Using Sentence Embedding and Spectral Clustering. In: Proceedings of the 19th International Conference on Natural Language Processing (ICON),Association for Computational Linguistics(ACL), New Delhi, India, pp 337–346.

4. Kamal Sarkar, Sohini Roy Chowdhury, (2021). A Novel Sentence Scoring Method for Extractive Text Summarization. In: Proceedings of International Conference on Frontiers in Computing and Systems(COMSYS), Advances in Intelligent Systems and Computing,(AISC,volume 1255), ISBN: 978-981-15-7834-2,Springer, Jalpaiguri, West Bengal, India, pp. 169–179. DOI: https://doi.org/10.1007/978-981-15-7834-2-16

5. Sohini Roy Chowdhury, Kamal Sarkar and Santanu Dam (2017), An Approach to Generic Bengali Text Summarization Using Latent Semantic Analysis,International Conference on Information Technology (ICIT), IEEE, Bhubaneswar, India, pp. 11-16, DOI: 10.1109/ICIT.2017.12.

STATEMENT OF ORIGINALITY

I Sohini Roy Chowdhury registered on 15/09/2017 do hereby declare that this thesis entitled "Some Techniques for Improving Extractive and Abstractive Automatic Text Summarization", contains a literature survey and original research work done by the undersigned candidate as part of Doctoral studies.

All information in this thesis has been obtained and presented in accordance with existing academic rules and ethical conduct. I declare that, as required by these rules and conduct, I have fully cited and referred all materials and results that are not original to this work.

I also declare that I have checked this thesis as per the "Policy on Anti Plagiarism, Jadavpur University, 2019", and the level of similarity as checked by iThenticate software is 5%.

Signature of the Candidate: Sohini Roychowdhury

Date: 27/07/2024

KSarkar 27/07/2024

Certified by Supervisor:
(Signature with date,seal)

Professor
Computer Sc. & Engg. Department
Jadavpur University
Kolkata-700032

CERTIFICATE FROM THE SUPERVISOR

This is to certify that the thesis entitled **Some Techniques for Improving Extractive and Abstractive Automatic Text Summarization** submitted by Sohini Roy Chowdhury, who got her name registered on 15.09.2017 for the award of Ph.D. (Engineering) degree of Jadavpur University, is absolutely based upon her own work under the supervision of Prof. Kamal Sarkar, Department of Computer Science and Engineering, Jadavpur University, Kolkata-700032, India, and that neither her thesis nor any part of the thesis has been submitted for any degree/diploma or any other academic award anywhere before.

KSarkar 27/07/2024

....

Signature of the Supervisor
and date with Office Seal

Professor
Computer Sc. & Engg. Department
Jadavpur University
Kolkata-700032

Dedication

To my family

Acknowledgements

Finally, it is time for me to acknowledge all those who inspired me, supported me and helped me to get to the place where I am today.

I take this opportunity to express a deep sense of gratitude to Prof. (Dr.) Kamal Sarkar for his supervision and invaluable co-operation. This thesis would not have been possible without constant inspiration and unbelievable support of him over the last few years. The Journey would remain incomplete without his valuable suggestions, unconditional help, and encouragement.

I would also like to thank all of my colleagues from Computer Science and Engineering Department, Jadavpur University for providing me a friendly research environment.

Finally, most important of all, I would like to dedicate the thesis to my parents Mr. Pulak Roy Chowdhury and Late Mrs. Supriya Roy Chowdhury, my husband Mr. Ranajit Paul and my son Mr. Sourish Paul, to honor their love, patience, encouragement, and support during my research.

Sohini Roychowdhury

Date: 27/07/2024

(Sohini Roy Chowdhury)

Abstract

In the digital era of today, we have access to a vast amount of information with just one mouse click. To meet information needs, users have to spend a lot of time and effort consuming the vast amount of online textual information and identifying relevant information. Therefore, an automatic text summary system that can produce accurate and relevant summaries from a document or a set of related documents is needed to manage the information overload problem. Text summarization is a process that creates a condensed version of the input (a document or a set of documents) by selecting sentences with pertinent information from the document(s). Text summarization can be broadly categorized into two types: extractive and abstractive. An extractive summarization system aims to generate a summary by selecting textual segments or sentences from the input. On the other hand, an abstractive summarization system produces a summary from the input by reformulation, introducing new words or phrases which may not be present in the original document. When text summarization is performed on a single document, the summarization process is called single-document summarization. Multi-document summarization is a process to create a summary from a group of related documents. Both single-document and multi-document summaries can be extractive or abstractive. By analyzing existing research work on text summarization, it was observed that most existing works concentrated on making an extractive summarization system better and a handful of research works focused on developing an abstractive summarization system. Generating an abstract is relatively tougher than generating an extract, because abstract generation needs a deeper analysis of the input.

Although research on text summarization had started in the long past, text summarization problems have not yet been solved due to its toughness. Existing research on text summarization has progressed in three main directions - extractive summarization, abstractive summarization, and hybrid sumarization (which combines extractive and abstractive both). In recent times, the pre-trained word embedding model and LLM (large language model) have proven useful to capture semantic similarity between texts. Text-to-text generative models are also found to be effective in text generation. We have integrated these models with the summarization models to improve text summarization performance. This thesis contributes to the fields of extractive and abstractive summarization. We investigate novel techniques for single- and multi-document text summarization. We used word embeddings and spectral clustering of sentences in the unsupervised single-document summarization process. We also propose a new method for single-document extractive summarization that uses neural networks to trade off saliency and diversity in the summary. For extractive multi-document summarization, we define a new hybrid sentence similarity measure for dealing with redundancy in the multi-document summaries. We propose an abstractive multi-document summarization method that uses Text-to-Text generative model for fusing the candidate pairs of sentences and generating abstract multi-document summary by selecting the most significant fused sentences using the KL divergence criterion.

We have evaluated the proposed approaches using benchmark datasets and a Bengali summarization dataset developed by us. The proposed unsupervised single document extractive text summarization approaches are evaluated using the Bengali text summarization dataset. We have evaluated the proposed neural network-based extractive single document extractive text summarization method on benchmark summarization datasets, namely the DUC2002 dataset and Daily Mail(DM) dataset. Multi-document extractive and abstractive text summarization methods have been evaluated on benchmark English dataset named DUC2004 dataset. The proposed approaches are compared with some state-of-the-art approaches. Experimental findings and analyzes establish the effectiveness of the proposed methods for single-document and multi-document text summarization.

Keywords: Text Summarization, Single Document Summarization, Multi-document summarization, Extractive Summarization, Abstractive Summarization, Unsupervised Text Summarization, Spectral Clustering, Latent Semantic Indexing, Deep Neural Networks, Hybrid Similarity Measure, Text-to-Text Generative Model, Sentence Fusion, Multidocument Abstractive Summarization.

Contents

List of Figures

List of Tables

1 Introduction and Scope of the Thesis

1.1 Introduction

In today's digital era, we have access to a huge amount of information with just one mouse click. With a vast amount of information available online, it is a challenging task to remove the noise and identify accurate information. Due to the availability of a large amount of information coming from heterogeneous sources, users take a long time to read each document in the collection and find the relevant topic. This leads to information overload problems. The information overload problem affects the skill of human decision making. Instead of wasting time, it also affects human capacity. So, it is also very essential to develop an improved mechanism that can effectively represent information succinctly so that users can quickly understand and digest the main themes of the documents.

Text summarization is an effective solution to this information overload problem because it helps to create a condensed version (a summary) of one or more documents by selecting pertinent information from the input, and reading a summary helps the reader to satisfy their information need by quickly and easily understanding the salient points or main themes of the original document instead of reading all of the documents individually. However, text summarization is a human ability. When a human summarizes a document, they apply their cognition power to understand a document and create a gist of it. So, it is difficult for a machine to create a summary from a document or a set of related documents. Therefore, the important research question is: can the machine create a summary of one or more documents? Generic text summarization is a field that has seen increasing attention from the NLP(Natural Language Processing) community to address this question. Since manual summarization is tedious process, if machine is able to summarize automatically, we can assign a machine to this boring job and this enables the users to manage information overload problem by considering summaries in place of original documents or a document set.

When the display area of the device is small, for example, on mobile devices or other hand-held devices, reading the content of the entire document takes a long time. To reduce reading time, a short summary of a document may be displayed on the device instead of displaying the entire document, and users may use a summary instead of the original document to satisfy their information needs. Moreover, if users are satisfied with the summary, the original documents need not be transferred to the device. Thus summarization saves bandwidth.

Text summarization is a process that creates a condensed version of the input (a document or a set of documents) by selecting sentences with pertinent information from the document(s). Text summarization can be broadly categorized into two types: extractive and abstractive. An extractive summarization system aims to generate a summary by selecting textual segments or sentences from the input. On the other hand, an abstractive summarization system produces a summary from the input by reformulation, introducing new words or phrases which may not be present in the original document. When text summarization is performed on a single document, the summarization process is called single-document summarization. Multi-document summarization is a process to create a summary from a group of related documents. Both single-document and multi-document summaries can be extractive or abstractive. By analyzing existing research work on text summarization, it was observed that most existing works concentrated on making an extractive summarization system better and a handful of research works focused on developing an abstractive summarization system. Generating an abstract is relatively tougher than generating an extract, because abstract generation needs a deeper analysis of the input.

Although research on text summarization had started in the long past, text summarization problems have not yet been solved due to its toughness. Existing research on text summarization can be broadly categorized as extractive summarization, abstractive summarization, and hybrid sumarization (which combine extractive and abstractive both). The earlier approaches to text summarization used sentence level features such as keywords, cue phraes , sentence position , similarity to the headline, etc. for scoring the sentences [2][3][4][5][6][7][8]. Cluster-based summarization is another approach that has also been widely used for text summarization [9][10]. In the first part of the thesis, we focus on developing novel unsupervised text summarization approaches because these approaches are useful when sufficient training data is not available. In recent times, machine learning models, the pre-trained word embedding model, and LLM (large language model) and their combination have proven useful in capturing semantic similarity between texts. Text-to-text generative models are also found to be effective in text generation and abstractive summarization. In the next parts of the thesis, we have investigated finding effective text summarization models by incorporating the above-mentioned models which are capturing semantics of the texts.

This thesis contributes to the fields of extractive and abstractive summarization. We investigate novel techniques for single- and multi-document text summarization. We used word embeddings and spectral clustering of sentences in the unsupervised single-document summarization process. We also propose a new method for single-document extractive summarization that uses neural networks to trade off saliency and diversity in the summary. For extractive multi-document summarization, we define a new hybrid sentence similarity measure for dealing with redundancy in the multi-document summaries. We propose an abstractive multi-document summarization method that uses Text-to-Text generative model for fusing the candidate pairs of sentences and generating abstract multi-document summary by selecting the most significant fused sentences using the KL divergence criterion.

We have evaluated the proposed approaches using benchmark datasets and a Bengali summarization dataset developed by us. The proposed approaches are compared with some state-of-the-art approaches. Experimental findings and analyzes establish the effectiveness of the proposed methods for single-document and multi-document text summarization.

1.2 Definition of text summarization

According to Inderjeet Mani(1999)[11], ″Text summarization is the process of distilling the most important information from a text to produce a bridged version for a particular task and user″. According to Maybury (1995) [12] ″Text summarization is the process of distilling the most important information from a set of sources to produce an abridged version for particular users and tasks.″ Dictionary definition of term ″text summarization″ by breaking it down into its components: ″Text″ is referred to as ″any form of written material″, ″Summary″ is defined as ″A short statement that gives only the main points of something, not the details.″ Text summarization is the art of compressing longer written content, such as articles, documents, or webpages, into a more concise form while preserving its core information and meaning. Extractive summarization involves the selection and extraction of sentences or passages from the source text, whereas abstractive summarization involves generating summaries by rephrasing the content into a more compact format, often utilizing paraphrasing and natural language generation techniques.

1.3 Application of text summarization

Text summarization finds extensive use in a wide range of fields, offering solutions to condense information and improve user understanding. The following are some prominent examples of its application.

- **Information Retrieval**: Summarization serves to provide users with a quick glimpse into the core aspects of a document, aiding in the decision of whether to dive into the complete content. Search engines frequently employ summarization to offer brief excerpts in search listings.

- **Content Generation**: Its utility extends to crafting succinct and well-structured summaries for lengthy articles, reports, or research papers, proving valuable in the creation of abstracts or executive summaries.

- **News Summarization**: Text summarization is employed by news organizations to craft concise and to-the-point summaries of news articles, which are subsequently integrated into news aggregators, mobile applications, and email newsletters.

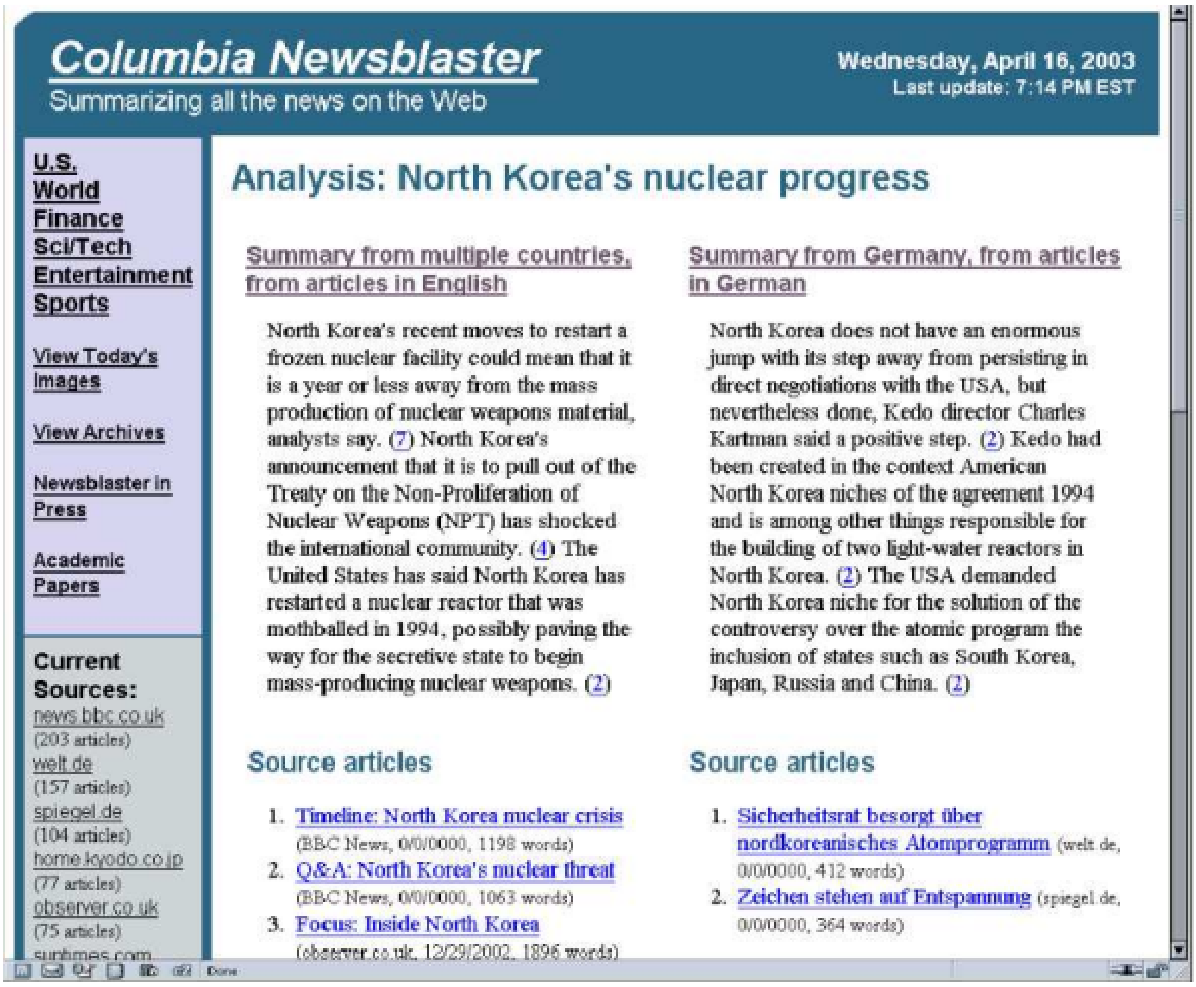
  Fig 1.1 presents the Columbia Newsblaster summarization application [1], which is a news summarization system that aims to tackle the challenge of allowing users to access news content from various languages and sources on the Internet. This system automatically gathers, arranges, and condenses news articles in multiple source languages, allowing users to explore news topics with English summaries and compare points of view from various countries.

- **Legal Documents**: Utilizing summarization techniques can facilitate the extraction of crucial information from legal documents, streamlining the process for lawyers and legal experts in reviewing contracts, court rulings, and other legal texts.

- **Medical Records**: In the healthcare sector, text summarization aids in condensing extensive medical records, simplifying the task for healthcare providers to swiftly comprehend a patient's medical background.

- **Social Media**: Social media platforms frequently employ summarization techniques to offer users concise renditions of lengthy posts or articles, enabling them to grasp the core points without having to go through the entire content.

- **E-commerce**: E-commerce platforms utilize summarization to craft succinct product descriptions, proving particularly valuable when managing extensive product catalogs.



Columbia Newsblaster
Summarizing all the news on the Web

Wednesday, April 16, 2003
Last update: 7:14 PM EST

U.S.
World
Finance
Sci/Tech
Entertainment
Sports

View Today's Images

View Archives

Newsblaster in Press

Academic Papers

Current Sources:
news.bbc.co.uk
(203 articles)
welt.de
(157 articles)
spiegel.de
(104 articles)
home.kyodo.co.jp
(77 articles)
observer.co.uk
(75 articles)
suntimes.com

Analysis: North Korea's nuclear progress

Summary from multiple countries, from articles in English

North Korea's recent moves to restart a frozen nuclear facility could mean that it is a year or less away from the mass production of nuclear weapons material, analysts say. (7) North Korea's announcement that it is to pull out of the Treaty on the Non-Proliferation of Nuclear Weapons (NPT) has shocked the international community. (4) The United States has said North Korea has restarted a nuclear reactor that was mothballed in 1994, possibly paving the way for the secretive state to begin mass-producing nuclear weapons. (2)

Source articles

1. Timeline: North Korea nuclear crisis (BBC News, 0/0/0000, 1198 words)
2. Q&A: North Korea's nuclear threat (BBC News, 0/0/0000, 1063 words)
3. Focus: Inside North Korea (observer.co.uk, 12/29/2002, 1896 words)

Summary from Germany, from articles in German

North Korea does not have an enormous jump with its step away from persisting in direct negotiations with the USA, but nevertheless done, Kedo director Charles Kartman said a positive step. (2) Kedo had been created in the context American North Korea niches of the agreement 1994 and is among other things responsible for the building of two light-water reactors in North Korea. (2) The USA demanded North Korea niche for the solution of the controversy over the atomic program the inclusion of states such as South Korea, Japan, Russia and China. (2)

Source articles

1. Sicherheitsrat besorgt über nordkoreanisches Atomprogramm (welt.de, 0/0/0000, 412 words)
2. Zeichen stehen auf Entspannung (spiegel.de, 0/0/0000, 364 words)

Done

Figure 1.1: Application of text summarization [1]

- **Chatbots and Virtual Assistants**: Summarization techniques are employed in chatbots and virtual assistants to deliver concise responses to user inquiries or to condense lengthy conversations.

- **Document Management**: Within corporate and institutional settings, document management systems can employ summarization to offer concise descriptions or abstracts of documents, simplifying the process of organization and information retrieval.

- **Educational Content**: Summarization techniques are employed to generate study guides, lecture notes, or video summaries, aiding students in swiftly comprehending the core concepts of a subject.

- **Content Curation**: Content curators leverage text summarization to sift through and condense pertinent content from the internet, which they then use for newsletters, blogs, or various publications.
- **Legal and Compliance**: In the realms of law and regulation, summarization serves as a valuable tool for identifying and extracting pertinent information from extensive volumes of legal documents, ensuring adherence to compliance standards.
- **Market Research**: Summarization techniques can be employed to extract insights from market research reports, customer feedback, and surveys, empowering businesses to base their decisions on data-driven insights.
- **Scientific Research**: Utilizing summarization methods enables the condensation and extraction of pivotal findings from scientific papers, simplifying the task for researchers to stay abreast of the most recent advancements in their respective fields.
- **Language Translation**: Summarization aids in the translation of lengthy texts by furnishing concise summaries in the target language, assisting translators in comprehending the content.
- **Email Management**: Email clients have the capability to utilize summarization for delivering previews or concise summaries of lengthy emails, facilitating users in managing their inbox effectively.

1.4 Classification of text summarization system

Text summarization systems can be classified based on some factors such as input size, information content, type of summary produced, summarization domain, processing level, nature of the output summary, summary language, and summarization algorithm. Figure 1.2 shows the detailed categorization of the text summarization.

1.4.1 Classification of Text summarization-based on the nature of summary produced by the system

Based on the nature of summary produced by the systems, they can be classified as extractive and abstractive summarization systems.

- **Extractive summarization:** Idea behind extraction based text summarization is to pick up sentences from the original source document(s) after identifying salient sentences. A machine can extract the important fragments of

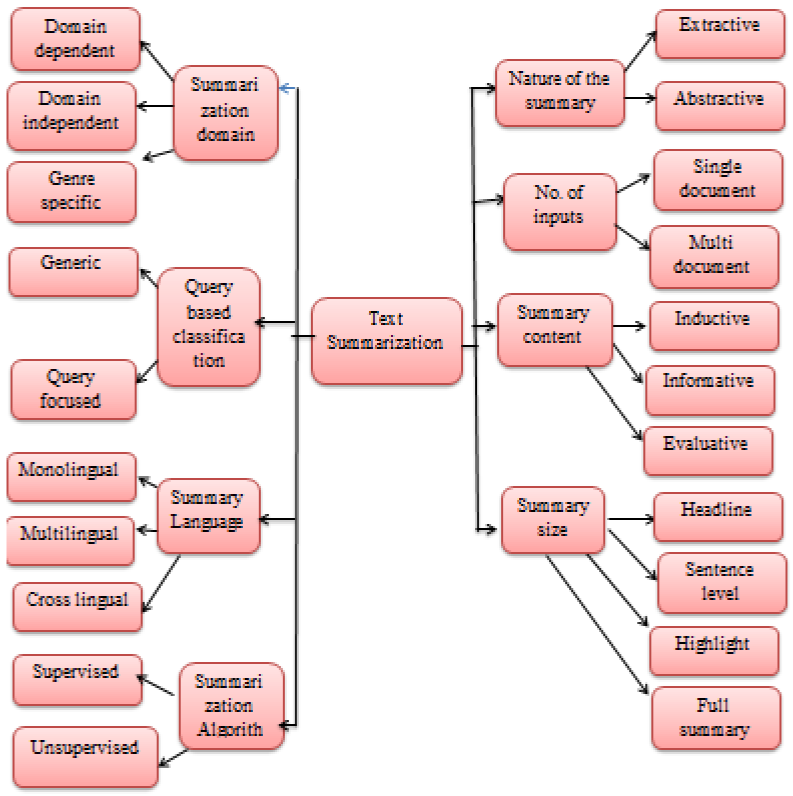
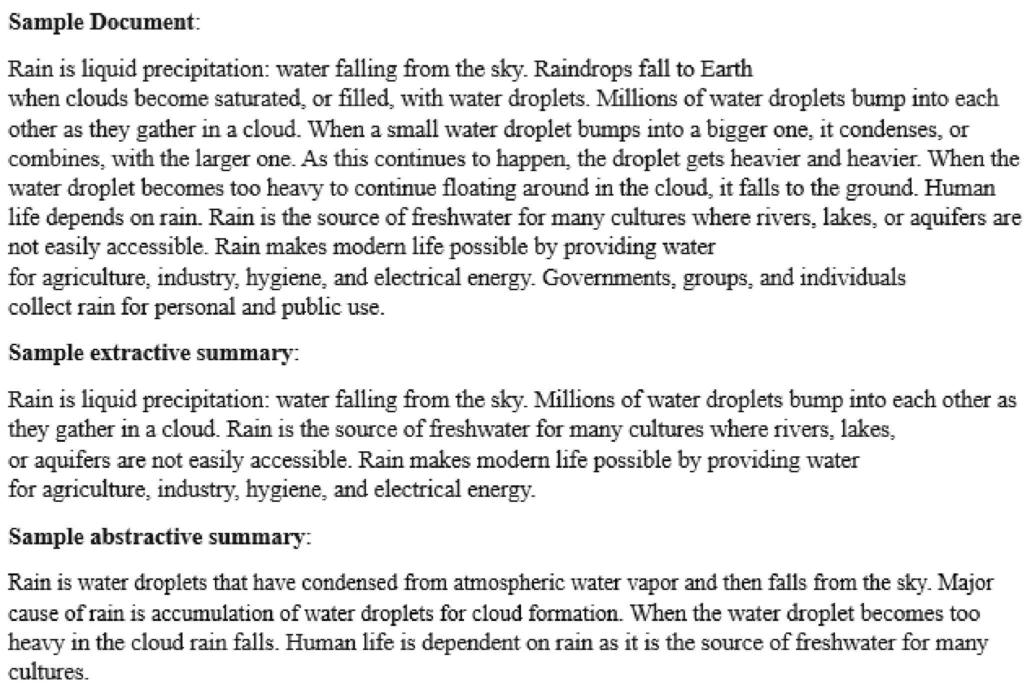


Figure 1.2: Classification of text summarization systems

the text from the document(s) to generate a summary. Most existing text summarization systems are sentence extraction based [13]. The sentence extraction based summarization methods assigns scores to each sentence based on sentence position[14][3], sum of the weights of the terms contained in the sentence[3][2], sentence centrality[15] which is computed based on the number of other sentences the sentence links to, number of cue phrases contained in the sentence etc. and then combines the various scores to assign a unique score to the sentence. Finally a summary is generated by choosing K top ranked non-redundant sentences and reordering the sentences according to their appearances in the text(s).

- **Abstractive summarization:** Abstraction based summary focuses on producing human-like summary. It aims to mimic human cognition for generating abstract. Abstractive text summarization involves understanding the underlying meaning of the text and using natural language generation (NLG) algorithms to generate a shorter summary by means of techniques such as paraphrasing, substituting synonyms, compressing, and fusing sen-

tences. This process is more similar to human-written summaries, as it aims to capture the essence of the original text(s) in a condensed form. An abstract may contain words or phrases which are not present in the input. Compared to extracting, abstracting is a more difficult task because it aims to imitate the human ability of abstracting. Some post processing techniques can also be performed on the output of extractive summary to create abstract. These techniques is to improve the coherence and cohesion in the summary[16][17][18][19][20]. Researchers in summarization domain recognized this problem early on and existing works in this context include sentence compression [21], [22]sentence fusion[23][24] and sentence ordering [25][26][27]. Abstraction aims to produce important material in a new way; fusion combines extracted parts coherently; and compression aims to throw out unimportant sections of the text.
An example of a sample extractive summary and abstractive summary is shown in Figure 1.3.



Sample Document:

Rain is liquid precipitation: water falling from the sky. Raindrops fall to Earth
when clouds become saturated, or filled, with water droplets. Millions of water droplets bump into each other as they gather in a cloud. When a small water droplet bumps into a bigger one, it condenses, or combines, with the larger one. As this continues to happen, the droplet gets heavier and heavier. When the water droplet becomes too heavy to continue floating around in the cloud, it falls to the ground. Human life depends on rain. Rain is the source of freshwater for many cultures where rivers, lakes, or aquifers are not easily accessible. Rain makes modern life possible by providing water
for agriculture, industry, hygiene, and electrical energy. Governments, groups, and individuals
collect rain for personal and public use.

Sample extractive summary:

Rain is liquid precipitation: water falling from the sky. Millions of water droplets bump into each other as they gather in a cloud. Rain is the source of freshwater for many cultures where rivers, lakes,
or aquifers are not easily accessible. Rain makes modern life possible by providing water
for agriculture, industry, hygiene, and electrical energy.

Sample abstractive summary:

Rain is water droplets that have condensed from atmospheric water vapor and then falls from the sky. Major cause of rain is accumulation of water droplets for cloud formation. When the water droplet becomes too heavy in the cloud rain falls. Human life is dependent on rain as it is the source of freshwater for many cultures.

Figure 1.3: Example of extractive and abstractive text summarization

1.4.2 Input size-based classification:

Based on number of input documents, the text summarization system can be classified into a single document text summarization system and a multidocument text summarization system.

- **Single document summarization system:** In case of single document text summarization, only one document is taken as input to summarize. Computations are performed by decomposing a document into sentences to identify summation-worthy sentences. So, for summarizing a long document, single-document text summarization is useful.

- **Multi document summarization system:** Multi document text summarization aims to create summary by taking multiple related documents as input. As there is a chance of recurrent information throughout various sources like news articles, it increases requirement for a tool which can remove redundancy. Multi document text summarization is an useful approach to solve this problem as it focuses on generating summary from various related articles by removing repetitive information. It also aims to capture both similarities and differences among collection of documents while generating a summary.

1.4.3 Content-based classification

Based on the summary content, the text summarization system can be classified into an indicative, informative, evaluative or critical summarization system.

- **Indicative:** An indicative summary of a document is short and objective. It helps readers decide whether the original document needs to be consulted or not.

- **Informative:** Informative summary is referred to as an explanatory abstract, which is a succinct and unbiased outline of a document, scholarly article, research paper, or study. Its objective is to furnish readers with a lucid comprehension of the principal ideas, noteworthy discoveries, and significant facets of the original text. Diverging from an indicative summary, which concentrates on encapsulating the content, an informative summary incorporates vital particulars, outcomes, conclusions, and ramifications of the source material. It delivers a condensed rendition of the document while still communicating the indispensable information. Compared to an indicative summary of a document, its informative summary is long.

- **Evaluative or Critical:** Evaluative or critical text summarization represents a category of summarization that transcends the mere compression of information. Instead, it aims to deliver an appraisal or critique of the underlying content. Within this framework, the summary not only accentuates primary points and key details but also integrates an evaluation of the content's caliber, pertinence, merits, demerits, or overall importance. Evaluative or critical text summarization finds widespread application in diverse domains, including academic literature assessments, product appraisals, critiques of films or books, and expert analyses. It furnishes readers not only with a concise rendition of the content but also with a discerning viewpoint that aids them in forming well-informed judgments or decisions regarding the subject material.

1.4.4 Summary size-based classification

Based on the size of the summary, the summarization system can be classified into headline, sentence level, highlight, and full summary.

- **Headline**: Headline summary generation entails the creation of brief and informative titles that effectively encapsulate the primary content and key highlights of longer textual pieces. These headlines are commonly employed in news articles, blog entries, and various content formats, serving to offer readers a swift glimpse into the core focus or significance of the material. The primary objective of headline summary generation is to distill essential information into a succinct and attention-grabbing statement, enticing readers to delve further into the complete article. In general headline summary is very short containing 10 to 20 words. It is challenging to automatically craft captivating and informative headlines. For content creators, publishers, and news agencies, headline summary generation proves to be an invaluable tool, enhancing reader engagement and simplifying the decision-making process for audiences contemplating whether to explore the full article [23].

- **Sentence level**: Sentence-level summary generation involves the creation of brief and meaningful summaries that focus on individual sentences within a longer text or document. This approach aims to extract or generate sentences that encapsulate the key information or significant points from the source text. Sentence-level summarization finds application in various natural language processing (NLP) tasks, including document summarization, text extraction, and the generation of content for chatbots or virtual assistants. Its purpose is to offer readers or users concise and standalone pieces of information that retain the core essence of the original content. In the context of sentence-level summarization, the primary objective is to condense the content effectively, enhancing accessibility and comprehension for the intended audience. This can be accomplished through methods such as sentence extraction, abstractive summarization, or by leveraging machine learning models tailored for this specific task.

- **Highlights**: Highlight summary creation involves the process of crafting succinct and eye-catching summaries that spotlight the most crucial or standout elements of a text or document. In highlight summarization, the primary aim is to pinpoint and present pivotal highlights, essential details, or standout components from the content, often leveraging techniques like text analysis, keyword extraction, or sentiment analysis. These summaries are typically more concise and sharply focused compared to traditional summaries. They are designed to swiftly capture the reader's attention by showcasing the most compelling or pertinent portions of the text. Highlight summaries find utility in various contexts, including news articles, product reviews, social media posts, and marketing materials. For content creators and marketers seeking to convey the most impactful information effectively, highlight summary generation proves to be an invaluable tool. Highlight summary is usually presented in bulleted form. It enables readers to swiftly grasp the essence of the content and make informed decisions about exploring it further.

- **Full summary**: Full summary generation is the process of crafting extensive and intricate summaries of lengthy texts, documents, or content pieces. Diverging from briefer summarization approaches like sentence-level or highlight summarization, the primary objective of full summary generation is to furnish readers with a profound grasp of the source material by compressing it into a meticulously organized, logically coherent, and information-rich summary. Full summary generation finds utility across diverse domains, encompassing academic research, literature surveys, content curation, and information retrieval. In these contexts, it functions as a valuable instrument for aiding individuals in efficiently accessing and comprehending extensive textual content.

1.4.5 Domain-based classification

Based on domains, text summarization systems can be classified into domain-dependent, domain-independent, and genre-specific text summarization systems.

- **Domain dependent**: Domain dependent text summarization involves the task of crafting succinct and logically structured summaries of textual materials confined to a specialized domain or subject matter. Domain-dependent summarization is tightly tailored to a specific realm, whether it be a field of study, industry, or a particular thematic focus. Numerous domain-specific summarization methodologies employ information extraction techniques [28] to pinpoint key information within documents. This domain-driven research also encompasses methods for Report Generation [29] and the summarization of Events sourced from specialized databases. In practice, domain-dependent text summarization proves its worth when the goal is to cater to a particular audience and offer valuable insights within a specialized domain or industry.

- **Domain independent**: Domain-independent text summarization, often referred to as generic text summarization, entails the task of generating concise and logically structured summaries of textual content without using domain-specific expertise. It involves compressing a text into a concise form while preserving its essential content and main concepts, irrespective of the particular subject matter or discipline. This methodology strives to generate summaries that can be applied universally across different domains or subjects, rendering them adaptable and valuable for diverse audiences and contexts. The majority of domain-independent methods employ statistical techniques, often in conjunction with sturdy or superficial language processing technologies, to identify significant portions of documents. These statistical techniques encompass vector space models, term frequency, and inverted document frequency, as indicated in previous studies [30]. In this case, the range of language technologies employed spans from domain independent lexical cohesion methods [31] to domain independent anaphora resolution techniques [32].

- **Genre specific**: Genre-specific text summarization involves the task of crafting concise and logically structured summaries of textual materials, each finely attuned to a specific genre or category of written or spoken discourse. In this context, 'genre' signifies a distinct style, form, or class of communication, encompassing domains like news reporting, scientific research literature, legal documents, or literary creations. It involves condensing a text into a shorter form while considering the distinct traits and demands of a specific genre or type of writing. This method customizes the summarization process to accommodate the particular style, language, and content commonly found in the genre.

1.4.6 Query-based classification

Based on whether the summary produced is specific to a given query or not, text summarization systems can be classified as generic and query-focused.

- **Generic:** A generic summary is aimed at a wide readership or a general public. Its purpose is typically to produce a summary of a more extensive body of work, such as an article, a book, or a report. Generic summaries are often composed in a manner that captures the essential aspects or core concepts of the original text(s), while avoiding excessive elaboration. They

are crafted to be informative and easily understandable by a diverse range of readers, regardless of their specific knowledge or expertise.
The architecture of the generic text summarization is shown in Figure 1.4. Here, a document is passed to a preprocessing stage. After preprocessing, the document is represented using embedding-based features or hand-crafted features. After that, sentences are assigned score and top ranked sentences are selected for post processing. After post-processing, a summary is generated.

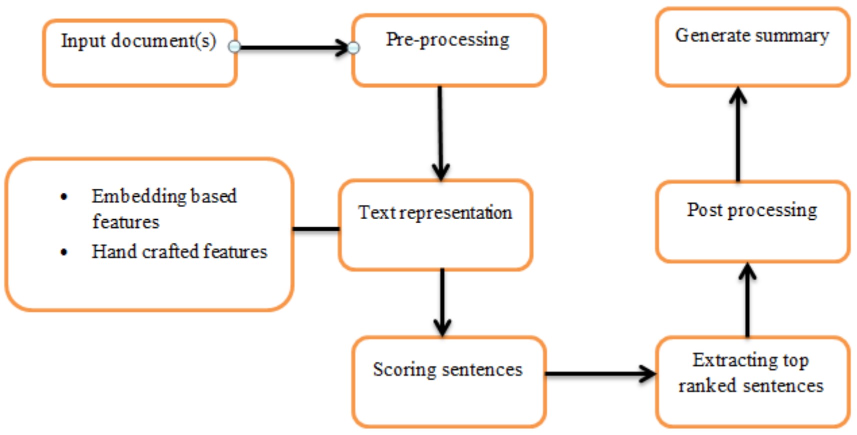


Figure 1.4: Architecture of generic text summarization

- **Query focused:** Query focused summaries summaries are crafted to meet the needs of a specific query asked by user. Here summary considers a depiction of the users' preferences, which may vary from comprehensive user models to profiles storing subject area keywords or even a specific query that includes terms that express the information requirement. An intriguing variation of query-focused summary is a summary generated as a response to a query. In this case, the summary should encompass the information that is pertinent to the query.
  The architecture of query-focused text summarization is shown in Figure 1.5. From the architecture of the query-focused summarization system, we can see that the query and the document are input to the model. Then it goes to the preprocessing step. Sentences are represented using embedding-based features, hand-crafted features, query-related features,similarity to query-based features. Sentences are assigned scores based on similarity to query- and sentence-content-based features. The top sentences are selected as a summary.

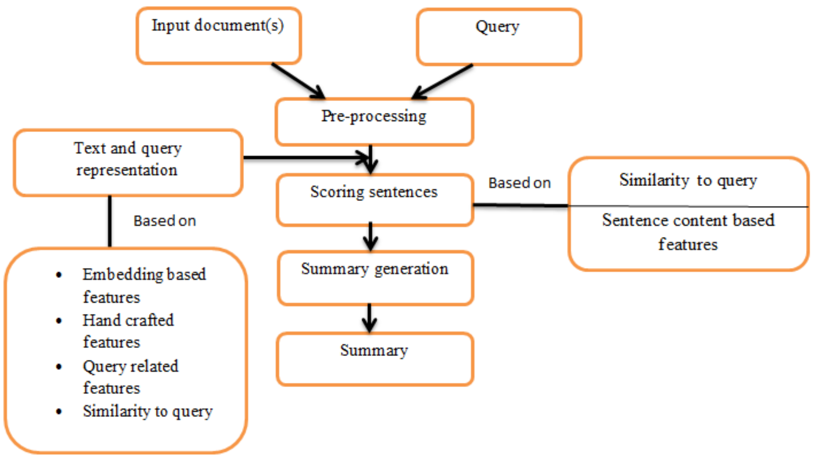


Figure 1.5: Architecture of query focused text summarization

1.4.7 Language-based classification:

Based on the type of languages, text summarization systems can be classified as monolingual, multilingual, and cross-lingual.

- **Monolingual**: Monolingual text summarization refers to the process of creating a summary of a single text within the same language, without using any translations or multilingual resources. Essentially, it revolves around the task of summarizing a document, article, or piece of content in the very language in which it was originally composed. This approach proves invaluable in situations where the objective is to furnish succinct and comprehensible renditions of textual material while upholding the language and contextual nuances of the source document. As such, it has an extensive application in tasks related to information retrieval and content summarization in various domains [33].

- **Multilingual**: Multilingual text summarization refers to the process of producing summaries from documents written in multiple languages. Its objective is to craft succinct and logically structured summaries that encapsulate the core points or essential information found in the source texts, regardless of the languages employed. In today's world, multilingual summarization plays a pivotal role by breaking down language barriers and facilitating access to vital information from a variety of sources in languages comprehensible to individuals and organizations [34].

- **Cross lingual**: Cross-lingual summaries encompass succinct and logically structured summaries of textual content, skillfully crafted to span across diverse languages. Their primary goal is to offer a purposeful and enlightening summary of materials originally authored in one language, now intelligibly presented in another language, thereby effectively bridging linguistic divides. In today's globally interconnected landscape, where information is disseminated in numerous languages, cross-lingual summaries serve as indispensable tools for fostering communication, facilitating information exchange, and supporting informed decision-making, empowering both individuals and organizations to leverage valuable content from a wide array of sources in languages that resonate with them.

1.4.8 Summarization algorithms-based classification:

Based on summarization algorithms used in text summarization systems, the text summarization system can be classified into supervised and unsupervised.

- **Supervised**: Supervised text summarization constitutes an approach to summarization that relies on annotated training data to train machine learning models in summarizing document(s). In this method, the model is fed a collection of input-reference summary pairs. Through the training process, the model learns to produce summaries by predicting and generating content that closely aligns with the reference summaries for a given input. This supervised approach has garnered substantial traction across a spectrum of natural language processing applications, presenting a methodical and data-driven means of generating summaries that harmonize with human expectations and preferences. Text summarization approaches using supervised algorithm is discussed later in this chapter.

- **Unsupervised**: Unsupervised text summarization is a method of generating summaries from textual content that does not rely on annotated training data. Instead, it operates by identifying and extracting important information based on the inherent structure, patterns, and content within the text itself. This approach proves especially valuable in situations where obtaining labeled training data is challenging or impractical. Unsupervised text summarization techniques draw upon a blend of natural language processing, clustering, and information retrieval methodologies to autonomously produce concise and coherent summaries from the input. Various unsupervised algorithms for generating extractive summaries are discussed in detail later in this chapter.

1.5 Existing text summarization approaches

We have surveyed existing research papers and identified the main text summarization approaches, and we have observed that most approaches are applied to extractive and abstract summarization. So, we have divided the existing approaches into two main groups: Extractive and abstractive. For each of these cases, the approaches differ when the input contains multiple documents. Therefore, in this section, existing approaches are divided into four groups: single-document summarization through extraction, single-document summarization through abstraction, multidocument summarization through extraction, and multidocument summarization through abstraction.

1.5.1 Single-Document Summarization through Extraction

We have reviewed various single-document text summarization methods. We have categorized the existing single document text summarization methods into three groups:

- Unsupervised methods
- Traditional machine learning-based methods
- Neural network-based methods

1.5.1.1 Unsupervised methods

The earlier work on unsupervised summarization presented in 1958 by Luhn [2] measured the importance of sentences based on the high frequency of words contained in them. Another system presented in [3] used the standard keyword-based method (keywords are words (except stop words) whose frequency is greater than a threshold) and the following three heuristic methods to determine the importance of sentences.

- Cue Method: Its basically gives impact on the most relevant sentences which is measured by the presence or absence of certain cue words, etc. in the cue dictionary.
- Title Method: Here, sentence weight is computed based on overlap between the sentence and the title or subheading.
- Location Method: This is based on the assumption that the highly relevant sentences occur earlier in the document.

The method [2][3][4][5][6][7][8] calculates scores based on various sentence-level features and assigns to textual units such as sentences, paragraphs, sentence segments of the document and extracts the units with higher scores to create a summary. On the other hand, unsupervised abstractive summarization methods extract the first important text units from the document and use information fusion, compression, and reformulation to generate the abstract of the document [35][36][37][38].

Generic text summarization approaches can be broadly categorized into different groups, Heuristic approaches that generate extractive summary after ranking sentences based on some heuristics such as sentence position within the document or whether a sentence contains any word occurring in the title of the document [3]. The next group includes text summarization approaches that are based on a document corpus (corpus-based methods). An example of such a method is TF*IDF (term frequency ·inverse document frequency). MEAD [10] is a text summarization tool that can be applied to both multi-document and single-document text summarization. MEAD uses a centroid-based summarization technique, which selects summary sentences based on their similarities with the centroid, which is a collection of document words whose TF*IDF weights are higher than the predefined threshold. MEAD also considers heuristics like sentence length, position, etc. for improving the centroid-based summarization performance. The other group consists of summarization methods which take a discourse structure into account. For example, the text summarization method using lexical chains presented in [39] searches for the chains of context words in the text and selects sentences based on the information contained in the strongest chains. The fourth group consists of approaches that represent the document as a graph whose vertices correspond to text segments such as sentences or paragraphs and edges correspond to interrelationships between the sentences. For example, a graph affinity-based text summarization approach has been presented in [40].The last group is called knowledge-rich approaches which require extensive encoding of word knowledge and domain knowledge. For example, the FRUMP system in [41] was supposed to map fragments of text to rich conceptual structures such as sketchy scripts, which contained information about the key elements of specific story types (e.g., earthquakes, demonstrations). The summarization system presented in [42] uses domain knowledge related to the medical domain such as MeSH (Medical Subject Headings) is NLM's (U.S. National Library of Medicine) controlled vocabulary thesaurus. Many experimental research works established the fact that the summarization performance is improved when automatic summarizers use the combination of the above stated methods [4][5][6][7].

Some unsupervised learning-based approaches use clustering of sentences by grouping similar sentences into clusters and generate summary by taking representative sentences into the summary. Most early text summarization algorithms faced the redundancy problem or the diversity problem. So, to deal with these problems and ensure good coverage, a clustering of sentences was used [9]. The idea of employing a clustering algorithm for text summarization was well described in [10]. This approach used three steps for text summarization: histogram-based clustering algorithm for sentence clustering, ordering of clusters, and extraction of summary-worthy sentences from the clusters to create the summary. In [43], a hierarchical agglomerative clustering algorithm was used to create clusters of sentences. To create a summary, the sentences were chosen in order from the largest to the smallest cluster. Another clustering-based approach presented in [44] incorporates cluster-level information in a graph model for ranking sentences. However, early work [10] suggested that the performance of clustering-based text summarization is highly dependent on the quality of the clusters produced. Clustering algorithms perform well when we have a clear idea of the attributes of data points [45]. Compactness-based clustering highlights spatial proximity among data points. For example, agglomerative average link clustering [9], k-means [46], highlights compactness. However, the resultant clusters that use this algorithm are spherical clusters. Modification of k-means was discussed in [47] which is defined as k-means++. Though it uses a better centroid initialization technique for improvements over k-means, still it suffers from some drawbacks because we need to specify the number of clusters to be formed in advance and it assumes a fixed shape of clusters. After investigating different existing clustering algorithms, we can find that spectral clustering is more suitable for our unsupervised summarization task. It embeds sentences in a low-dimensional Eigen space and performs clustering on the data points mapped to the low-dimensional embedding space. It does not assume any fixed shape of the cluster, but rather emphasizes graph partitioning [48] based on connectedness among the vertices representing the data points. So, it is very useful when the shape of cluster is non-convex. Today, the spectral clustering algorithm has been used in a wide range of application areas such as image clustering [49], shape clustering [50], motion clustering [51] and many more. Gupta et al.[52] presented a text summarization approach based on spectral clustering, which uses textual inclusion (TE) and spectral clustering (ATESC) to calculate sentence connectedness scores. It is used to measure the saliency of a sentence in the input. A maximum entropy-based model for the text summarization system proposed in [53]. In this work, features are word pairs, sentence length, sentence position, and discourse features (e.g., whether sentence follows any heading like 'Introduction', etc.) to choose the most salient sentences from the document in a summary.

Graph-based approaches are another process under this category, where a document is represented as a graph, sentences or paragraphs are represented as vertices, and edges are interrelationships between sentences [40]. The sentences are ranked on the basis of the centrality scores of the sentences. The centrality score for a sentence is defined by the degree of the sentence in the graph [54]. The variant of the PageRank algorithm is also used to calculate the centrality score [55]. Latent Semantic Indexing (LSI) [56] is an unsupervised way to create a summary using Singular Value Decomposition (SVD) from a term-by-sentence matrix created from the document. In this approach, a sentence is represented by a dense vector in the concept space and the sentence score is computed based on its similarity with the latent concepts in the concept space obtained through the SVD process. In [57], key concepts are identified and then a summary is produced by extracting a subset of sentences that maximize the key concepts in the summary. This is also an unsupervised approach. The manifold ranking approach [58] to topic-focused summarization assigns scores to sentences based on the information richness of each sentence biased toward a given topic and information novelty, which is measured by imposing a diversity penalty for each sentence.

1.5.1.2 Traditional Machine Learning-Based Methods

Most supervised machine learning approaches use a corpus of document-summary pairs to train machine learning (ML) algorithms that predict the summary worthiness of sentences in a document [59][60]. For training a supervised machine learning algorithm, sentences are represented by a set of hand-crafted features and are labeled summary worthy (Yes) or summary unworthy (No) using a manual or semi-automatic process. The supervised machine learning models that are widely used for extractive text summarization are: Naive Bayes, Artificial Neural Networks (ANN), Support Vector Machines (SVM), Hidden Markov Models (HMM), etc. Naive Bayes classifier was used in [61]. With this classifier, the authors developed an automatic text summarization system 'Dimsum' that used a WordNet dictionary to improve summary quality.

The Naive Bayes method can be applied for the extraction of key words [62] from a document to generate a summary. The Naive Bayes Classifier was also applied to the output of the latent semantic analysis technique to select summary sentences [63]. Text summarization can be performed using the hidden Markov model (HMM) [64]. This method uses a feature set that includes the position of the sentence, the number of terms in the sentence, and the likelihood of words. The position dependence, feature dependency, and markovity are handled by HMM. The probability that a sentence is selected in summary varies with the features of the current sentence. Another HMM-based approach has been proposed in [37] for the headline generation task. In this approach, the headline generation task is modeled as finding a set of words H that maximizes the likelihood that H is a headline of a given story S. QCS [65] is a summarization system that uses the Hidden Markov Model to generate a single document extract for each document belonging to a document cluster and the summary for the cluster is produced by choosing sentences from the single document extracts. The Hidden Markov Model (HMM) was also used in headline generation (very short summary) [36]. For headline generation, it generates the most likely sequence of words, given the document. The support vector machine (SVM) is a supervised learning model that is also applied to extractive text summarization. The SVM-based method proposed in [66] improves the performance of query-focused text summarization. It can detect relevant information to be included in a query-focused summary. The one-Class Support Vector Machine (OSVM) based approach proposed in [67] can learn from positive examples only, but suffers from underfitting and overfitting problems when data are inadequate. A CRF (Conditional Random Fields)-based summarization model proposed in [68] views summarization as a sequence learning task and assigns binary labels to the sentences of the input document based on the labels assigned to the previous sentences. The machine learning algorithm presented in [60] uses a meta-learner called 'bagging' that uses the C4.5 decision tree as a base learner. In this meta-learning approach, the main idea is to train a bag of decision trees on various instances of the training set and combine decisions of the trained decision trees to mark a sentence as summary-worthy or not.

1.5.1.3 Neural network-based methods

The extractive summary can be generated using neural networks. A Neural Network is trained to learn to include salient sentences in summary [69]. It is trained using a manually labeled training dataset. The sentences are represented as a set of hand-crafted features [70]. Abstract features are learned at the hidden layers of the neural network and the output layer provides output as probability vectors that determine whether a sentence is summary worthy or not. The work presented in [69] considers sentence representation as a numerical vector using the Word2Vec library. Recently, neural network-based approaches for extractive text summarization have become popular. The idea of a recursive autoencoder has been used for text summarization [71] and various natural language applications [72].

An approach to query-focused summarization that incorporates the idea of a deep autoencoder has been proposed in [73]. In this approach, document-wise local vocabulary is maintained, and normalized term frequency is used for representing sentence vectors submitted as the input to a deep autoencoder which is a variant of a denoising autoencoder that maps sentences to an abstract concept space. The sentences are then ranked by computing relevance scores based on the similarity between embedded presentations of each sentence and query.

SummaRuNNer [74] is a simple recurrent neural network (RNN) based summarization model that considers summary generation as a sequence learning task. This model scans the original document from the beginning and starts assigning 0 (summary unworthy) or 1 (summary worthy) labels to sentences based on the information contained in the current sentence and the labels of the sentences previously encountered. The multilayer Extreme Learning Machine (ELM) classifier based on deep network architecture [75] has been applied to extractive text summarization. This considers many features to discriminate between important and unimportant sentences. The model is trained with manually labeled important and unimportant sentences. In [76], a new deep learning-based extraction technique is proposed for query-focused summarization. The framework presented in [76] uses a deep unsupervised model to extract concepts from the deepest hidden layers, and the sentence that shares the concept terms present in the query is given greater importance. For the generation of a summary, this model used dynamic programming to choose a subset of sentences that maximizes importance within a given summary restriction. The attention-based encoder-decoder model for extractive text summarization has been presented in [77]. This model extracts sentences or words. For sentence extraction, CNN-based sentence representations are sent as input to attention-based RNN which sequentially assigns 1 or 0 labels to the sentences of the input document. On the other hand, the word extractor, which is a slight variant of the sentence extractor, performs a generation task by predicting the next word in the summary. A RNN-based model [22] that uses hand-crafted word level features as input and recursively computes the parent node representation in a parse tree by combining its pair of children. A regression process is applied at all the nonterminal nodes to measure the importance of the non-terminal nodes using the learned representation, and a sentence-level ranking score is obtained at the root node.

1.5.2 Single-Document Summarization through Abstraction

Extractive systems' summaries contain redundancy and lack coherence since sentences taken from many documents or portions of the same text might not make sense when combined. Single-Document summarization through abstraction is the process of distilling the key points and presenting a single document into a cohesive succinct summary through the creation of inventive, re-worded, or abstract content through the fusion of multiple sentences and reformulation. This approach is not the same as extraction-based summarization, which selects and assembles pre-existing sentences or phrases from the original content. Rather, the goal of abstractive summarization is to provide summaries that are more eloquent, coherent, and suitable for the context, rather than necessarily exactly replicating the original text's phrasing. With the advent of Transformer-based models like

GPT and BERT, which perform exceptionally well in generative summarization tasks, this field has made significant strides.

The earlier work on abstract summary generation focused on post-processing and revising extractive summaries to increase the coherence and cohesiveness of the summary. This was a transition to an abstract summarization [16] [17] [18] [19] [20]. Previous work in this area also includes sentence ordering [25] [26] [27], sentence fusion [23] [24], and sentence compression [21] [22] to improve cohesiveness. In these approaches, the main goal was to remove unnecessary portions of the text, create new content through the production of important material, and logically merge the extracted components. In an effort to produce abstractive summaries, a graph-based multi-sentence compression method was introduced in [78]. One way to think of sentence compression is to produce a tiny form of text summary. Previous studies concentrated mainly on the removal of superfluous words [79]. However, some works used sequence-to-sequence neural network models to generate an abstract summary [80][81][82]. However, the goals of contextual alignment or compression in these methods might not meet the demands of a lengthy text summary. When summarizing lengthy texts, it is not necessary for the summaries to accurately reflect every aspect of the original articles, and simply compressing the text might not be enough to exclude intricate details. Experiments conducted by [80] also showed this pattern, noting that model performance was influenced by input length. This suggests that it is difficult to directly apply sentence compression approaches to the longer text summarization task.

Research into longer text inputs and wider summarizing circumstances has since surfaced. [83] presented the Abstract Meaning Representation-based (AMR) solution; nevertheless, it requires an additional AMR-to-text model, and finding suitable training data for low-resource languages is difficult. A reinforcement learning model was utilized to produce summaries that improve the extraction of keywords from the source materials [84] fine-tuned two large-scale pre-trained models, BERT [85] and GPT-2 [86] to increase the generated summaries' fluency and coverage, respectively. [87] presented a novel framework for unsupervised abstractive summarization using Generative Adversarial Networks (GAN). Their methodology was based on the idea that, given an input document, the generator should output a condensed human-readable text that contains enough information to allow the reconstructor to reconstruct the original document. To determine whether the generated text is machine-generated or human-readable, they employed the discriminator of the GAN. Their method is especially appropriate for the circumstance, where solutions should not rely on large pretraining corpora or paired data, as it does not require any additional data or pre-trained models.

Other text summarization methods are customized for different settings [88] [89], or are based on the target domains; examples include review summarization [90], meeting speech summarization [91], and summarization of five-sentence stories [92]. These methods frequently make use of assumptions or strategies unique to a given domain.

1.5.3 Multidocument Summarization

Summarizing multiple documents, a relatively recent task, involves creating a unified summary for a collection of interconnected source documents. The three primary challenges posed by this task include (1) detecting and managing redundancy, (2) identifying significant distinctions between documents, and (3) maintaining summary coherence, even when the content originates from various source documents.

1.5.3.1 Multidocument Summarization through extraction

Different approaches for performing extractive multi-document text summarization have been proposed over time. Initial studies on extractive summarization involve ranking sentences using basic features such as sentence position, term frequency, or specific key phrases [93, 94, 2, 42, 95]. The next step is to select the top *n* non-redundant sentences based on the compression ratio. In [96], a method based on information extraction is introduced for multi-document summarization. This method identifies similarities and differences between the documents in the set. An improved technique for calculating sentence similarity with the aim of enhancing the performance of multi-document summarization was proposed by [97]. The prevalent approach to automated extractive summarization involves scoring phrases or sentences to generate summaries. Sentence scoring is widely embraced in the majority of contemporary methods. The scoring techniques are classified into word scoring, sentence scoring, and graph scoring [98]. In word scoring techniques, sentences are assigned scores based on the significance of words and their frequency in the text [2] ,[99], [100]. In particular, words such as proper nouns, places and objects, considered determinants, receive higher scores [101, 102]. Text scoring methods consider formal properties such as bold, italicized, and underlined words [103]. Sentences beginning with phrases like 'Briefly,' 'Finally,' and 'As a result' are identified as sign phrases and subsequent sentences are deemed important [99]. Similarly, sentences that contain title words are considered for inclusion in the summary by increasing their importance [104]. Sentence scoring methods also consider sentence length, giving more weight to longer sentences [103], [105]. Scoring involves assigning points based on the position of the sentence and whether it includes numerical values [99],[101],[106]. In

Reference [107], the authors outlined an approach to extractive summarization designed to assist learners facing reading difficulties. Graph-based representations are commonly employed in text analysis methodologies because of their highly effective solutions. Reference [108] introduced TextRank, which incorporates a graph-based representation to summarize text by identifying intersections in the content. Similarly, LexRank, presented in Reference [15], utilizes an algorithm based on eigenvector centrality, which is a form of the node centrality method. Both TextRank and LexRank draw inspiration from the PageRank algorithm [109], a document summarization framework that identifies central sentences based on mutual information between term and sentence sets [110].

Random Walk has been used to generate summaries of primary documents. In Reference [111], a summarization system targeting the biomedical domain was introduced. Using the Unified Medical Language system, a graph was derived from concepts and relationships using a semi-dictionary-based approach, followed by the application of the PageRank algorithm. In Reference [112], a novel graph based on reinforced random walk was suggested. In Reference [113], a multilayered representation was used that involved documents, sentences, and words. The authors in Reference [114] incorporated graphs to depict documents, employing link generation for automated document summarization. They established the document structure by revealing text relationships and evaluating summaries through comparison with manually created ones. Reference [115] introduced a graph-based approach to ensure semantic continuity, where nodes represented document terms, and edges reflected semantic relationships. The graph diameter calculation for all nodes described the shortest and longest paths as the weakest and strongest bonds. In Reference [116], graph structures were defined from documents, and nodes and edges were established based on local similarities.

1.5.3.2 Multidocument Summarization through abstraction

We have categorized the existing abstractive multidocument summarization methods into three groups:

- **Compression based approach**
- **Text generation based approach**
- **Neural Network-based and transformer based methods**

1.5.3.2.1 Compression based approach The main idea of compression-based approaches is to remove unimportant words from a sentence to compress it[117],[118]. Barzilay and McKeown (2005) [23] proposed a technique to effectively fuse multiple sentences, and the generated fused sentence contains salient information from multiple sentences. Common information is identified by developing an approach to align syntactic trees for the input with paraphrase information. Subtrees of input sentences were created and subsets of generated subtrees were matched. It was done using a bottom-up approach which performs multisequence alignment. A fusion lattice was constructed by combining fragments and enclosing the alignment along with lattice linearization. Filippova, strube (2008)[119]proposed an unsupervised method for sentence compression, which involved creating a dependency tree and removing subtrees. The approach consisted of three steps: tree transformation, tree compression, and tree linearization. The tree transformation step used grammatical properties, such as verbs and prepositions, to modify the tree. In the tree compression step, integer linear programming with a probability function was used to compress the tree. Finally, the tree was linearized to generate a compressed sentence that maintained the order of words in the source sentence.

The approaches mentioned above focus on creating abstractive summaries by removing unnecessary words only and do not involve the generation of new words to create an abstract. However, this limitation was overcome by employing a text generation approach, which allows the creation of new words and phrases to produce abstractive summaries.

1.5.3.2.2 Text generation based approach The approach proposed by Bing et al.(2015)[120] for abstractive summarization involves introducing new words and phrases to generate a summary. The approach uses a combination of phrase selection and merging to generate a summary sentence. The method employs integer linear optimization to select the most salient phrases from a pool of information presented by the phrases in the input corpus. First, noun and verb phrases are extracted, and a score is computed for each phrase to identify salient concepts. The important phrases are then selected from different source sentences and these phrases are merged to generate a new sentence. The algorithm guarantees that the newly generated sentence contains the maximum number of Summary Content Units (SCUs). Chenal and Cheung (2016) [121] performed an aggregation task to create a new output sentence. The approach involved clustering meaningful sentence fragments from the input sentences. The authors created a new dataset with different granularities from pre-existing sentences and conducted a user study to confirm the validity of the datasets. Two gold standard clusters were created and used as an evaluation method for the approach. Finally, a clustering model was created using logistic regression and hierarchical clustering to perform the task of aggregating sentence fragments into a new output sentence.

1.5.3.2.3 Neural Network-based and transformer-based methods The availability of large-scale data and advancements in neural network architectures have enabled the development of data-driven approaches to text summarization that are powered by end-to-end deep neural models. These approaches have replaced the earlier systems that were developed based on expert knowledge and heuristics. The model proposed by Chen and Bansal(2018) [122] is a two-stage approach to abstractive summarization, where the first stage involves extracting important sentences from the input document, and the second stage involves rewriting those sentences to generate a summary. The two stages are trained separately and then combined using a policy-based reinforcement learning algorithm, which learns to dynamically adjust the importance of the two stages based on the input document. The resulting model achieves state-of-the-art performance on several benchmark datasets for summarization. This model was named a fast abstractive summarization model, which aims to implement sentence rewriting.

The pointer-generator network proposed by Liu and Manning (2017) [123] is a hybrid model that combines the strengths of extractive and abstractive summarization methods. It uses pointer networks to copy words from the source text, allowing it to accurately reproduce information, and a generator network to generate new words, enabling it to create more concise summaries. In addition, a coverage mechanism is employed to ensure that the summary does not repeat information that has already been included. To incorporate sequence-to-sequence learning, Song et al. (2018)[124] proposed a transformer-based method to generate an abstractive summary. In this approach, the dependency parse tree structure is combined with a copy mechanism to include important words in the summary. A structure-infused copy mechanism is used to copy source words along with their relations to the summary based on their semantic and structural importance in the source sentences. GloVe embeddings were employed to convert input words into vectors.

Lewis et al. (2020) [125] proposed BART, a denoising autoencoder for pretraining sequence-to-sequence models. This model uses a two-step pertaining process, where random noise is added to the input text, and then a seq2seq model is trained to reconstruct the original text. The architecture used is a Transformer-based neural machine translation model, which can also be referred to as a bidirectional encoder or GPT (with a left-to-right decoder). This model has proven to be highly effective for text generation tasks as well as abstractive summarization tasks.

1.6 Evaluation

Assessing the quality of a summary has proven to be a challenging issue, primarily due to the absence of a clear-cut ’ideal’ summary. Even for relatively straightforward news articles, human summarizers tend to reach a consensus only around 60% of the time, as measured by the overlap in sentence content. Employing multiple models for system evaluation might mitigate this problem, but researchers also need to explore alternative methods that could yield more satisfactory models, potentially driven by a specific task.

Two broad categories of metrics have been developed: form metrics and content metrics. Form metrics focus on factors such as grammatical correctness, overall text coherence, and organization, typically rated on a point scale (Brandow, Mitze, and Rau 1995) [126]. The measurement of content, on the other hand, is more challenging. Typically, system output is compared sentence-by-sentence or fragment-by-fragment to one or more human-generated ideal abstracts. Similarly, in information retrieval, metrics such as precision (the percentage of extraneous information in the system’s summary) and recall (the percentage of important information omitted) are recorded. Other commonly used metrics include kappa (Carletta, 1996) [127] and relative utility (Radev, Jing, and Budzikowska, 2000) [128], both of which consider the performance of a summarizer that randomly selects passages from the original document to create an extract. In the Document Understanding Conference (DUC), 2001 and DUC 2002 summarization competitions (Harman and Marcu, 2001; Hahn and Harman, 2002) [129] [130], NIST employed the Summary Evaluation Environment (SEE) interface (Lin, 2001) [131] to capture precision and recall values. These competitions, modeled after TREC, have established baseline standards for single-document and multidocument summarization and have made available several hundred human abstracts for training (another popular source of training data being the Ziff-Davis corpus of computer product announcements). Despite limited consensus among human judges, DUC demonstrated that humans outperform machines in summary production.

The Summarization Evaluation Conference (SUMMAC) (Mani et al. 1998; Firmin and Chrzanowski 1999), [132] [133], which was one of the largest task-oriented evaluation conferences, included three tests: the categorization task (evaluating how well humans can categorize a summary compared to its full text), the ad hoc task (assessing how well humans can determine the relevance of a full text to a query based solely on the summary), and the question task (evaluating how well humans can answer questions about the primary content of the source text after reading only the summary). However, interpreting the results is not straightforward; studies (Jing et al. 1998 ; Donaway, Drummey, and Mather 2000) [134] [135] showed that the same summaries receive different scores under different measures or when compared to different (yet presumably equivalent) ideal summaries created by humans. Regarding inter-human agreement, Jing et al. find relatively high consistency in the news genre only when the summary (extract) length is kept relatively short. Marcu (1997a) [136] provides some evidence that other genres exhibit less consistency. As for the lengths of summaries produced by humans when not constrained by a specific compression rate, both Jing and Marcu observed considerable variation. However, it is now widely accepted that, for single news articles, automated systems produce generic summaries that are indistinguishable from those created by humans.

The quest for automated summary evaluation is a goal shared by many. Clearly, when an ideal extract has been created by humans, the evaluation of extractive summaries is straightforward. Marcu (1999) and Goldstein et al. (1999)[137] [138] independently developed an automated method for generating extracts corresponding to abstracts. However, when there is an insufficient number of available extracts, it remains unclear how to overcome the challenges of low inter-human agreement. Although using a variation of the Bilingual Evaluation Understudy (BLEU) scoring method (based on matching n-grams between the system output and the ideal summary) developed for machine translation (Papineni et al. 2001) [139] holds promise, it is not yet sufficient (Lin and Hovy 2002b) [140]. The BLEU (Bilingual Evaluation Understudy) metric is commonly employed to evaluate the accuracy of machine-generated text, particularly in machine translation (MT) contexts. Its purpose is to gauge the alignment between machine-generated text and human-generated reference translations. BLEU is extensively utilized in natural language processing (NLP) and machine translation applications. BLEU assesses the likeness between the machine-generated text and one or more reference translations by comparing n-grams (sequences of n words) within the generated text to those in the reference(s). BLUE score can be computed as a product of the brevity penalty (BP) and the geometric average precision using equation 1.1.

$$BLUEscore = BrevityPenalty(BP) * Geometric\ Average\ Precision \tag{1.1}$$

Precision can be defined as fraction of common words found among Target Sentence(TS) and Reference sentence(RS). For example:
Predicted Sentence(PS): ”Weather update predicts rain on Monday.”
Target Sentence(TS): ”Weather update predicts rain with thunderstorm”.

$$1gramPrecision(p_1) = \frac{common\ 1grams\ among\ RS\ and\ TS}{Length\ of\ TS} \tag{1.2}$$

In equation 1.2 common words among the predicted sentence and the target sentence are ”weather”,” update”, ”predicts”, ”rain”. The length of the target sentence is 6. Therefore, 1 gram precision = 0.6

$$2gramPrecision(p_2) = \frac{common\ 2grams\ among\ RS\ and\ TS}{Length\ of\ TS} \tag{1.3}$$

In Equation 1.3 the common 2grams among the predicted sentence and the target sentence are "weather update"," predicts rain", "update predicts". The length of the target sentence is 6. Therefore, 2 gram precision = 0.5

n-gram precision(p_n) can be calculated using equation 1.4.

$$ngramPrecision(p_n) = \frac{common\ ngrams\ among\ RS\ and\ TS}{Length\ of\ TS} \tag{1.4}$$

Now Geometric Average Precision can be calculated using equation1.5

$$\begin{aligned} Geometric\ Average\ Precision(N) &= \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \\ &= (p_1)^{(1/4)}.(p_2)^{(1/4)}.(p_3)^{(1/4)}.(p_4)^{(1/4)} \end{aligned} \tag{1.5}$$

In baseline N = 4 and uniform weights $w_n = 1/N$

In case of a 1-length sentence, the precision score is 1 which is misleading. To overcome this, Brevity Penalty(BP) is used as it penalizes the short sentences. The Brevity Penalty (BP) can be calculated using equation 1.6.

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-\frac{r}{c})} & \text{if } c \leq r \end{cases} \tag{1.6}$$

In Equation 1.6, c is defined as the length of the predicted sentence and r is defined as the length of the target sentence.

For automatic summary evaluation, there is another widely used automatic summary evaluation package called ROUGE [141], which measures the quality of the summary based on n-gram overlap between a summary generated by a system and a set of available reference summaries [40]. In addition to computing the word n-gram overlap, it also counts various kinds of other overlapping units between the system summary and the reference summaries. The latest version of the ROUGE package is ROUGE 1.5.5. The ROUGE toolkit reports the ROUGE-N score where N can be set to 1, 2, 3, and 4. The ROUGE toolkit is also used to measure skip Bigram-based ROUGE scores. Among various types of ROUGE scores, the unigram-based ROUGE score (ROUGE-1) is more consistent with human judgment [40].

Along with ROUGE-1 scores, many state-of-the-art summarization systems have been evaluated using ROUGE-2 (bigram-based) and ROUGE-SU4 (skip bigrams with skip distance up to 4 words [141]). To evaluate the summarization models proposed in this thesis , we have used ROUGE 1.5.5 which reports precision, recall, and F-measure. Some ROUGE recall metrics are defined below.

- **ROUGE-N** :

  ROUGE-N quantifies the degree of n-gram overlap, where n-grams are sequences of n words, between the machine-generated text and the reference text. The ROUGE-N formula can be expressed as:

$$ROUGE - N = \frac{\sum_{S \in ReferenceSummaries} \sum_{gram_n \in N} Count_{match}(gram_n)}{\sum_{S \in ReferenceSummaries} \sum_{gram_n \in N} Count(gram_n)} \tag{1.7}$$

  Where: n stands for the length of the n-gram, $gram_n$, and $Count_{match}(gram_n)$ is the maximum number of n-grams co-occurring in a candidate summary and a set of reference summaries.

  For example, when calculating the ROUGE-1 recall (unigram overlap), the individual words (unigrams) matched between the generated text and the reference are counted and then divided by the total number of words in the reference.

- **ROUGE-L**:

  ROUGE-L assesses the longest shared subsequence in both the machine-generated text and the reference text. The ROUGE-L formula is given by:

$$ROUGE - L = \frac{\sum_{S \in \text{ReferenceSummaries}} \text{length}(S)}{\sum_{S \in \text{ReferenceSummaries}} \text{longest\_common\_subsequence}(S, C)} \tag{1.8}$$

  Where: ReferenceSummaries denotes the collection of reference summaries. S stands for each specific reference summary within this collection. C denotes the summary generated by the summarization system. The function length(S) computes the size of the reference summary S.
  The function $longest\_common\_subsequence$(S, C) determines the length of the longest common subsequence between the reference summary S and the candidate summary C. This metric underscores the importance of preserving word order and maintaining sentence structure in the evaluation process.

- **ROUGE-W (Weighted Precision)**:

  ROUGE-W computes a weighted precision score by assigning varying weights to different n-grams on the basis of their length. The ROUGE-W formula is expressed as follows:

$$ROUGE\text{-}W = \frac{\sum_{S \in \text{ReferenceSummaries}} \text{length}(S)}{\sum_{S \in \text{ReferenceSummaries}} \text{weighted\_longest\_common\_subsequence}(S, C)} \tag{1.9}$$

Where: ReferenceSummaries denotes the collection of reference summaries. S denotes each specific reference summary within this collection. C denotes the summary generated by the summarization system. The function length(S) computes the size of the reference summary S.
The function $weighted\_longest\_common\_subsequence$(S, C) determines the length of the weighted longest common subsequence between the reference summary S and the candidate summary C.

- **ROUGE-S (Skip-bigram overlap)**:

  ROUGE-S assesses skip-bigram overlap, allowing gaps between words during the matching procedure. The formula closely resembles that of ROUGE-N, but accommodates word skips:

$$ROUGE\text{-}S = \frac{Number\ of\ matching\ skip - bigrams\ between\ system\ and\ reference}{Total\ number\ of\ skip - bigrams\ in\ reference} \tag{1.10}$$

  Skip-bigrams denote word pairs with a predefined maximum gap size between them.

1.6.1 Example of computing ROUGE-1

Reference sentence(RS): "The ball is blue."
Machine-Generated summary (MGS): 'The new blue ball is very beautiful'

$$ROUGE - 1Recall(R) = \frac{common\ words\ among\ RS\ and\ MGS}{Length\ of\ RS} \tag{1.11}$$

In equation1.11 common words among the reference sentence and the machine-generated summary are "The"," ball", "is", "blue". The length of the reference sentence is 4. Therefore, ROUGE-1 recall (R) = 1.

$$Rouge1Precision(P) = \frac{common\ words\ among\ RS\ and\ MGS}{Length\ of\ MGS} \tag{1.12}$$

In equation1.12 common words among the reference sentence and the machine-generated summary are "The"," ball", "is", "blue". The length of the machine-generated summary is 7. Therefore, ROUGE-1 precision (P) = 0.57

$$Rouge1F - Score = \frac{2 * P * R}{P + R} \tag{1.13}$$

Putting the values of P and R in the equation1.13, we get the value of the ROUGE-1 F score, which is 0.72 for the given example.

1.7 Scope of the Thesis

In this thesis, we investigate the problem of single and multi-document text summarization. A summary of the scope of the thesis:

- Develop novel techniques to solve single-document and multi-document extractive text summarization.
- Develop novel techniques to solve multi-document abstractive text summarization.
- Explore enhancing the traditional extractive single document extractive text summarization methods using word embedding and spectral clustering of sentences
- Explore hybrid sentence similarity measure combining lexical similarity and semantic similarity for improving extractive multi-document text summarization.
- Explore text-to-text generative deep learning model for sentence fusion and generating abstract summary from multiple documents by selecting fused sentences using the K-L divergence criterion.
- Evaluation of the proposed summarization models using benchmark datasets and a Bengali summarization dataset developed by us.

1.8 Motivation and Contribution of the Thesis

In today's digital era, in response to a single query, information retrieval systems, such as a search engine, the search interface to a domain-specific digital library, return a large number of documents. To meet information needs, users have to spend a lot of time and effort identifying and consuming a vast amount of online documents, including news articles, scientific articles, social media texts, etc. Having manually a crux with relevant information from a long document or a document cluster is a tedious task. During accessing the web using hand-held devices, bandwidth becomes a crucial issue. Instead of transferring the whole document, a gist of the document is transferred first to the hand-held devices, and the users' needs are satisfied with the gist, and bandwidth is saved. Therefore, we need an automatic text summary system to produce accurate and relevant summaries to manage the information overload problem and save bandwidth in some applications. Text summarization helps to create a condensed version of a document by selecting sentences with pertinent information from the document(s). Text summarization can be broadly categorized into two types: extractive and abstractive. An extractive summarization system aims to generate a summary by selecting textual segments or sentences from the document(s). On the other hand, an abstract summarization system produces a summary from a document(s) by introducing new words or phrases which may not be present in the original document. Single-document summarization is a process of creating a summary from only one document, and multi-document summarization is a process to create a summary from a cluster of documents. Both single-document and multi-document summaries can be extractive or abstractive. By analyzing existing research work on text summarization, it was observed that many researchers focused on developing a better extractive summarization system. Several research works focused on developing an abstractive summarization system. Generating an abstract is relatively tougher than generating an extract, because abstract generation needs a deeper analysis of the input.

Although research on text summarization was started in the long past, adequate success was not achieved because text summarization is a tough task. Most of the earlier studies on text summarization focused on using surface-level word and sentence features to score sentences and rank them. The earlier summarization system was far from generating a quality summary. Therefore, there is scope for improvement in extractive text summarization. In recent times, an effective Deep Learning technique known as transfer learning has transformed and revolutionized the field of natural language processing. Several transfer learning techniques and architectures have been developed to advance the state of the art on a variety of NLP tasks. Pre-trained word embedding have been more popular due to the widespread availability and simplicity of integration with a new task. The use of pre-trained word embedding for word and document representation can be effective for capturing semantic matching between words or sentences that can help in improving both extractive and abstractive text summarization performance.

Although abstractive summarization is tougher than extractive summarization, the advent of the text-to-text generative model has given us a new way to deal with abstractive text summarization problems. This gives us additional motivation to design and develop a system that can produce extractive and abstractive summary from a document or a set of documents. The current thesis focuses on devising new extractive and abstractive text summarization techniques that can produce summaries from single document or multiple documents to help users in managing information overload problems and saving bandwidth for handheld devices. We explore the unsupervised techniques since it does not require a large amount of training data. We also explore the supervised techniques that use pre-trained word embedding and text-to-text generative models. The main contributions of the thesis are summarized as follows:

- In Chapter 2, we propose three unsupervised methods for extractive text summarization-(1) a sentence ranking-based summarization method that combines saliency score and novelty score for sentence scoring (2) a latent semantic indexing-based summarization method that uses singular value decomposition (SVD) of a term-by-sentence matrix created from a document to map each sentence to a latent semantic space and calculates the score of the sentence by the magnitude of the sentence vector after multiplying its components by the corresponding singular values, and (3) a spectral clustering-based summarization method that uses word embedding-based sentence representation and a spectral clustering algorithm to identify various topics covered in a document by producing topical clusters of sentences and generate an extractive summary by selecting salient sentences from the identified topics. The spectral clustering method is useful in text summarization because it does not assume any fixed shape of the clusters, and the number of clusters can automatically be inferred using the eigengap method. We have used word embedding-based sentence representation and a spectral clustering algorithm to identify various topics covered in a Bengali document and generate an extractive summary by selecting salient sentences from the identified topics. We have also compared the performance of the summarization methods mentioned above. We evaluated the sentence-ranking-based summarization method that combines the saliency score and the novelty score using the benchmark summarization dataset named DUC2002 dataset. We also evaluate the proposed LSI-based method and the proposed spectral clustering-based method on the Bengali summarization dataset developed by us.

- In Chapter 3, We propose a supervised neural model that learns from the data how to balance the saliency and the diversity in the summary during extractive summary generation by selecting summary-worthy sentences in a summary. A set of hand-crafted features is designed and those features are used for calculating the saliency of a sentence. To model diversity in the summary a set of diversity features is defined based on the semantic relations between the sentence to be selected in the summary and other sentences previously selected in the summary. Using the set of saliency features and a set of diversity features, a neural model is trained in supervised mode. Given a sentence of a document and the previously selected summary sentences(throughout this chapter, the group of previously selected sentences is referred to by Summary-so-far) as input, the proposed learning model decides whether the given sentence is worth including in the summary or not. The model is trained in such a way that it favors the sentences that are salient and whose inclusion improves the summary diversity. Initially, the Summary-so-far is set to empty. During testing, the sentences from the input document are individually submitted in text order to the trained model. If the model finds that a sentence is suitable for inclusion in the summary, it is added. In this framework, the model is invoked repeatedly until the summary of the desired length is generated. We evaluated the performance of the proposed model on open benchmark datasets, and its performance is compared with many state-of-the-art summarization systems.

- In Chapter 4, We explore an approach to extractive multi-document text summarization. Since sentence similarity measure plays an important role in graph-based multi-document text summarization, we define a hybrid similarity measure that combines a lexical similarity score and a semantic similarity score. We hypothesize that enhancing the similarity measure through hybridization can improve the performance of graph-based summarization. To compute lexical similarity, we represent sentences using the bag-of-words model and TF*IDF-based vector representation. Finally, lexical similarity is measured by the cosine of the obtained vectors. On the other hand, to compute the semantic similarity between two sentences, we first pass sentences to BERT (Bidirectional Encoder Representations from Transformers), a transformer-based language model, which processes each sentence as a whole and produces a sentence vector. Then we compute the cosine of the sentence vectors obtained from BERT. This similarity is considered semantic similarity because BERT produces dense sentence vectors by capturing the semantics of the sentences.

- Human summarizes document(s) by identifying and representing the main themes in a form called an abstract. While abstracting a document(s), human applies some cognition procedure for reformulating the main concepts of the document(s). Since abstracting is a human ability, it is tough for a machine to generate a human-like summary from a document or document set. However, due to the progress in the deep learning-based generative model, abstractive summarization has recently received attention of many researchers. In chapter 5, we propose an abstractive multi-document summarization approach that uses an unsupervised extractive summarization model to select a small number of significant sentences from the input and then passes the extracted sentences to a deep learning-based sentence fusion model that produces a number of fused summary sentences. Finally, a subset of the fused sentences is selected using a criterion that maximizes the information content of the final summary. KL divergence (Kullback-Leibler divergence)-based method is used to select some informative nonredundant sentences among a list of fused sentences to produce the final summary. In this summarization method, a fused sentence is selected in a summary if the term distribution in the summary after adding this fused sentence mostly agrees with the term distribution in the source documents. This is based on the hypothesis that the best summary is a subset of sentences that has the closest term distribution to that in the input documents. A transformer-based text-to-text sentence fusion model is trained on a large sentence fusion dataset generated automatically from the multinews summarization dataset.

1.9 Organization of the Thesis

The structure of our thesis is detailed as follows.
In Chapter 2, we propose some unsupervised extractive text summarization algorithms to improve the extractive text summarization performance.

In Chapter 3, we present supervised extractive text summarization using a neural network. We have used a neural network-based learning model that learns to include a sentence in a summary by taking into account both the saliency of the sentence and the diversity of the summary.

In Chapter 4, we focus on extractive multidocument text summarization using a hybrid sentence similarity measure combining a lexical similarity measure and a BERT-based semantic similarity measure.

In Chapter 5, we describe abstractive text summarization. We have performed multi-document abstractive text summarization that uses an unsupervised summarization method for extractive summary generation and a deep learning-based sentence fusion model to fuse related sentence pairs for an abstract generation.

In Chapter 6, we examine the contributions of our research and discuss possible future research avenues to wrap up the thesis.

2 Unsupervised Single Document Extractive Text Summarization Methods

2.1 Introduction

Unsupervised approaches for text summarization do not require labeled training data and aim to identify the most relevant sentences in a document based on their content and structure. These techniques rely on statistical and natural language processing methods to analyze the document and identify important content. This approach can be useful when working with large volumes of unstructured data or when labeled data are not readily available. The unsupervised extractive text summarization approach involves several sentence-ranking-based methods and a sentence clustering-based method [142]. The sentence ranking approach ranks sentences based on sentence scores where a score of a sentence is computed using sentence content-based features such as word importance , cue phrases, etc. and document structure-based features such as sentence position, relation of a sentence to other sentences of the document. When relation of a sentence with other sentences is computed, sentences are represented either in a vector space or a semantic space. For mapping sentences to vector space, the bag-of-words model and TFIDF term weight are used, and for mapping sentences to a semantic space, latent semantic indexing is used. The cluster-based approach groups similar sentences into multiple groups, called clusters, and selects representative sentences from each cluster to form a summary. In this case, sentences can be represented using the bag-of-words model or latent semantic indexing model.

In this chapter, we propose three unsupervised methods for extractive text summarization-
(1) A sentence ranking-based summarization method that combines saliency score and novelty score for sentence scoring. (2) A latent semantic indexing-based summarization method that uses singular value decomposition (SVD) of a term-by-sentence matrix created from a document to map each sentence to a latent semantic space and calculates the score of the sentence by the magnitude of the

sentence vector after multiplying its components by the corresponding singular values. (3) A spectral clustering-based summarization method that uses word embedding-based sentence representation and a spectral clustering algorithm to identify various topics covered in a document and generate an extractive summary by selecting salient sentences from the identified topics. (4) We also compare the performance of these methods and also present the comparison results in this chapter.

The first part of this chapter discusses methods for unsupervised extractive text summarization. The second part of this chapter describes our own dataset on which the proposed models are tested. In the third part of this chapter, we present experimental results and parameter tuning. Finally, the last part of this chapter concludes with a discussion of the effectiveness of our approaches, the limitations of the proposed methods, and future scope.

2.2 Proposed Unsupervised Extractive Text Summarization Methods

In this section, we discuss in detail three single-document text summarization methods. First, we discuss a sentence ranking-based summarization method that combines the saliency score and novelty score for sentence scoring. Second, we discuss a latent semantic indexing-based summarization method and finally we discuss a spectral clustering-based summarization method. These methods are discussed in the following sub-sections.

2.2.1 Text Summarization Combining Saliency Score and Novelty score

Saliency based sentence ranking is the basic step of extractive text summarization. The saliency of a sentence is often measured on the basis of the important words that the sentence contains. One of the drawbacks of this sentence extraction method is that it mainly extracts sentences related to the most common topic in the document. But the input document may contain multiple topics or events and the users may like to see in the summary the salient information for each different topic or event. To alleviate such a problem, the diversity-based reranking method or the sentence clustering-based method is commonly used. The popular diversity-based sentence reranking [143] which is often used to allow diverse information in the summary by removing redundant information is the MMR-based reranking (maximal marginal relevance). In this method, after ranking the sentences of a document based on scores, when the summary is generated, the top-ranked sentence is selected first and the next sentence is selected if it is sufficiently dissimilar to the previously selected sentences. This process is repeated until the desired summary length is reached. Since the order of ranks of the sentences in the summary is further revised at the time of summary generation, this process is generally known as reranking. The main problem with the reranking-based method is that it hampers the speed of the summarization process. To alleviate this problem, we propose a novel summarization method that computes the score of a sentence by combining the saliency and novelty of the sentence. Without using any re-ranker, the proposed method can automatically take care of the diversity issue while producing a summary.

We have developed a sentence ranking method that assigns a score to each sentence of the document based on two important factors: (1) saliency, which refers to how much salient information contained in the sentence and (2) novelty, which refers to how much novel information contained in the sentence. Although our proposed method does not use reranking, it can automatically deal with the redundancy as well as diversity issue by setting up a trade-off between saliency and novelty of a sentence while scoring the sentence.

The idea of the saliency is implemented using the TF-TDF based term weighting method[7] [6] that assigns a score to a sentence based on the number of important words contained in the sentence where an important word is identified by the TF-IDF weight of the word. However, novelty is measured using how many novel words are contained in the sentence. We define a word as novel to a sentence if it appears first in the sentence, that is, it is the word that was not seen before in the sentences previously encountered in the document. Since we observe that some sentences contain more novel words than other sentences, we distinguish among the novel words present in a sentence based on some features such as how many times the novel word is repeated after its first occurrence and whether it is a part of proper name or not. According to our assumption, a word that is novel to a sentence X cannot be novel to other sentences appearing in the document after X. We observe that, when two sentences have equal or near equal saliency scores, our proposed sentence scoring method gives higher preference to the sentence containing more novel information. The rationale behind such preference over sentences is that a sentence containing a more number of novel terms (words) appears most likely at the junction of ending old topic and starting new topic, and a term remains novel for the moment it enters first into the discourse. We assume that the important novel words chosen by the writer of the document while transiting from the old topic to a new topic play an important role in extractive summarization. So, by capturing salient and novel information, our proposed summarization approach selects sentences containing important and novel information while creating summary.

2.2.1.1 Preprocessing:

Before computing saliency and novelty score of each sentence, the stop words (a set of highly frequent but less important words and a set of common verbs) are removed from the sentence and then it is stemmed using Porter Algorithm [144].

2.2.1.2 Sentence Scoring Based on Saliency:

TF-IDF based score for a sentence S is:

$$\text{TF-IDF based score (S)} = \sum_{w \in S} \text{TF-IDF}(w) \tag{2.1}$$

Where TF-IDF(w) is the product of term frequency of the word w and Inverse Document frequency(IDF) of the word w. IDF(w) is computed as $\log\left(\frac{N}{\mathrm{df}(w)}\right)$, where N is corpus size and df(w) is called document frequency of w, that is, how many documents of the corpus contain the word w at least once. To prevent longer sentences from getting the higher TF-IDF based score, the words with TF-IDF value less than a predefined threshold value are considered as low content words and so they are removed [7] from each sentence.

2.2.1.3 Sentence Scoring Based on Novelty

The novelty of a sentence S is measured based on how many novel terms are contained in S:

$$\text{Novelty Score (S)} = \sum_{w \in S} \text{Weight}(w) \tag{2.2}$$

Where:

$$\text{Weight}(w) = \begin{cases} \text{TF-IDF}(w) & \text{if } w \text{ is novel to } S \text{ \& } w \text{ occurs at least twice in document} \\ \partial \cdot \text{TF-IDF}(w) & \text{if } w \text{ is novel to } S \text{ and } w \text{ is part of a proper noun} \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

We call a word novel for a sentence if it occurs first in the sentence, that is, the word was not occurring in the other sentences of the document before occurring in the current sentence. A novel term is assigned an additional weight proportional to the TF-IDF weight of the word. The novel word that occurs only once in the document is assigned a weight of 0 because longer sentences may contain many unimportant novel terms. The rationale behind using such formula given in Equation 2.3 used to score novel words is to discriminate between

the most important novel words and the least important novel words and to maintain uniformity of the novelty score with the scale on which the saliency score is calculated. This proper scaling is necessary because we finally combine the saliency and novelty of a sentence to have a unique score for the sentence. In equation 2.3,δ is considered as a boosting factor and δ is set to a value greater than 1 to assign a higher weight to the novel word. To identify whether a word is part of a proper noun or not, we have used the BNLP (Bengali Natural Language Processing) toolkit for POS tagger. This tagger is used to identify whether a word is a part of a proper noun or not.
In equation 2.4 δ, is considered a boasting factor and is set at a value greater than 1 to assign a higher weight to the novel word which is a part of the proper name. We observe that setting the value of δ = 2is good enough for our experiments.

2.2.1.4 Combining Saliency Score and Novelty Score

Overall score of a sentence S is a weighted combination of the saliency and novelty score.

$$\text{Overall Score (S)} = \text{Saliency Score}(S) + \lambda \cdot \text{Novelty Score}(S) \tag{2.4}$$

Where: λ is the weight assigned to the novelty score. In equation 2.4, there is the only one tuning parameter that is tuned through experimentation to obtain the best summarization performance.

2.2.1.5 Summary Generation

After ranking the sentences of the input document based on the combined score of saliency and novelty, the top-ranked sentences are selected from the ranked list one by one until the given summary length is reached. We have not used the MMR-based diversity-based re-ranking [143] or its variants while generating summary. The reason is that our proposed summarization method automatically takes care of the diversity issue by exploiting novelty score used in sentence ranking.

2.2.2 Text Summarization Using Latent Semantic Analysis:

The traditional sentence scoring method based on term weighting [2][3][4][5] and sentence salience-based method presented earlier in this chapter do not take into account the semantics of the input text. They assign a score to a sentence based primarily on the frequency of the term and other features. For example, a ″Bank″ can be a financial institution or a river bank. The frequency-based method counts both occurrences when the frequency of the word ″bank″ is counted. The different meanings of the word ″Bank″ can only be understood by considering their contexts. Latent Semantic Analysis (LSA) is used to solve this problem. LSA can help uncover the hidden structure and term associations that exist within the text. LSA can capture semantic relations among words in a text by mapping terms and sentences in a semantic space.

We propose a sentence scoring method that assigns a score to a sentence based on the importance of semantic concepts shared by the sentence. For deriving the latent semantic concepts in a document and finding the embedding of each sentence in the latent semantic concept space, we have used the latent semantic analysis(LSA) technique that applies singular value decomposition (SVD) on a term-by-sentence matrix created from a document to map each sentence to a latent semantic space and calculates the score of the sentence by the magnitude of the sentence vector after multiplying its components by the corresponding singular values.

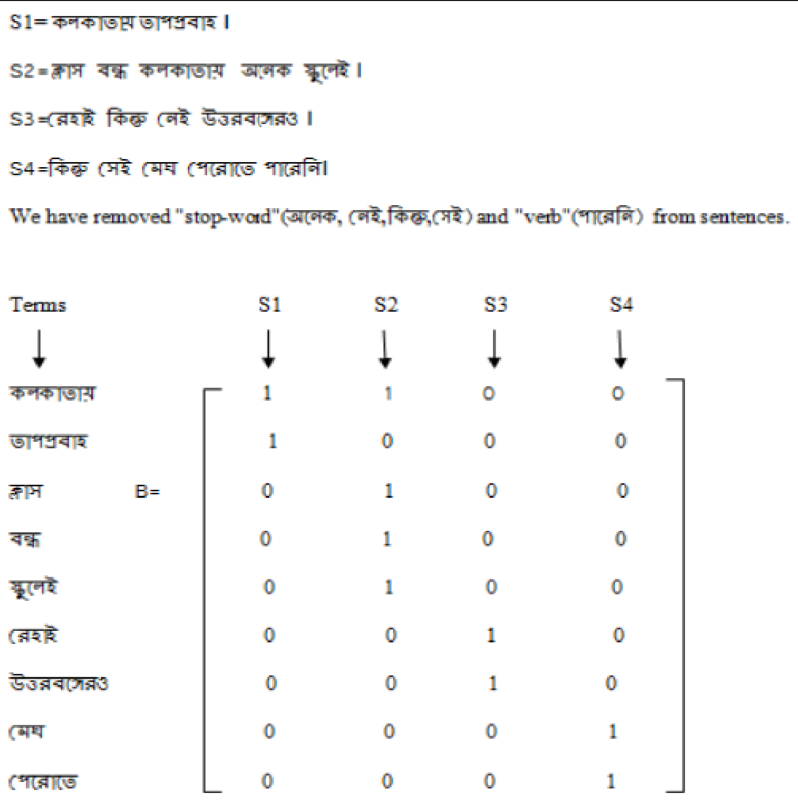
We have developed our method for the Bengali language. The efficiency of our approach lies in creating a sentence score calculation. We have experimented with varying dimensions of the sentence vectors to find the best LSI based summarization model. We observed that the dimension of the sentence vectors affects the summarization results. Our LSI based summarization model gives better results than the baseline model.

In our approach, first, a document representation is developed in two steps. The first step is the creation of a term-by-sentence matrix, $B = [B_1, B_2, ..., B_d]$, where each column B_i represents the vector of term weights for the i^{th} sentence in the document under consideration. Fig.2.1 shows a sample term-sentence matrix of order $n \times d$, where n is the number of distinct terms in a document and d is the number of sentences in the document. We create a term-by-sentence matrix for a document with the n number of terms obtained after removing stop words and verbs from the document. If the document has n terms and d sentences, the term-by-sentence matrix B will be of order $n \times d$. Each entry b_{ij} in matrix B is filled with the term frequency (TF) of the i-th term (the term by which row i of matrix B is indexed) in sentence j. The weight based on TF of a term is calculated as the number of times the term occurs in the document [145].

After creating the term-by-sentence matrix, the next step is to apply SVD (Singular Value Decomposition). When SVD of matrix B is performed, this results in three matrices M, N, and P shown in equation2.5.

$$B_{n\times d} = M_{n\times n} \cdot N_{n\times d} \cdot P^{\top}_{d\times d} \tag{2.5}$$

From the point of view of NLP (Natural Language Processing), M is a term-by-concept matrix of size n × n. N is a concept-by-concept matrix (rectangular diagonal) of size $n \times d$ with positive real numbers in descending order on the diagonal. P^T is a concept-by-sentence matrix of size $d \times d$. When the dimensionality



S1= কলকাতায় তাপপ্রবাহ ।

S2= ক্লাস বন্ধ কলকাতায় অনেক স্কুলেই ।

S3= রেহাই কিন্তু নেই উত্তরবঙ্গেরও ।

S4= কিন্তু সেই মেঘ পেরোতে পারেনি।

We have removed "stop-word"(অনেক, নেই,কিন্তু,সেই) and "verb"(পারেনি) from sentences.

$$B = \begin{array}{c|cccc}
\text{Terms} & S1 & S2 & S3 & S4 \\
\hline
\text{কলকাতায়} & 1 & 1 & 0 & 0 \\
\text{তাপপ্রবাহ} & 1 & 0 & 0 & 0 \\
\text{ক্লাস} & 0 & 1 & 0 & 0 \\
\text{বন্ধ} & 0 & 1 & 0 & 0 \\
\text{স্কুলেই} & 0 & 1 & 0 & 0 \\
\text{রেহাই} & 0 & 0 & 1 & 0 \\
\text{উত্তরবঙ্গেরও} & 0 & 0 & 1 & 0 \\
\text{মেঘ} & 0 & 0 & 0 & 1 \\
\text{পেরোতে} & 0 & 0 & 0 & 1
\end{array}$$

Figure 2.1: Sample term by sentence matrix

of the M, N and P^T matrices is reduced to the k most important dimensions, we can obtain three matrices: an n × k matrix M′, a k × k matrix N′, and a k × d matrix P'^T. We demonstrate this in Fig.2.2.

Since SVD maps n dimensional vectors (where a vector corresponds to a sentence) to k dimensional space by breaking down the original matrix B into k independent base vectors expressing k different concepts or topics in the document, SVD can capture the semantic relationships between terms so that terms and sentences can be clustered based on semantic basis. A left singular vector that corresponds to a column of the matrix M represents a pattern of combination of words that repeats in the document, and one of the left singular vectors represents the most significant pattern. As each particular word combination pattern describes a certain concept in the document, the facts described above naturally lead to the hypothesis that each singular vector represents a salient concept of the document, and the magnitude of its corresponding singular value present in the matrix N represents the degree of importance of the salient concept. Similarly, each row of the matrix P^T represents a salient concept and an entry p_{ij} in the

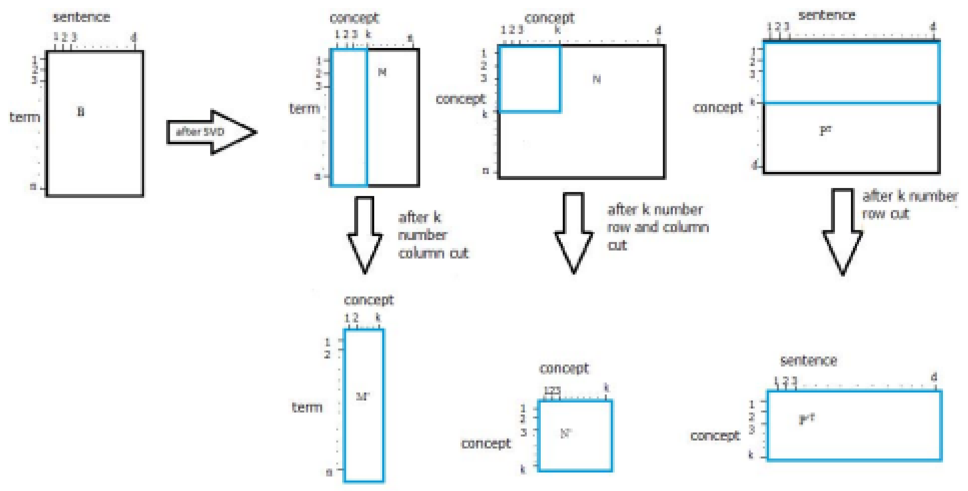


Figure 2.2: SVD of term-by-sentence matrix

matrix P^T represents the degree of similarity of sentence j to the salient concept i.

After creating the term-by-sentence matrix and performing SVD of the matrix, the next step is to select important sentences to create a summary for the document. For creating summaries with varying compression ratio, the sentences should be ranked. The method we used here to extract the sentence for summary is described in Algorithm 1.

Algorithm 1 LSA based Sentence Extraction and summary generation

```
Input: Matrix created by SVD for a document
Output: Summary with user specified compression ratio(r).
1: Specify the value of k that indicates the number of most significant dimensions
   chosen by the user. Reduce the matrix N and P to obtain the matrices N′with
   order of k x k and $P^{'T}$ with order of k x d.
2: For each sentence vector (column vector) in $P^{'T}$ (column vector in $P^{'T}$ becomes
   a row vector in $P'$), multiply each component of the vector by the corre-
   sponding singular values. The reason for such multiplication is to give higher
   weights to the index values in the matrix $P^{'T}$ that correspond to the highest
   singular values (the most significant concepts). For each sentence vector in the
   resultant matrix, compute the score of the sentence by computing the length
   of each sentence vector in the k-dimensional space. More formally,

   $$Score_m = \sqrt{\sum_{j=1}^{k}\left((P'_{jm})^2 \cdot (N_j)^2\right)} \quad (2.6)$$

   Where: $Score_m$ is basically the length of the $m^{th}$ sentence vector in the
   k-dimensional concept space, $N_j$ is the $j^{th}$ singular value and $P'_{jm}$ is the value
   in the row $j$ and column $m$ in the matrix $P^{'T}$ ($P'_{jm}$ is basically the similarity of
   the sentence $m$ to the $j^{th}$ salient concept).
3: Sentences are ranked based on scores obtained using equation 2.6.
4: Top ranked r% sentences are selected in the summary and ordered according
   to their appearance in the text. Here, r is specified by the user.
```

2.2.3 Text Summrization Using Sentence Embedding and Spectral Clustering

Single document extractive text summarization produces a condensed version of a document by extracting key sentences from the document. The most significant and diverse information can be obtained from a document by breaking it into topical clusters of sentences. Cluster-based text summarization is an approach that involves grouping of similar sentences in a document and selecting a representative sentence from each cluster to form the summary, without the need for labeled training data. The idea of employing a clustering algorithm for text summarization was well described in [146]. This approach used three steps for text summarization: histogram-based clustering algorithm for sentence clustering, ordering of clusters, and extraction of summary-worthy sentences from the clusters to create the summary. In [43], a hierarchical agglomerative clustering algorithm was used to create clusters of sentences. To create a summary, the sentences were chosen in order from the largest to the smallest cluster. Another clustering-based approach presented in [44] incorporates cluster-level information in a graph model for ranking sentences. However, early work (Sarkar, 2009a) [146] suggested that the performance of clustering-based text summarization is highly dependent on the quality of the clusters produced.

Clustering algorithms perform well when we have a clear idea of the attributes of data points [45]. Compactness-based clustering highlights spatial proximity among data points. For example, agglomerative average link clustering [9], k-means [46], highlights compactness. The resultant clusters using this algorithm are spherical clusters. Modification of k-means was discussed in [47] which is defined as k-means++. Though it uses a better centroid initialization technique for improvements over k-means, still it suffers from some drawbacks because we need to specify the number of clusters to be formed in advance and it assumes a fixed shape of clusters. After investigating different existing clustering algorithms, we can find that spectral clustering is more suitable for the text summarization task because it embeds sentences in a low-dimensional eigenspace and performs clustering on the data points mapped to the low-dimensional embedding space. It does not assume any fixed shape of the cluster, but rather emphasizes graph partitioning [48] based on connectedness among the vertices representing the data points. So, it is very useful when the shape of cluster is non-convex. Today, the spectral clustering algorithm has been used in a wide range of application areas such as image clustering [49], shape clustering [50], motion clustering [51] and many more. Gupta et al. [52] presented a text summarization approach based on spectral clustering, which uses textual inclusion (TE) and spectral clustering (ATESC) to calculate sentence connectedness scores. It is used to measure the saliency of a sentence in the input.

The method we have proposed differs from the existing methods. We use the spectral clustering algorithm to segment a document into multiple topical clusters and create a summary by choosing the most relevant sentences topic by topic. The spectral clustering method is useful in text summarization because it does not assume any fixed shape of the clusters, and the number of clusters can automatically be inferred using the eigengap method. On the other hand, the existing approaches, decompose the entire document into a collection of sentences and rank the sentences based on some features to create a summary. So, our proposed approach uses an effective clustering-based method that produces a summary covering all important topics in a document. However, to our knowledge, our approach is a new attempt to use a spectral clustering algorithm in the Bengali text summarization domain.

The steps of our proposed system are illustrated in Figure 2.3. Each step of the proposed system is discussed in this section.

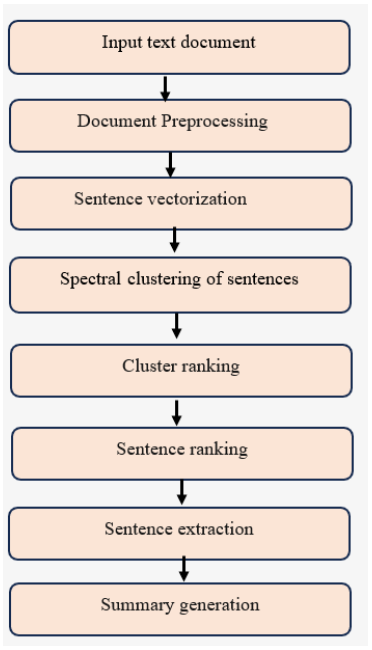
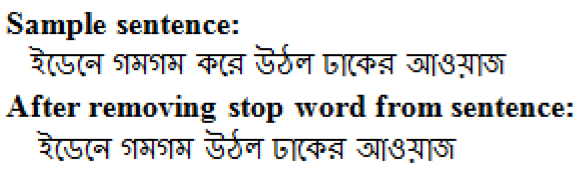


Figure 2.3: Steps of the proposed summarization system

2.2.3.1 Preprocessing

Sentences are identified using a sentence tokenizer available with the NLTK toolkit. A sentence is split into words. The stop words are discarded from the sentences. Stop words denotes unimportant frequent words in the data set. A predefined human-made list of Bengali words [1] was considered for the removal of stop words. 363 stop words were considered in that stop-word list. A sample sentence after discarding the stop words from it is represented in Figure 2.4.



Sample sentence:
ইডেনে গমগম করে উঠল ঢাকের আওয়াজ
After removing stop word from sentence:
ইডেনে গমগম উঠল ঢাকের আওয়াজ

Figure 2.4: Removal of stop word for Bengali sentence

2.2.3.2 Sentence Vectorization

After preprocessing, sentences are passed to the vectorization step. The vector for a sentence is obtained by taking an average of the vectors corresponding to the

[1]http://fire.irsi.res.in/fire/static/resources

words that appeared in the sentence. FastText word embeddings [2] were used to obtain word vectors. Since the size of a word vector is 300, the dimension of the sentence vector obtained using the average rule is 300. The value for the feature "sentence position" is appended at the end of the sentence vector, which increases its dimension to 301. The value of the sentence position feature is calculated as the division of the position of the sentence in the document by the total sentences in the document. Hence, our final sentence vector is of dimension 301. The rationale behind including sentence position in the sentence vector is to encourage locally coherent sentences to fall in the same cluster. This helps in segmenting a document in a better way.

2.2.3.3 Sentence Clustering

In this step, the sentence vectors are clustered into clusters of different sizes. The idea is to group similar and closer sentences into the same cluster. To cluster the sentences, we have used the spectral clustering algorithm. To implement spectral clustering, we first calculate the affinity matrix from a document graph in which a node corresponds to a sentence vector, and the edge between two nodes is weighted by the similarity between the corresponding two vectors. The affinity matrix is created using the similarity function given in Equation 2.7, which is basically a Gaussian similarity function.

$$A_{ij} = exp(\frac{-d^2(s_i, s_j)}{\sigma^2}) \tag{2.7}$$

Where σ is a control parameter that controls the context window in our case. In equation 2.7, $d(s_i, s_j)$ denotes distance between two sentence vectors s_i and s_j. Distance between two points (x_1, x_2) and (y_1, y_2) is calculated using the formula of Euclidean Distance defined in equation 2.8. In our approach, we varied $sigma$ and got the best result when it is set to 10.

$$dist = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{2.8}$$

From the affinity matrix, the graph Laplacian matrix is obtained using equation 2.9.

$$L = D - A \tag{2.9}$$

where A is the affinity matrix and D is the degree matrix such that

$$d_i = \sum_{j|(i,j)\in E} W_{ij} \tag{2.10}$$

[2] https://fasttext.cc/docs/en/crawl-vectors.html

where E is the set of edges in the graph and W_{ij} refers to the similarity between two points x_i and x_j corresponding to two different sentences in a document. After normalizing the graph Laplacian matrix, the Eigen values and Eigen vectors of the normalized graph Laplacian matrix are used to embed sentences in a low-dimensional Eigen space [147]. Finally, a simple k-means clustering algorithm is applied to cluster the low-dimensional dense vectors to obtain hard clusters. The main problem with the K-means cluster algorithm is that it needs to specify the value of K in advance. In our case, we used an Eigen-Map heuristic method to determine the value of K. The main idea is to choose the value K such that all eigenvalues λ_1,....λ_k are very small but λ_{k+1} are relatively large. The details of this method can be found in [147]. We have used this method to determine the number of clusters. Thus, the number of topical segments is automatically inferred in an unsupervised way.

2.2.3.4 Cluster Ranking

The clusters are ranked in ascending order on the basis of the average of the position values of the sentences present in that cluster. The cluster ranking enables us to identify the most significant clusters from which sentence extraction will occur first. The rationale behind using the position-based cluster-ranking method is to ensure the selection of sentences in the summary from the topical clusters in order as they appear in the text (position-based topical order). This is useful for creating an informative extract of sentences that improves the coverage of various topics in a document.

2.2.3.5 Within-cluster Sentence Ranking

A particular cluster may have multiple sentences present in it. To identify the most salient sentence from each cluster, the sentences within a particular group are ranked using the graph-based lexical centrality method published in [54]. In this method, a weighted adjacency matrix is constructed for the graph representing each cluster where the sentences in the cluster are considered as the vertices and the cosine similarity is considered as the edge weights between two sentence vectors. Cosine similarity is one of the popular similarity measures between two vectors. It is the cosine of the angle between two vectors, which means the dot product of two vectors divided by the product of their lengths. The cosine similarity is calculated using the equation 2.11, where A and B are two sentence vectors belonging to a cluster. The rank of a sentence (a vertex) in a particular cluster graph is the sum of all cosine weights of the edges to all other vertices in the cluster graph. The higher the sum of edge weights, the higher is the rank of the sentence. This score is used to identify the sentence that is the most central to the cluster.

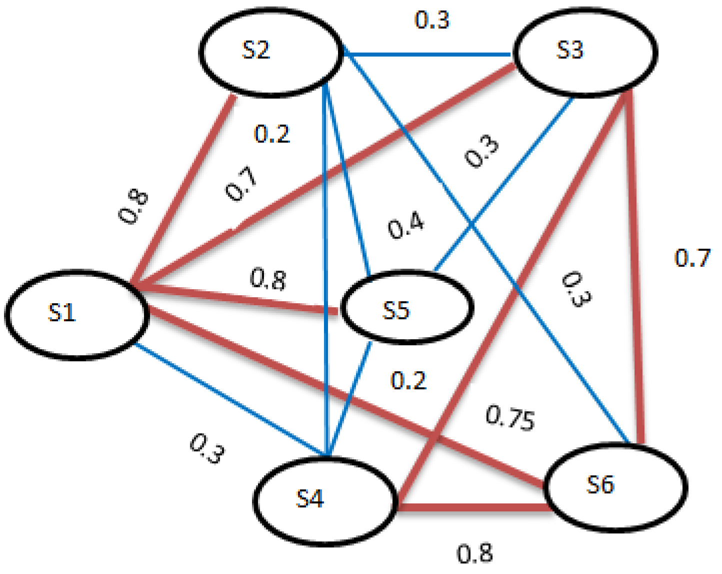


Figure 2.5: A sample similarity graph for the sentences in a cluster

$$cosine - similarity = \frac{A.B}{||A||||B||} \tag{2.11}$$

For example, let us assume that six sentences are present in a cluster. Now, an adjacency matrix is created for that cluster using the cosine similarity value. A sample weighted matrix for the graph representing a cluster is as follows.

$$\begin{array}{c|cccccc} & s_1 & s_2 & s_3 & s_4 & s_5 & s_6 \\ \hline s_1 & 0 & 0.8 & 0.7 & 0.3 & 0.8 & 0.75 \\ s_2 & 0.8 & 0 & 0.3 & 0.2 & 0.4 & 0.3 \\ s_3 & 0.7 & 0.3 & 0 & 0.6 & 0.3 & 0.7 \\ s_4 & 0.3 & 0.2 & 0.6 & 0 & 0.2 & 0.8 \\ s_5 & 0.8 & 0.4 & 0.3 & 0.2 & 0 & 0.9 \\ s_6 & 0.75 & 0.3 & 0.7 & 0.8 & 0.9 & 0 \end{array}$$

In the above sentence similarity matrix, the row sum gives the sum of the similarities of a given sentence to the other sentences in the cluster they belong to. From the above sentence similarity matrix, we can observe that s_6 has the highest saliency score and it should be selected first as a summary worthy sentence from the cluster.

2.2.3.6 Summary Generation

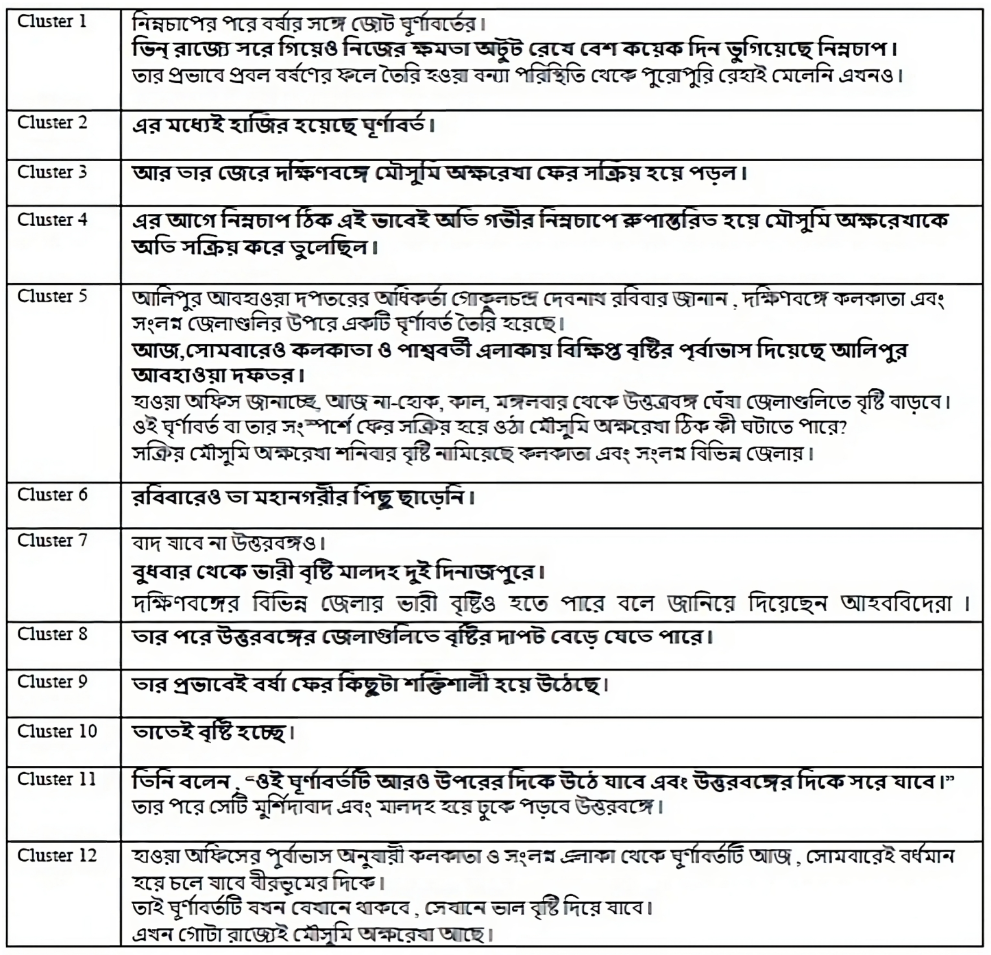
After cluster ordering,ranking of the sentences within each group (cluster) is carried out. Then the sentence extraction process begins to create a summary for each input document. Here, we select the first-ranked sentence in the first cluster, followed by the first-ranked sentence in the second cluster, and so on. If the number of clusters produced by the clustering algorithm is less than the required number of sentences, the process is repeated in a round-robin fashion until we obtain the required number of sentences. Once the required number of sentences is extracted to attain the desired summary length, the process is stopped. In our approach, a summary of 100 words is taken for evaluation. The algorithm for the overall summary generation process is shown in Algorithm 2.

Algorithm 2 Summary generation using spectral clustering based document segmentation

```
Input: A text document.
Output: Summary of the document.
 1: Breaking the input document into sentences.
 2: Removal of stop words from the sentences.
 3: Calculation of the 300-dimensional sentence vector by taking an average of
    word vectors obtained using the fastText open source.
 4: Calculate the positional feature value for each sentence and append it with the
    300-dimensional sentence vector obtained in the previous step. The positional
    feature value is calculated as the position of the sentence in the document
    divided by the total number of sentences in the document.
 5: Sentence affinity matrix is created using the Gaussian similarity function.
 6: Compute the normalized graph Laplacian for the sentence affinity matrix.
 7: Eigenvalue decomposition is performed on the normalized graph Laplacian
    to get eigenvalues and eigenvectors. The eigenvectors are arranged in the
    ascending order of eigenvalues.
 8: Take the first d eigenvectors to form an $N \times d$ matrix U. A matrix T is obtained
    from U by normalizing the rows of U to norm 1.
 9: Eigen gap heuristic is applied to identify the gap which gives the optimal
    number of clusters, K.
10: Treating each row of T as a spectral embedding of a sentence, a simple K-means
    algorithm is applied on T to obtain K clusters where K is computed using the
    Eigen gap heuristic.
11: Clusters are ranked on the basis of the average position values of the sentences
    belonging to a cluster.
12: The sentences within each cluster are assigned scores based on the graph
    centrality-based saliency measure [54].
13: For creating a summary, the top-ranked cluster contributes first its best sen-
    tence to the summary and then the second-ranked cluster contributes, and so
    on. If the number of clusters produced by the clustering algorithm is less than
    the required number of sentences, the process is repeated in a round-robin
    fashion until we obtain the required number of sentences.
```

The clustering and summarization results are shown for an example Bengali input document. The clusters produced by the spectral clustering algorithm are shown in Figure 2.6, and a reference summary and the system generated summary is shown in Figure 2.7.

The bold sentences in Figure 2.6 are the sentences selected by the summarization model proposed by us. Although we have shown in Figure 2.7 a system-generated summary consisting of 11 sentences, the first 100 words of it are taken during evaluation using the ROUGE package.

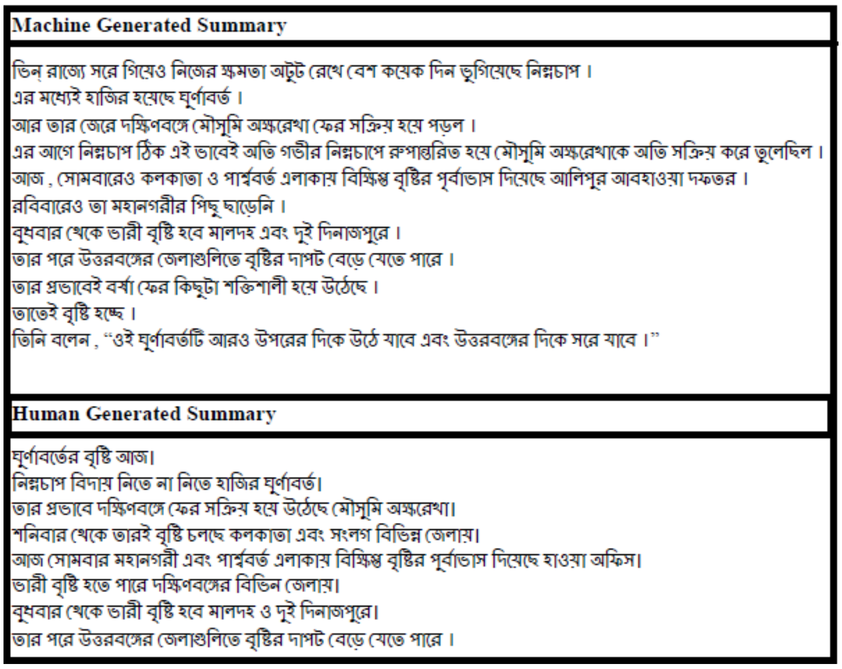


Cluster 1	নিম্নচাপের পরে বর্ষার সঙ্গে জোট ঘূর্ণাবর্তের। **ভিন্ রাজ্যে সরে গিয়েও নিজের ক্ষমতা অটুট রেখে বেশ কয়েক দিন ভুগিয়েছে নিম্নচাপ।** তার প্রভাবে প্রবল বর্ষণের ফলে তৈরি হওয়া বন্যা পরিস্থিতি থেকে পুরোপুরি রেহাই মেলেনি এখনও।
Cluster 2	**এর মধ্যেই হাজির হয়েছে ঘূর্ণাবর্ত।**
Cluster 3	**আর তার জেরে দক্ষিণবঙ্গে মৌসুমি অক্ষরেখা ফের সক্রিয় হয়ে পড়ল।**
Cluster 4	**এর আগে নিম্নচাপ ঠিক এই ভাবেই অতি গভীর নিম্নচাপে রূপান্তরিত হয়ে মৌসুমি অক্ষরেখাকে অতি সক্রিয় করে তুলেছিল।**
Cluster 5	আলিপুর আবহাওয়া দপ্তরের অধিকর্তা গোকুলচন্দ্র দেবনাথ রবিবার জানান , দক্ষিণবঙ্গে কলকাতা এবং সংলগ্ন জেলাগুলির উপরে একটি ঘূর্ণাবর্ত তৈরি হয়েছে। **আজ,সোমবারেও কলকাতা ও পার্শ্ববর্তী এলাকায় বিক্ষিপ্ত বৃষ্টির পূর্বাভাস দিয়েছে আলিপুর আবহাওয়া দফতর।** হাওয়া অফিস জানাচ্ছে, আজ না-হোক, কাল, মঙ্গলবার থেকে উত্তরবঙ্গ ঘেঁষা জেলাগুলিতে বৃষ্টি বাড়বে। ওই ঘূর্ণাবর্ত বা তার সংস্পর্শে ফের সক্রিয় হয়ে ওঠা মৌসুমি অক্ষরেখা ঠিক কী ঘটাতে পারে? সক্রিয় মৌসুমি অক্ষরেখা শনিবার বৃষ্টি নামিয়েছে কলকাতা এবং সংলগ্ন বিভিন্ন জেলায়।
Cluster 6	**রবিবারেও তা মহানগরীর পিছু ছাড়েনি।**
Cluster 7	বাদ যাবে না উত্তরবঙ্গও। **বুধবার থেকে ভারী বৃষ্টি মালদহ দুই দিনাজপুরে।** দক্ষিণবঙ্গের বিভিন্ন জেলায় ভারী বৃষ্টিও হতে পারে বলে জানিয়ে দিয়েছেন আহবিদেরা ।
Cluster 8	**তার পরে উত্তরবঙ্গের জেলাগুলিতে বৃষ্টির দাপট বেড়ে যেতে পারে।**
Cluster 9	**তার প্রভাবেই বর্ষা ফের কিছুটা শক্তিশালী হয়ে উঠেছে।**
Cluster 10	**তাতেই বৃষ্টি হচ্ছে।**
Cluster 11	**তিনি বলেন , "ওই ঘূর্ণাবর্তটি আরও উপরের দিকে উঠে যাবে এবং উত্তরবঙ্গের দিকে সরে যাবে।"** তার পরে সেটি মুর্শিদাবাদ এবং মালদহ হয়ে ঢুকে পড়বে উত্তরবঙ্গে।
Cluster 12	হাওয়া অফিসের পূর্বাভাস অনুযায়ী কলকাতা ও সংলগ্ন এলাকা থেকে ঘূর্ণাবর্তটি আজ , সোমবারেই বর্ধমান হয়ে চলে যাবে বীরভূমের দিকে। তাই ঘূর্ণাবর্তটি যখন যেখানে থাকবে , সেখানে ভাল বৃষ্টি দিয়ে যাবে। এখন গোটা রাজ্যেই মৌসুমি অক্ষরেখা আছে।

Figure 2.6: Clusters produced using spectral clustering algorithm

2.3 Evaluation and Experimental Results

Since no publicly available dataset is available for Bengali text summarization, we have tested our proposed approaches on our own dataset consisting of 102 Bengali document-summary pairs. The average number of sentences in each document is 40. Each document has approximately 40 sentences. All the documents have been collected from a famous Bengali daily newspaper, Anandabazar Patrika. For evaluation, we have taken 100 words from each summary generated by the system using the proposed approach.



Machine Generated Summary

ভিন্ রাজ্যে সরে গিয়েও নিজের ক্ষমতা অটুট রেখে বেশ কয়েক দিন ভুগিয়েছে নিম্নচাপ ।
এর মধ্যেই হাজির হয়েছে ঘূর্ণাবর্ত ।
আর তার জেরে দক্ষিণবঙ্গে মৌসুমি অক্ষরেখা ফের সক্রিয় হয়ে পড়ল ।
এর আগে নিম্নচাপ ঠিক এই ভাবেই অতি গভীর নিম্নচাপে রুপান্তরিত হয়ে মৌসুমি অক্ষরেখাকে অতি সক্রিয় করে তুলেছিল ।
আজ , সোমবারেও কলকাতা ও পার্শ্ববর্ত এলাকায় বিক্ষিপ্ত বৃষ্টির পূর্বাভাস দিয়েছে আলিপুর আবহাওয়া দফতর ।
রবিবারেও তা মহানগরীর পিছু ছাড়েনি ।
বুধবার থেকে ভারী বৃষ্টি হবে মালদহ এবং দুই দিনাজপুরে ।
তার পরে উত্তরবঙ্গের জেলাগুলিতে বৃষ্টির দাপট বেড়ে যেতে পারে ।
তার প্রভাবেই বর্ষা ফের কিছুটা শক্তিশালী হয়ে উঠেছে ।
তাতেই বৃষ্টি হচ্ছে ।
তিনি বলেন , “ওই ঘূর্ণাবর্তটি আরও উপরের দিকে উঠে যাবে এবং উত্তরবঙ্গের দিকে সরে যাবে ।”

Human Generated Summary

ঘূর্ণাবর্তের বৃষ্টি আজ।
নিম্নচাপ বিদায় নিতে না নিতে হাজির ঘূর্ণাবর্ত।
তার প্রভাবে দক্ষিণবঙ্গে ফের সক্রিয় হয়ে উঠেছে মৌসুমি অক্ষরেখা।
শনিবার থেকে তারই বৃষ্টি চলছে কলকাতা এবং সংলগ বিভিন্ন জেলায়।
আজ সোমবার মহানগরী এবং পার্শ্ববর্ত এলাকায় বিক্ষিপ্ত বৃষ্টির পূর্বাভাস দিয়েছে হাওয়া অফিস।
ভারী বৃষ্টি হতে পারে দক্ষিণবঙ্গের বিভিন জেলায়।
বুধবার থেকে ভারী বৃষ্টি হবে মালদহ ও দুই দিনাজপুরে।
তার পরে উত্তরবঙ্গের জেলাগুলিতে বৃষ্টির দাপট বেড়ে যেতে পারে ।

Figure 2.7: System generated summary and reference summary of the document

2.3.1 Evaluation

To calculate the performance score of the proposed models, we have used the popular summary evaluation package called ROUGE [141] which is already described in the chapter 1 (Subsection 1.6).

2.3.2 Experiments

We have carried out the following experiments to prove the effectiveness of our proposed models.

Experiment 1 In the first experiment, we combined the saliency score and the novelty score to compute the final score of a sentence. Without using any re-ranker or clustering of sentences, the proposed approach can automatically take care of the diversity issue while producing a summary. The method is discussed in detail in Section 2.2.1.

Experiment 2

In the second experiment, we have implemented an LSI(Latent Semantic Indexing) based model for performing unsupervised text summarization. It captures semantic relationships among words in a text by mapping relationships among terms and sentences in semantic space. Here, we propose a sentence scoring method that assigns a score to a sentence based on the importance of semantic concepts shared by the sentence. This approach is described in Section 2.2.2.

Experiment 3

In the third experiment, we have used sentence embedding and spectral clustering to perform unsupervised text summarization. The spectral clustering method is useful in text summarization because it does not assume any fixed shape of the clusters, and the number of clusters can automatically be inferred using the eigengap method. In our approach, we have used word embedding-based sentence representation and a spectral clustering algorithm to identify various topics covered in a Bengali document and generate an extractive summary by selecting salient sentences from the identified topics. This method is explained in Section 2.2.3.

2.3.3 Results and discussion

We have compared performance of our three proposed models on our own Bengali dataset consisting of 102 Bengali document-summary pairs. The average number of sentences in each document is 40. Each document has approximately 40 sentences. All the documents have been collected from a famous Bengali daily newspaper, Anandabazar Patrika.

2.3.3.1 Comparison with existing methods

We have carried out six experiments to prove the effectiveness of our best model proposed. These experiments include implementations of five state-of-the-art unsupervised methods with which our proposed approach is compared. A brief description of the models implemented by us is given below.

Luhn, 1958[2] :This approach was developed by Luhn [2]. It generates a summary from a document by considering that the sentence containing more frequent words is more important than the sentence containing less frequent words. In this method, stop words are removed before sentence weight calculation.
Giines and Dragomir R., 2004[148]: This approach was developed by [148]. It uses Lexrank, which is a graph-based approach that represents sentences as vertices of a graph and considers the cosine similarity between any two sentences as the weight of the edge between the corresponding vertices. Finally, Google's page rank algorithm is applied to the graph to rank sentences.
Ani and Lucy, 2005[149]: This approach was developed by [149]. The approach is named "Sumbasic" which considers term frequency as the saliency of a term. Here probability of a word is calculated based on its frequency and each sentence is assigned a score equal to the average probability of the words contained in the sentence. The sentence with the highest score is selected first in the summary.

Before selecting the next sentence, the words present in the already selected sentence are penalized by multiplying their probability values by themselves, and the sentences are re-ranked using the newly calculated probability values. After re-ranking the sentences, the sentence with the highest score is selected as the second sentence of the summary. This process is continued until the summary of the desired length is obtained.
Randa and Paul, 2004[150]: This model was developed by [150]. It is called as Textrank. It is also a graph-based approach similar to that used in LexRank[148]. LexRank used TF*IDF-based term weight and cosine similarity value as the edge weight whereas TextRank used word overlap-based similarity value as the edge weight.
DUC Lead Baseline: This is the lead baseline model, where a summary is generated by considering the first 100 words of the input document. This is the baseline defined in DUC 2001 and DUC 2002 shared tasks on single document summarization.

We have implemented five existing summarization systems using existing summarizers that are available in a Python package named Sumy [3]. Sumy is a Python library of various existing summarization tools that can be used to extract summary from HTML pages or plain text.

The comparison results are shown in Table 2.1.

MODEL	Rouge 1	Rouge 2	Rouge SU4
Model 1(Combining saliency and novelty)	0.4079	0.2656	0.2697
Model 2 (LSA based model)	0.4172	0.2703	0.2751
Model 3 (Spectral clustering based model)	**0.4481**	**0.2844**	**0.2848**
Luhn, 1958[2]	0.3929	0.2324	0.2293
Giines and Dragomir R., 2004[148]	0.3693	0.1995	0.1980
Ani and Lucy, 2005[149]	0.3602	0.1831	0.1836
Randa and Paul, 2004[150]	0.3499	0.1846	0.1835
DUC Lead Baseline	0.2733	0.1501	0.1487

Table 2.1: Performance of our proposed best summarization model and its comparison with some existing summarization methods

As we can see from the table 2.1, our proposed model (Model 1) performs significantly better than the other baseline models to which it is compared. The proposed model also performs significantly better than Model 3 [148] which uses the graph-based ranking of all sentences of the input document considering all sentences in the document as a single cluster. Compared to the system "LexRank " [148], we use spectral clustering-based document segmentation and within-cluster

[3]https://github.com/miso-belica/sumy

sentence ranking. It is evident from the results that, instead of taking the input document as a single cluster of sentences, if the document is segmented into multiple topical clusters and the summary is generated by choosing sentences from the clusters one by one, this produces a summary which is better in quality than that produced by the system called ”LexRank”.

We have calculated performance improvement using equation 2.12 that computes the difference between the ROUGE scores obtained by the proposed model and the model to which it is compared.

$$PI = M - N \tag{2.12}$$

where PI denotes performance improvement, M is the score for the proposed approach, and N is the score for the approach to which the proposed approach is compared. Performance Improvement of our proposed approach over other approaches is shown in Table 2.2.

MODEL	Rouge 1	Rouge 2
Luhn, 1958[2]	0.0552	0.052
Giines and Dragomir R., 2004[148]	0.0788	0.0849
Ani and Lucy, 2005[149]	0.0879	0.1013
Randa and Paul, 2004[150]	0.0982	0.0998
DUC Lead Baseline	0.1748	0.1343

Table 2.2: Performance improvement achieved by our proposed model in comparison with some state-of-the-art summarization methods

As we can see from Table 2.2, the proposed approach shows improvement over the lead baseline and the LexRank[148] by 0.1748 and 0.0788 ROUGE-1 points respectively.

2.3.4 Parameter tuning

In this subsection, we present how we obtain the optimal values for the tunable parameters of the proposed models.

Parameter tuning for experiment 1

The value of the parameter λ mentioned in equation 2.4 has been varied to achieve the best performance. The effect of varying λ is shown in Figure 2.9. Indicates that the best rouge score is obtained by setting λ to 15.

Parameter tuning for experiment 2

Since SVD maps *n* dimensional vectors (where a vector corresponds to a sentence) to k dimensional space by breaking down the original term-by-sentences matrix into k independent base vectors that express k different concepts or topics in

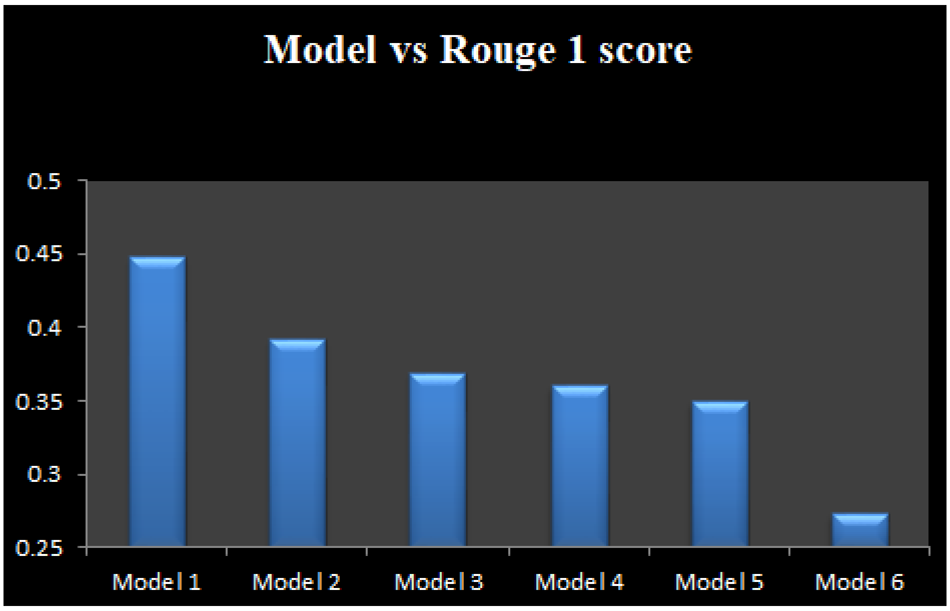


Figure 2.8: Performance of our proposed best summarization model and its comparison with some existing summarization methods

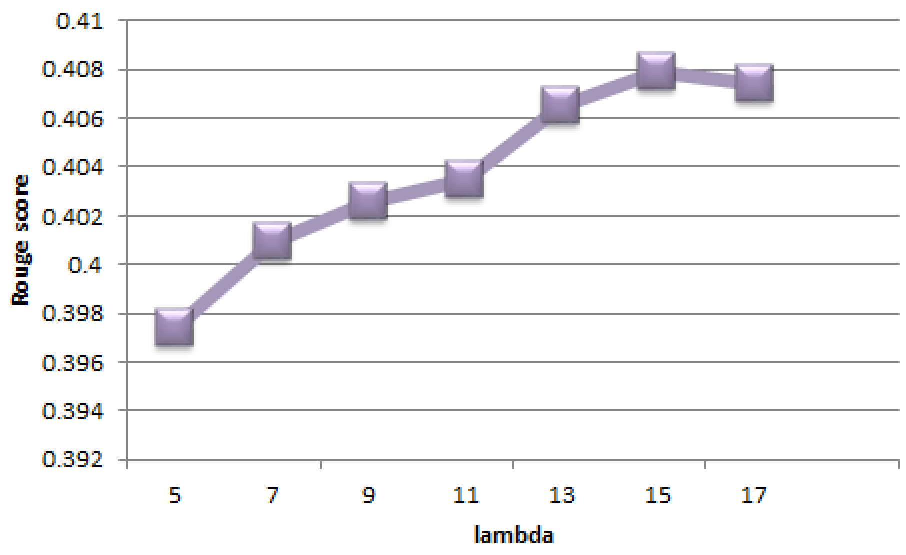


Figure 2.9: Performance of combining saliency and novelty score model varying lambda

the document, the most important system parameter whose value affects our LSI based system performance is k. We need to adjust the value of k for better performance. To tune the values of k, we have randomly chosen 48 documents from our total collection of 102 documents and tested the system performance on the entire collection of 102 documents with varying values of k. We have shown in Fig. 2.10 the graph that shows how the value of k affects the summarization system performance. As we can see from Fig. 2.10, the best F-Score is obtained when the value of k is set to 4.

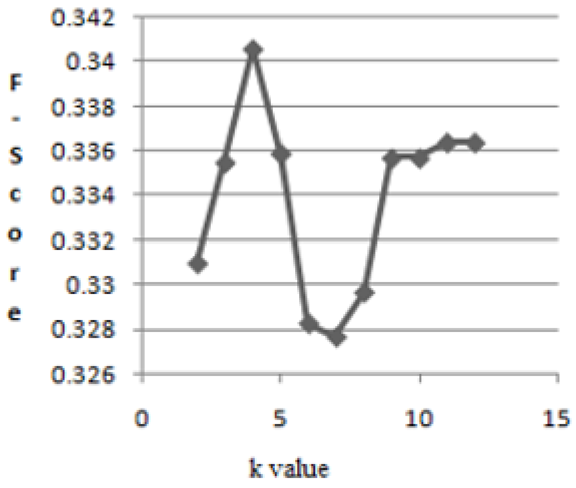


Figure 2.10: Effect of the varying values of k

Parameter tuning for experiment 3
To implement spectral clustering, we first calculate the affinity matrix from a document graph in which a node corresponds to a sentence vector, and the edge between two nodes is weighted by the similarity between the corresponding two vectors. The affinity matrix is created using the similarity function given in Equation 2.7, which is basically a Gaussian similarity function. Here, σ is a control parameter that controls the context window in our case. We varied $sigma$ and got the best result when it is set to 10 which is shown in figure 2.11.

2.4 Chapter Summary:

In this chapter, we have proposed three single document extractive summarization methods (1) A method that combines saliency and novelty score for sentence ranking, (2) LSA based method for single document text summarization, and (3) a spectral clustering-based single document extractive text summarization method. The proposed methods have been evaluated using a Bengali text summarization dataset developed by us. Each of the proposed methods has shown better performance in the Bengali summarization dataset. Among the three methods proposed by us, the spectral clustering-based summarization method performs the best.

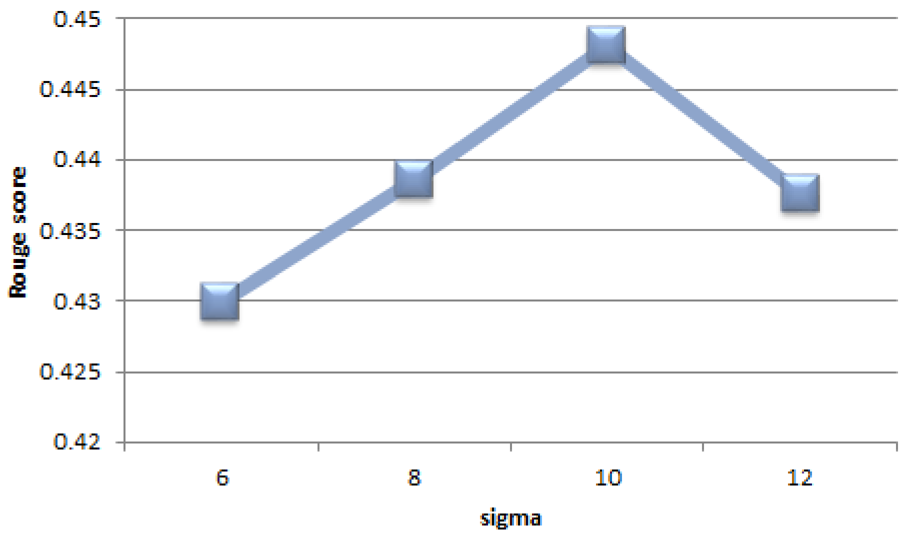


Figure 2.11: Effect of the varying values of sigma

However, one of the problems of our proposed approach is that the saliency and diversity of the generated summary are not balanced well. To address this, we apply the neural network model. The neural network is effective in handling this issue. In our next chapter, we will discuss extractive text summarization using a neural network.

3 Supervised Extractive Single Document Text Summarization Using Neural Networks

3.1 Introduction

Unsupervised approaches to extractive text summarization presented in Chapter 2 generate a summary from a document using a two-stage process. In the first stage, the sentences are ranked based on their scores where a score of a sentence is calculated using one or more features such as number of keywords words, sentence position, novel information, similarity to the latent concepts of the document, similarity to the centroids of clusters into which a document is decomposed, etc. In the second stage, the summary generation process starts with the top-ranked sentence and selects the next sentences one by one from the ranked list. To improve summary diversity, a sentence is included in the summary if the sentence is sufficiently dissimilar from the already selected sentences. Sentence selection is continued until the summary of the desired length is reached. The second stage is greedy in nature and uses a predefined similarity threshold value to check the dissimilarity of a sentence with the already selected sentences. Due to this fixed similarity threshold, which is manually tuned, in most cases, this approach fails to manage the diversity in a summary.

In this chapter, we propose a summarization approach that uses a neural network-based learning model that learns to include a sentence in a summary by taking into account both the saliency of the sentence and the diversity in the summary. For this purpose, the model is trained using two types of features- saliency features and diversity features. We have evaluated our approach using two open benchmark datasets- the DUC dataset and the Daily Mail dataset. Experimental results show that the proposed neural summarization approach is effective in producing better non-redundant informative summaries and outperforms many existing summarization approaches to which it is compared.

The main difference between the proposed summarization approach and the existing sequence learning-based approaches to extractive text summarization[151][68] is as follows. The proposed approach includes a sentence in a summary based on the sentence's saliency features (a set of hand-crafted features) and the diversity features for trading off saliency and diversity in the summary. The diversity features ensure whether the inclusion of the sentence in the summary improves diversity in the summary or not. In contrast, the existing recurrent neural network(RNN)-based approaches assign 0 or 1 labels to a sentence based on the information contained in the current sentence and all the labels of the past sentences including both the summary worthy and unworthy sentences encountered before the current sentence. Diversity issues are less emphasized in these learning-based approaches.

To implement our summary generation model, we have used deep neural networks(DNN) and word embedding-based sentence representation. The deep neural network (DNN) that we have used is a feedforward neural network consisting of multiple hidden layers with Relu activation functions.

The notation used throughout this chapter is summarized in Table 3.1.

Table 3.1: Notation used throughout the chapter

Notation	Description
SFV(S)	Saliency feature vector for an individual sentence S
RVec(S,$summ$)	Relations of the current sentence S with the sentences selected in the summary in the previous time steps
RV_i	Relation vector for the sentences S_i
FV(S)	Overall feature vector when a sentence S is submitted
TF(w)	Number of times term w appears in a document
IDF(w)	Inverse document frequency of the term w
Summary-so-far	A set of sentences included in the summary at a time step t

In the first part of the chapter, we discuss training data preparation, which is crucial for the proposed model. In the second part, we discuss how the trained model works to generate a from an input text document In the third part, we present experimental results. Performance of our proposed approach and comparison with existing approaches are also discussed in this chapter. In Subsection 3.3.2, we present error analysis. Finally, we include the chapter with discussion of the future scope of our proposed approach.

3.2 The proposed approach to extractive text summarization using neural network

We present the framework for the proposed neural summarization model that learns to include the salient sentences in the summary by taking summary diversity into account. We first discuss the working procedure of the proposed model and then discuss how the model is trained. Given a sentence of a document and the previously selected summary sentences(throughout this chapter, the group of previously selected sentences is referred to by Summary-so-far) as input, the proposed learning model decides whether the given sentence is worth including in the summary or not. The model is trained in such a way that it favors the sentences which are salient and whose inclusion improves the summary diversity. The saliency of a sentence is defined by a set of handcrafted features and the summary diversity is defined by a set of relation features which are defined by the semantic relations of the sentence with the other sentences in the Summary-so-far. Initially, the Summary-so-far is set to empty. During testing, the sentences from the input document are submitted in text order to the trained model one by one, and if the model finds them suitable for inclusion in the summary, they are added to the summary. In this framework, the model is repeatedly invoked until the summary of the desired length is generated.

Figure 3.1 shows an example of how our proposed DNN-based neural model for text summarization works. In this figure, SFV(S) is the saliency feature vector for an individual sentence S, which is computed based on a set of hand-crafted features discussed in the training phase section and RV (we call it relation vector) is a dynamic feature vector which dynamically gets new values based on the relations of the current sentence S with all other sentences selected in the summary at the previous time steps. The function RVec(S_t,$summ_{t-1}$) finds the semantic relations between the input sentence S_t at time step t and the previously selected sentences at time step t-1 and returns RV_t. For example, RV_t becomes a 5-dimensional vector at time step t, if the summary which is created so far (i.e., $summ_{t-1}$) contains 5 sentences. In this case, each component of RV_t is a semantic similarity value between a sentence under consideration(S_t) and a previously selected summary sentence belonging to $summ_{t-1}$. To compute the semantic relation between a given sentence and a previously selected summary sentence, we represent both sentences into sentence vectors using word embeddings and apply the cosine similarity measure. The implementation of the function RVec(S_t,$summ_{t-1}$) is discussed later in this chapter. We have used this type of relation vector so that, before selecting a sentence S, DNN can detect how much diverse (redundant) information S contains in comparison with other sentences previously selected in the summary. In Figure 3.1, S_1, S_2, .., S_n are used to denote sentences from a document, for which

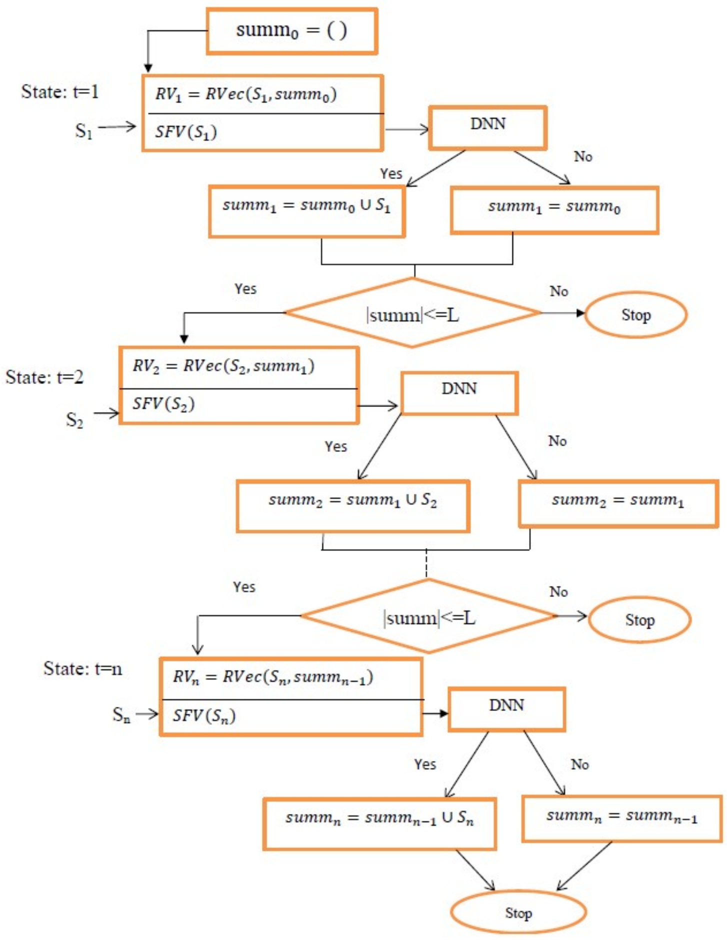


Figure 3.1: Working procedure of the proposed model: S_1, S_2, .., S_n are the sentences belonging to an input document to be summarized. DNN accepts two inputs at each time step: a saliency feature vector for an individual sentence S, denoted by SFV(S), and a diversity feature vector (or a relation vector) denoted by RV. $summ_t$ stores the summary created at time step t. RV_t is returned by a function RVec(S_t, $summ_{t_1}$) that finds the relations of the current sentence S_t with the sentences selected in the summary at the previous time step t-1. Since $summ_0$=( ) at t=0, RV_1 is taken as a zero vector at t=1. In the subsequent time steps, RVs dynamically get new values since *summ* is dynamically updated based on the output of the net at the previous time steps. Similarly, RV_2 is a relation vector when S_2 is submitted as an input sentence.

the corresponding saliency feature vectors are denoted by SFV(S_1), SFV(S_2), .., SFV(S_n) respectively. The relation vectors in the time steps: t = 1, t = 2, and t = n are RV_1, RV_2, and RV_n, respectively. At each time step t, the input to the

Deep Neural Network (DNN) is a concatenation of the saliency vector SFV(S_t) and the relation vector RV_t. RV_1 is a zero vector because the summary at t = 0 is empty (that is, $summ_0$=( ), which means it initially contains NULL sentences). In the first step(t=1), SFV(S_1)+RV_1 is submitted to the model. If the model predicts Yes, S_1 is included, otherwise it is not included in the summary, that is, at t=1, $summ_1 = summ_0 \cup S_1$ if S_1 is selected based on the prediction of the model, otherwise $summ_1 = summ_0$. Now, at time t = 2, when the sentence S_2 is being submitted to the model, the new input to the model is $SFV(S_2) + RV_2$, where SFV(S_2) is the saliency feature vector for S_2 and RV_2 is the relation vector returned by RVec(S_2, $summ_1$), where $summ_1$ is the summary created at time step t=1. Thus, to generate a summary from a document, the sentences S_1, S_2,..,S_n of a document are scanned sequentially from the beginning and are submitted to the trained model one by one. In any state where a sentence is added to the summary, the sentence is deleted from the input set, the scanning is stopped at this input sentence number, and the input sentence set is reduced by 1. Then, this reduced input set is scanned again from the beginning, and the process is repeated. In this way, any sentence which is not selected in the previous time step due to the unfavorable RV pattern can get further chances to be assigned to the summary if the RV pattern becomes favorable to it in the subsequent time steps. At each time step, the produced summary length is compared with the desired summary length. The process stops if the length of the produced summary exceeds the desired summary length(which is 6 sentences in our case). In Figure 3.1, $|summ| < L$ is used to compare the length of system summary with the desired summary length.

The major problem with relation vector RVs is that they may not be equal in size all the time because the summary size may grow from one time step to another time step. To deal with this problem, we have carefully designed the training data. This issue is discussed in detail in the training data preparation section of this chapter.

The deep neural network(DNN) that we have used for implementing the proposed summarization model shown in Figure 3.1 is basically a feed-forward neural network having multiple hidden layers and a softmax layer. The specification of the DNN used for implementing the proposed model is given in Figure 3.2.

The development process of the proposed summarization model has two main phases - (1) the training phase and (2) the testing phase. The training phase has several important steps: (1) breaking a document and an associated reference summary into sentences, (2) representing each training sentence into a certain pattern suitable for training the proposed DNN-based summarization model, (3) generating positive and negative examples, (4) Converting each positive and negative example into numerical feature vectors, (5) labeling feature vectors, (6) training a DNN model. The testing phase also includes (1) breaking a test document into

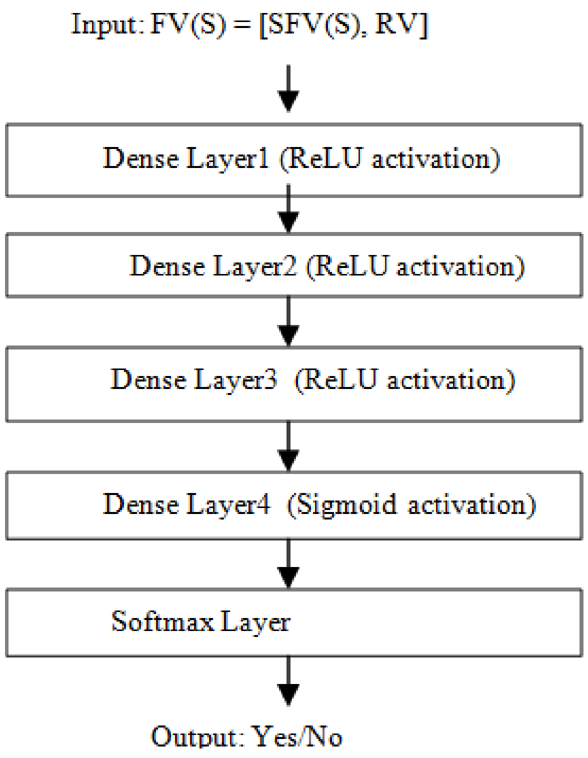

Input: FV(S) = [SFV(S), RV]
Dense Layer1 (ReLU activation)
Dense Layer2 (ReLU activation)
Dense Layer3 (ReLU activation)
Dense Layer4 (Sigmoid activation)
Softmax Layer
Output: Yes/No

Figure 3.2: Block diagram of DNN used for implementing the proposed model. FV(S): Overall feature vector when a sentence S is submitted, SFV(S): Saliency feature vector, RV: Relation vector representing semantic relations between S and the other sentences previously selected in the summary

sentences, (2) representing each test sentence in a certain pattern similar to that used for representing the training sentence, (3) converting each test pattern to a feature vector, and (4) invoking the trained DNN model to label the test sentence as either ”Yes” (summary worthy) or no (summary unworthy). The general architecture of our proposed model is shown in Figure 3.3. In Figure 3.3, L denotes the desired summary length specified by the user, $|Summary - so - far|$ denotes the length of the summary created up to a certain time step, and S_LIST is a list of sentences obtained after breaking a test document.

In the subsequent subsections, we have discussed in detail each step shown in Figure 3.3.

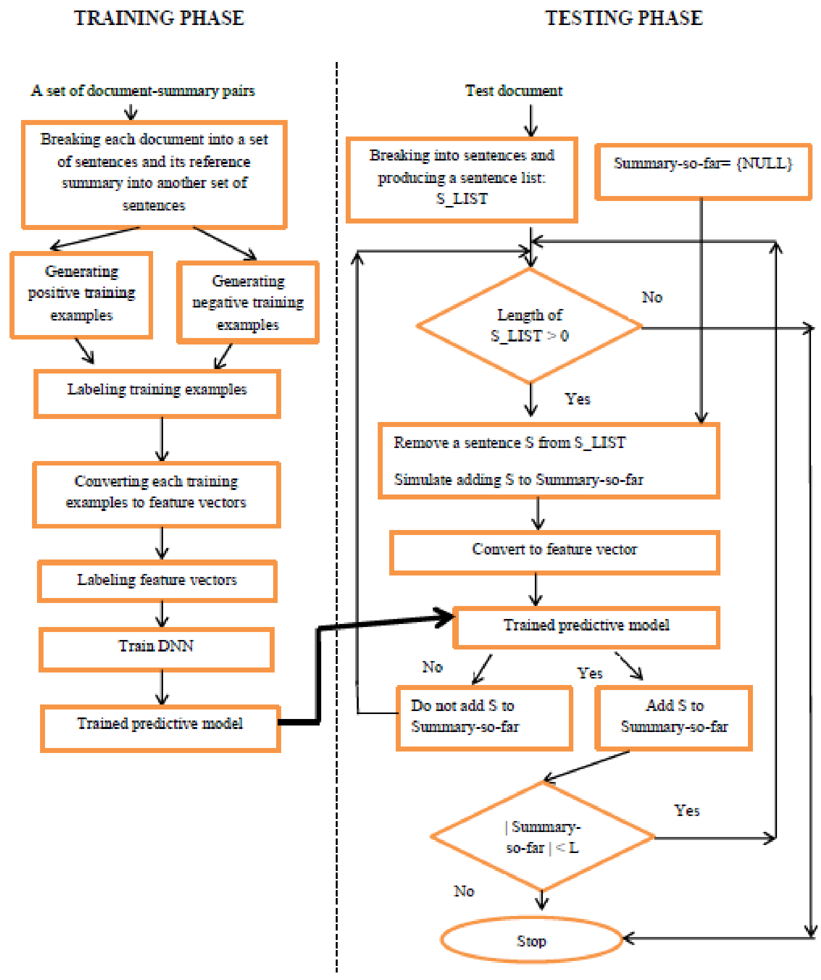

TRAINING PHASE
TESTING PHASE
A set of document-summary pairs
Breaking each document into a set of sentences and its reference summary into another set of sentences
Generating positive training examples
Generating negative training examples
Labeling training examples
Converting each training examples to feature vectors
Labeling feature vectors
Train DNN
Trained predictive model
Test document
Breaking into sentences and producing a sentence list: S_LIST
Summary-so-far= {NULL}
Length of S_LIST > 0
No
Yes
Remove a sentence S from S_LIST
Simulate adding S to Summary-so-far
Convert to feature vector
Trained predictive model
No
Yes
Do not add S to Summary-so-far
Add S to Summary-so-far
| Summary-so-far | < L
Yes
No
Stop

Figure 3.3: Overall architecture of the proposed model

3.2.1 Training phase

The proposed model needs to be trained in such a way that it can predict whether a given sentence is a summary sentence or not based on the saliency of the sentence and its relations with the sentences which have already been selected in the summary. Since we have used the traditional backpropagation algorithm for training the proposed model, to make the model workable in the manner mentioned above, the training data need to be designed in an appropriate way. The process of preparing training data is discussed in detail in the next subsection.

3.2.1.1 Training data preparation

The proposed model is intended to produce an extractive summary. For training such an extractive summarization model, we need to have a set of document-extract pairs. We have carried out our experiments using DUC datasets [1] and Daily Mail dataset[2] [152] for single-document text summarization. The DUC2001 and DUC2002 datasets are used, respectively, to train and test the proposed model. For the Daily Mail(DM) dataset, 90% of the documents are used for training. The DUC 2001 dataset contains 309 articles. For every article in the DUC 2001 dataset, multiple reference summaries (human abstracts) are available, whereas one reference summary(highlights) is available for each document in the DM dataset. Since the available reference summaries are not extracts, they cannot be directly used for training a supervised learning model for extractive summarization. Therefore, we need to collect golden extracts for each training document. However, manual creation of golden extracts from a large number of training documents is a laborious task. We have applied a greedy oracle method to create an oracle extract for each training document. The oracle method[153] is a method that uses a set of gold summaries (human-created abstracts) and the corresponding document to produce an extract of the document. The greedy oracle method compares each sentence in the document with the golden abstracts using a similarity measure and selects the top-ranked K sentences to form the oracle extract. The oracle method that we have used to generate the oracle extract is given in Algorithm 3.

Since the single document summary generation task in DUC 2001 and 2002 was to generate a 100-word summary from each document and the Daily Mail(DM) dataset contains a relatively short reference summary for each document, we set the summary length to 6 sentences. We have experimentally determined this summary length. We have observed that fixing the summary length to 6 sentences is sufficient for a 100-word summary generation. For the proposed summarization model, fixing the summary length to a predefined number of sentences is also necessary for obtaining the equal-length feature vectors for all the sentences in the input document. Although all the algorithms used for preparing training data are designed with the assumption that the learning model is trained for the 6-sentence summary generation task, the designed algorithms can be easily ported to a *m*-sentence summary generation task where *m* can take any value between 1 and the document length.

After generating the 6-sentence oracle extract from each training document, we use it to generate training examples suitable for training the proposed learning model, we want to train our learning model in such a way that it can predict the summary worthiness of a sentence S based on the input, which is the concatenation

[1]https://duc.nist.gov/data.html

[2]https://github.com/JafferWilson/Process-Data-of-CNN-DailyMail

Algorithm 3 The oracle method for generating an oracle extract from a training document

Input:A document D, a set of reference summaries(human created abstracts) R, and the size of oracle extract K
Output: A oracle extract OE

```
1:  for each sentence $S_i \in$ D do
2:      sim-sum=0
3:      for each reference summary r $\in$ R do
4:          $S_i$=Porter-Stemmer($S_i$)
5:          r=Porter-Stemmer(r)
6:          Similarity($S_i$,r)=$\frac{|S_i \cap r|}{|r|}$
7:          sim-sum=sim-sum+Similarity($S_i$,r)
8:      end for
9:      score-dictionary[$S_i$] = $\frac{sim-sum}{|R|}$
10: end for
11: Sort score-dictionary based on value (scores for sentences)
12: Choose top-ranked K sentences from the sorted dictionary
    to create an oracle extract OE
13: Return OE
```

of the saliency feature vector(SFV) for S and the diversity feature vector or relation vector(RV) that represents the relations of S with the sentences that have been selected in the summary in the previous time steps. To achieve this, we first represent the training examples in a suitable format. Since the proposed model is designed to generate the 6-sentence summary from each document, we represent each training example in the following format. [<<P_1, P_2, P_3, P_4, P_5>, S>, Yes], where P_1,P_2, P_3, P_4, P_5 are the positions of the sentences already selected in a summary and S is the sentence for which the model predicts summary worthy (yes) or not (summary unworthy). The objective of designing training examples in this manner is to train the model to respond positively when the current sentence S is important and contains information that is sufficiently diverse compared to the sentences that have been previously selected in the summary.

According to the above format of positive training example, there are five positions in the summary for the already selected sentences and S is the sentence under consideration for inclusion in the summary when some or none of these five positions are filled with summary sentences. If S is suitable for being selected in the summary, the training example is labeled as 'Positive'. When summary generation starts, we assume that all the previous five positions are occupied by NULL sentences. However, assuming that all the previous five positions are already occupied by five summary sentences which can be NULL or non-NULL or mixed, we can generate all possible positive training examples in the above-

Algorithm 4 Algorithm for generating positive training examples

Input: A set of document-oracle extract (d, OE) pairs.

Format of a positive example: $[<< P_1, P_2, P_3, P_4, P_5 >, S >, Yes]$, where P_1, P_2, P_3, P_4, P_5 are the first five positions in the summary and S is the sentence which is under consideration for selection when P_1, P_2, P_3, P_4, P_5 are filled with the already selected sentences or NULL sentences .

Output: T_{pos}, a set of positive examples

```
1: $T_{pos} = ()$
2: for each training document- oracle extract pair(d,OE) do
3:     for $k = 1$ to 6 do                    ▷ The summary length is set to 6 sentences
4:         for each possible way of choosing $k$ sentences from
           the oracle extract OE do
5:             Name the chosen sentences as: $S_{e_1}, S_{e_2}, .., S_{e_k}$
6:             for $j = 1$ to $k - 1$ do
7:                 $P_j = S_{e_j}$        ▷ Fill the position $P_1$ with $S_{e_1}$, $P_2$ with $S_{e_2}$, so on
8:             end for
9:             for $i = k$ to 5 do        ▷ There are 5 positions in the Summary-so-far
10:                $P_i = NULL$           ▷ Fill the remaining positions with NULL sentences
11:            end for
12:            set $S = S_{e_k}$.
13:            Label this instance $EX$ as "Yes".
14:            $T_{pos}$=$T_{pos} \cup EX$
15:        end for
16:    end for
17: end for
18: return $T_{pos}$
```

mentioned format using Algorithm 4.

Algorithm 4 accepts a set of document-extract(oracle) pairs created from the training dataset using Algorithm 3 and produces positive training examples in the following format.
[<<P_1, P_2, P_3, P_4, P_5>, S>, Yes], where P_1, P_2, P_3, P_4, P_5 are considered as the positions which are occupied by some or none of the summary sentences and S is the sentence considered for inclusion in the summary.

We assume that all positions P_1, P_2, P_3, P_4, P_5 are initially occupied by NULL sentences and a positive training example generated by the algorithm for k=1 is:[<<NULL, NULL, NULL, NULL, NULL>,S_e>, Yes], where S_e is a summary sentence chosen from the training extract for a document. This type of positive training example represents the situation where the Summary-so-far is empty and the model needs to predict whether S_e is a summary sentence or not. Since S_e is a sentence chosen from the oracle extract, this training example is labeled 'Yes,' that is, by this training example, the model is trained to predict 'Yes' in this situation. Since we can choose S_e in ${}^6C_1$ possible ways from the oracle extract of six summary sentences, we can have ${}^6C_1$ positive examples of this kind.

Now for k=2, the algorithm generates positive training examples by choosing a pair of sentences $< S_{e_1}, S_{e_2} >$ from the oracle extract and this type of training example represents the situations where the Summary-so-far contains S_{e_1} in the position P_1 and S is set to S_{e_2}. Since S_{e_2} is a legitimate summary sentence, using this training example, we want to teach the learning model to predict 'Yes' in this situation. The kind of positive training example looks like the following: [<<S_{e_1}, NULL, NULL, NULL, NULL >,S_{e_2}>,yes]. Therefore, for k=2, we can have the second type of positive examples and the number of positive examples of this type is ${}^6C_2$. Thus, the different types of positive example are generated for the different values of k, which vary from 1 to 6. Thus, the total number of positive examples generated by Algorithm4 for each document-oracle extract pair is:
$({}^6C_1 + {}^6C_2 + {}^6C_3 + {}^6C_4 + {}^6C_5 + {}^6C_6) = 63$.

Since the output of the learning model at each time step can be either "Yes" (summary worthy) or "No" (summary unworthy), we also need to have negative examples to make our model robust. Each negative example represents the situation where the model should predict 'No'. The format of the negative example is similar to that of the positive example, but its generation process is different. A negative example looks like the following: [<<P_1, P_2, P_3, P_4, P_5>, S>, No], where P_1, P_2, P_3, P_4, P_5 are the positions that are occupied by some or none of the summary sentences in the previous time steps and S is any summary unworthy sentence chosen from the training document at the current time step. When S is a summary unworthy sentence and Summary-so-far is empty, we can create the first type of negative training example in the following format: [<< NULL, NULL, NULL, NULL, NULL>, S>, No]. This type of negative instance is designed to enable the learning model to predict 'No' when the Summary-so-far is empty and S is a summary unworthy sentence.

We consider that the sentences that do not belong to the training extract are summary unworthy sentences for the corresponding training document. The training extract is usually very short compared to the size of the corresponding document, and most sentences in the document are summary unworthy sentences. If all these summary-unworthy sentences are taken for generating negative examples, because this would lead to a large number of negative examples and make the training data imbalanced.

To alleviate the data imbalance problem, summary unworthy sentences are carefully chosen from the training document for generating negative examples. We assume that the summary unworthy sentences appearing near the summary worthy sentences in the training document can be more useful for generating negative examples because this makes the model more robust. Here the intuition is that choosing the negative examples from the neighborhood of the positive examples can help in detecting minute differences between the positive and negative examples. Therefore, we choose summary-unworthy sentences from within the window of size 3, where a summary-worthy sentence is at the center of the window. For example, if S_k is any sentence chosen from the oracle extract, which appears in the training document at position k and if its neighboring sentences S_{k-1} and S_{k+1} do not occur in the oracle extract, S_{k-1} and S_{k+1} are considered to generate negative examples. Since we have six summary-worthy sentences in a training extract, we choose at most $6 * 2$ =12 summary-unworthy sentences from each training document to generate negative examples.

The procedure for generating negative examples is given in Algorithm 5. Algorithm 5 generates negative examples when a document and its oracle extract are submitted to it. For generating all negative examples for our entire training dataset we need to apply Algorithm 5 to each of the document-oracle extract pairs obtained from our training dataset.

In the first part of Algorithm 5, a pool of summary unworthy sentences is created. With this pool of sentences, the algorithm initially generates the negative examples in ${}^{12}C_1$ ways. Here each way gives a negative example representing the situation where no summary-worthy sentence is previously selected and one summary unworthy sentence is submitted to the model for consideration. For k=1, the algorithm generates the negative examples representing the situations where the Summary-so-far contains one summary-worthy sentence and the sentence S which is under consideration for selection is summary-unworthy. In this situation, the model's output should be "No". We can generate the second type of negative examples in ${}^{6}C_1 * {}^{12}C_1$ ways because we can choose one summary-worthy sentence from the oracle extract in ${}^{6}C_1$ ways and, for each of the ${}^{6}C_1$ possible ways, we can choose a summary unworthy sentence in ${}^{12}C_1$ ways. Thus, for the different values of k, Algorithm5 generates the different types of negative examples representing the different situations that the learning model will face in the future. Thus, for each pair of document-oracle extracts, the total number of negative examples that can be generated using Algorithm5 is at most
$[{}^{12}C_1 + {}^{6}C_1 * {}^{12}C_1 + {}^{6}C_2 * {}^{12}C_1 + {}^{6}C_3 * {}^{12}C_1 + {}^{6}C_4 * {}^{12}C_1 + {}^{6}C_5 * {}^{12}C_1]$ =756. For some cases, where the training document size is small, we can have less than 756 negative examples also. So, we can have at most 756 negative training examples for each document-extract pair.

Algorithm 5 Algorithm for generating negative training examples

Input: A document-oracle extract pair(d,OE) where d is a sequence of sentences, d: $[S_1, S_2, .., S_n]$ and OE is its oracle extract.
Format of a negative example: $[<< P_1, P_2, P_3, P_4, P_5 >, S >, No]$, where P_1, P_2, P_3, P_4, P_5 are the first five positions in the summary, which are filled with the already selected sentences or NULL sentences and S is any summary unworthy sentence chosen from the training document.
Output: T_{neg}, a set of negative examples

```
1: POOL= ( )                          ▷ creating a pool of summary unworthy sentences
2: for each sentence $S_{OE}$ ∈ the oracle extract OE do
3:     Search $S_{OE}$ into the document d
4:     k = position of $S_{OE}$ in d
5:     if $S_{k-1} and S_{k+1} \notin$ OE then
6:         $POOL = POOL \cup S_{k-1} \cup S_{k+1}$
7:     end if
8: end for ▷ Generating negative examples where Summary-so-far ▷ contains
   only NULL sentences and S is summary unworthy
9: $T_{neg} = ()$
10: for $j = 1$ to 5 do
11:     $P_j = NULL$          ▷ Initially, Summary-so-far contains five NULL sentences
12: end for
13: for each sentence $S_p$ ∈ POOL do
14:     Generate a negative example $EX$ by setting $S = S_p$.
15:     Label this instance $EX$ as "No".
16:     $T_{neg} = T_{neg} \cup EX$
17: end for ▷ Generating negative examples where Summary-so-far ▷ contains at
    least one summary sentence and S is summary unworthy
18: for $k = 1$ to 5 do             ▷ The Summary-so-far has positions for 5 sentences
19:     for each possible way of choosing $k$ sentences from OE do
20:         Name the chosen sentences as $S_{e_1}, S_{e_2}, .., S_{e_k}$
21:         for $j = 1$ to $k$ do
22:             $P_j = S_{e_j}$          ▷ Fill the position $P_1$ with $S_{e_1}$, $P_2$ with $S_{e_2}$, so on
23:         end for
24:         for $i = k + 1$ to 5 do
25:             $P_i = NULL$        ▷ Fill the remaining positions with NULL sentences
26:         end for
27:         for each sentence $S_p$ ∈ POOL do
28:             Generate a negative example $EX$ by setting $S = S_p$.
29:             Label this instance $EX$ as "No".
30:             $T_{neg} = T_{neg} \cup EX$
31:         end for
32:     end for
33: end for
34: return $T_{neg}$
```

3.2.1.2 Transforming training examples to numerical vectors

Since neural networks cannot understand natural text, the format of the training examples discussed above cannot be understood by the neural networks. Therefore, we need to transform them into numerical vectors. The input to the neural model is the concatenation of two vectors (1) SFV - saliency feature vector, which is computed based on the features of an individual sentence, and (2) RV - relation vector, which is computed based on the relation of the current sentence with the sentences which were previously selected in the summary. We have presented a fixed-size format for the representation of a training example. This fixed format is necessary to make the RV vector equal for each training example. Since RV gets dynamically new values and its size depends on the number of sentences previously selected in the summary. With the help of the fixed format of the training examples, we ensure that all RVs are of equal lengths. The relationship between two sentences is measured by a similarity value obtained using a semantic similarity measure discussed later in this section.

We consider nine saliency features for defining the saliency of a sentence. First, we discuss below the saliency features based on which the saliency feature vector SFV is computed.

Sentence centrality[54]. This feature is a variant of the degree feature presented in [54]. The degree of a sentence representing a vertex in the similarity graph corresponding to a document represents the prestige of the sentence in the collection of sentences. Here, a document is represented as a similarity graph where each node corresponds to a sentence, and the weight of an edge between two sentences is calculated using the cosine similarity between the sentence vectors for the corresponding sentences. Here, a sentence is represented using a TF*IDF-based vector. The work presented in [54] considers the similarity graph in which an edge between two nodes is constructed if the weight of the edge is greater than a predefined threshold value. The degree of a node that corresponds to a sentence is calculated by counting the number of other nodes to which the node is connected.

Unlike the degree feature presented in [54] that counts the number of connected nodes, we define the soft degree feature whose value is calculated by summing up the weights of the edges by which the sentence under consideration is connected with the other sentences in the graph. In our case, the sentence vector is an embedding vector that is obtained by taking the average of the Glove word vectors[154] for the words (stop words removed) contained in the sentence. Since the dimension of the word embedding vector for the sentence S is set to 50, the dimension of the sentence vector is also 50. The sentence vector is calculated using Equation 3.1.

$$WordEmdeddingVec(S) = \frac{1}{||S||} \sum_{W \in S} Glove(W) \tag{3.1}$$

where W∈ S and W is not a stop word
Glove(W)=Glove word vectors[154]

An example graph with four sentences is shown in Figure 3.4 and the adjacency matrix representation for the graph is shown in Figure 3.5. The rightmost column of the matrix is the values of soft degree for the sentences in the graph. For example, the value for the soft degree feature for the sentence S_1 can be computed by taking the sum of the first row of the adjacency matrix.

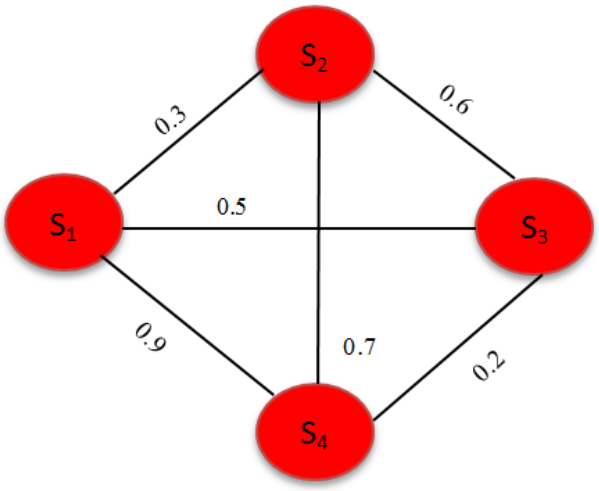


Figure 3.4: Sample similarity graph

$$\begin{array}{c|ccccc} & S_1 & S_2 & S_3 & S_4 & \text{Soft degree value} \\ \hline S_1 & 0 & 0.3 & 0.5 & 0.9 & 1.7 \\ S_2 & 0.3 & 0 & 0.6 & 0.7 & 1.6 \\ S_3 & 0.5 & 0.6 & 0 & 0.2 & 1.3 \\ S_4 & 0.9 & 0.7 & 0.2 & 0 & 1.8 \end{array}$$

Figure 3.5: Sample similarity matrix

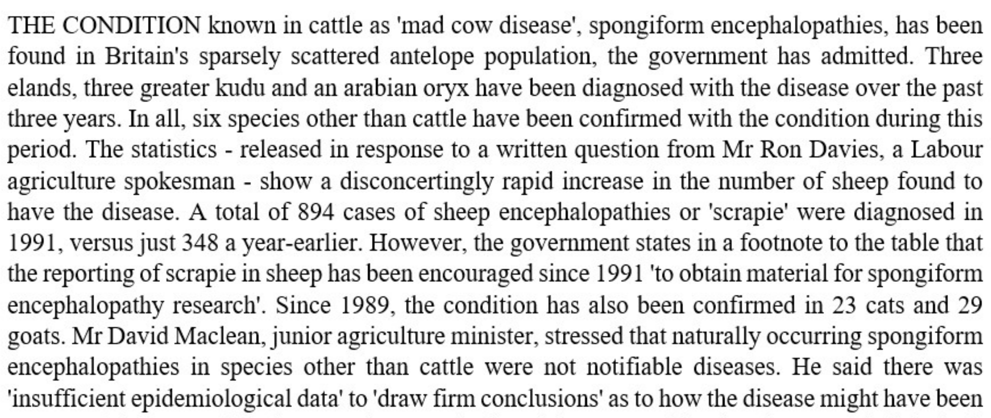
This feature measures the centrality of the sentence. From a summarization perspective, the centrality feature helps in determining the saliency of the sentence. The centrality feature value is normalized by dividing its value by the maximum soft degree value for the similarity graph.

TF-IDF[54]: To calculate the value of this feature for a sentence, TF*IDF values for the words (stop words are not considered) contained in the sentence are summed up. TF(Term Frequency) is calculated as the number of times a particular word appeared in the document. Inverse Document Frequency(IDF) for a word is a measure of how rare a word is. It is the logarithmic value of the ratio of the total number of documents in the corpus and the number of documents that contain the word at least once. TF-IDF weight for a sentence S is computed as follows:

$$Weight(S) = \sum_{w \in S} TF - IDF(w) \tag{3.2}$$

where TF(w)=number of times a term w appears in a document
IDF(w)=log $N/df(w)$
where N is the total number of documents and df(w) is the number of documents with the term w in it.

This feature measures the information richness of the sentence. The feature is normalized using the max-min procedure.
Sentence length: This feature is defined by the number of words(stop words are removed) contained in a given sentence. Since our task is to generate a short summary from a document, this feature helps to eliminate too short or too long sentences. It is normalized by dividing its value by the length of the longest sentence in the document. For example, if the input sentence is ”Jagadish Chandra Bose is a great scientist in India.”, the sentence obtained after stop-word removal (Stop words are the unimportant low-content words. We have used the stop-word list available in the NLTK toolkit.) is ”Jagadish Chandra Bose great scientist India.”. The length of this sentence after the removal of the stop words is 6 because it contains 6 words.
Distance from the beginning of the document: The position of a given sentence



THE CONDITION known in cattle as 'mad cow disease', spongiform encephalopathies, has been found in Britain's sparsely scattered antelope population, the government has admitted. Three elands, three greater kudu and an arabian oryx have been diagnosed with the disease over the past three years. In all, six species other than cattle have been confirmed with the condition during this period. The statistics - released in response to a written question from Mr Ron Davies, a Labour agriculture spokesman - show a disconcertingly rapid increase in the number of sheep found to have the disease. A total of 894 cases of sheep encephalopathies or 'scrapie' were diagnosed in 1991, versus just 348 a year-earlier. However, the government states in a footnote to the table that the reporting of scrapie in sheep has been encouraged since 1991 'to obtain material for spongiform encephalopathy research'. Since 1989, the condition has also been confirmed in 23 cats and 29 goats. Mr David Maclean, junior agriculture minister, stressed that naturally occurring spongiform encephalopathies in species other than cattle were not notifiable diseases. He said there was 'insufficient epidemiological data' to 'draw firm conclusions' as to how the disease might have been contracted in cases other than scrapie. Scrapie, he said, was considered to be transmissible 'both maternally and horizontally'.

Figure 3.6: A Sample input document

in the document is taken as the distance of the sentence from the beginning of the document. The first sentence of the document has a position value of 1, the second sentence has a position value of 2, and so on. This feature is useful for summarizing news documents because the sentences appearing earlier in the news document are more important due to the journalistic way of news reporting. It is normalized by dividing its value by the document length (number of sentences in the document). For example, the sample document shown in Figure 3.6 contains 10 sentences. The final normalized value of this feature for the first sentence of the document is:

$$\frac{position\ of\ the\ sentence\ in\ the\ document}{document\ length} = \frac{1}{10} = 0.1$$

Similarly, the normalized value of this feature for the second sentence is
$\frac{2}{10} = 0.2$

Distance from the end of the document: The distance of a given sentence from the end of the document is the difference between the position of the given sentence and the position of the last sentence in the document. This feature is defined to give attention to the salient sentences occurring in the conclusion paragraph. The feature is normalized by dividing its value by the document length (number of sentences in the document).

For example, the normalized value of this feature for the first sentence in the document shown in Figure 3.6 is:
$\frac{\text{Distance from the end}}{\text{Document Length}} = \frac{10-1}{10} = 0.9$. Here, the distance of the first sentence from the end of the document is 10 Similarly, the normalized value of this feature for the second sentence of the document is $\frac{10-2}{10} = 0.8$

Similarity with the first sentence: It is defined by the cosine similarity between a given sentence and the first sentence of the document. Here a sentence is represented by the sentence embedding obtained by Equation 3.1. Since the leading sentences of a news document are important, this feature is defined to give attention to the sentences which are linked to the lead sentence. Equation 3.3 is used to compute the similarity of a given sentence S with the first sentence S_1.

$$cosine - similarity(S1, S) = \frac{V(S1).V(S)}{||V(S1)||.||V(S)||} \tag{3.3}$$

Where: V(S) is the embedding vector for the sentence S. V(S) is obtained by taking the average of the vectors of content words (stop words are not considered) in sentence S. For a given sentence S, V(S) is calculated using Equation 3.3.

Similarity with the second sentence: It is defined by the cosine similarity between a given sentence and the second sentence of the document. Here a sentence is also represented by the sentence embedding vector obtained by Equation

3.1and the cosine similarity between two sentences is computed using Equation 3.3. When the first sentence is too short, this feature can be useful.

Similarity with headline: It is defined by the cosine similarity between a given sentence and the headline of the document. Here both the sentence and the headline are represented by the embedding vectors obtained by Equation 3.1 and the cosine similarity between these two vectors is computed using Equation 3.2. If the document does not have any headline, this feature value is set to 0.
The headline is a very important piece of information for the document. If the headline is present, this feature favors the sentences which are highly linked to the headline.
. **Parts of proper noun**: The value of this feature is computed by counting the number of words that are parts of proper nouns occurring in a sentence. We identified proper nouns by using the Standford POS tagger available in the NLTK toolkit. For example, the sentence ″Mr David Maclean, junior agriculture minister, stressed that naturally occurring spongiform encephalopathies in species other than cattle were not notifiable diseases.″ is tagged as follows:
″ [(′Mr′, ′NNP′), (′David′, ′NNP′), (′Maclean′, ′NNP′), (′,′, ′,′), (′junior′, ′JJ′), (′agriculture′, ′NN′), (′minister′, ′NN′), (′,′, ′,′), (′stressed′, ′VBD′), (′that′, ′IN′), (′naturally′, ′RB′), (′occurring′, ′VBG′), (′spongiform′, ′NN′), (′encephalopathies′, ′NNS′), (′in′, ′IN′), (′species′, ′NNS′), (′other′, ′JJ′), (′than′, ′IN′), (′cattle′, ′NNS′), (′were′, ′VBD′), (′not′, ′RB′), (′notifiable′, ′JJ′), (′diseases′, ′NNS′), (′.′, ′.′)]″

In the tagged sentence given above, the tag ″NNP″ indicates a proper noun. We have identified proper nouns by checking whether the words are tagged as ″NNP″. Thus, the words ′Mr′, ′David′, and ′Maclean′ are the proper nouns. Therefore, the number of proper nouns in the above sentence is 3.
This feature is normalized by dividing its value by the maximum number of proper nouns that a sentence in the input document can contain. To find the maximum number of proper nouns, we count the proper nouns in each sentence of the input document and find the maximum.
We have observed that proper names are important entities for the political news document because the political news document is usually centered around political personnel. This feature is defined to give attention to sentences that contain person names.

The algorithms for obtaining a numerical vector corresponding to a training example are shown in algorithm 6 and algorithm 7. A training example in the format: [<<P_1, P_2, P_3, P_4, P_5>, S>, Class] is converted into a numerical vector using these algorithms. Here, $Class$ can be ′Yes′ or ′No′. The process given in Algorithm 6 finds FV which is the final numerical vector representation for a

Algorithm 6 :FeatureVectorCreation ([<<P_1, P_2, P_3, P_4, P_5>, S>, Class])

Input: The training example, [<<P_1, P_2, P_3, P_4, P_5>, S>, Class], where P_1, P_2, P_3, P_4, P_5 represent the sentence position indices in Summary-so-far that can contain summary worthy sentences or NULL sentences or both and S is a candidate sentence which can be summary worthy or summary unworthy and $Class \in [Yes, No]$

Output: Final labeled feature vector $< FV, Class >$, where FV is the numerical vector for a training example [<<P_1, P_2, P_3, P_4, P_5>, S>, Class]

```
1: Calculate the saliency feature values for S using the saliency feature set.
2: Represent the saliency feature values in a form of vector SFV(S)=[$a_1$,$a_2$,..,$a_n$],
   where $a_i$ is the value for the $i^{th}$ feature of S and $n$ is the number of saliency
   features.
3: RV(S)= RVec(S, $< P_1, P_2, P_3, P_4, P_5 >$)      ▷ The function RVec given
                                                  in Algorithm 7 returns a feature
                                                  vector representing the summary
                                                  diversity when S is to be included
4:                  ▷ Concatenate SFV(S) and RV(S) to form FV for the example
5: FV=[SFV(S), RV(S)]
6: Return $< FV, Class >$
```

Algorithm 7 : RVec($S, < P_1, P_2, P_3, P_4, P_5 >$)(Computing feature vector(relation vector) representing summary diversity when S is considered to be included in the summary)

Input: P_1, P_2, P_3, P_4, P_5 are the five positions in Summary-so-far that can contain summary-worthy sentences or NULL sentences or both and S is a candidate sentence to be included in the summary.

Output: Relation vector RV(S) representing the summary diversity when S is considered to be included in the summary.

```
1: $RV(S) = ()$
2:            ▷ Find semantic similarity of S with the sentence at each position in
   Summary-so-far
3: for $j = 1$ to 5 do
4:     $S_j$ = Sentence at position $P_j$
5:     $b_j$ = cosine(WordEmdeddingVec(S),WordEmdeddingVec($S_j$))
6:     $RV(S)$ =RV(S).append($b_j$))
7: end for
8: Return RV(S)
```

training example. FV is obtained by concatenating the saliency feature vector(SFV) with the relation vector RV representing summary diversity. RV is computed based on the semantic relations of S with the already selected sentences (Summary-so-far). The semantic relation between two sentences is computed based on cosine similarity between embedding vectors for the sentences. The embedding vector for a non-NULL sentence S, WordEmdeddingVec(S) is obtained by taking the average of the Glove word vectors[154] for the words (stop words removed) contained in the sentence. It is calculated using Equation 3.1. Initially, at the time step t = 0, Summary-so-far contains five NULL sentences, and the word embedding vector for each NULL sentence is taken as a zero vector, that is, in this case, WordEmdeddingVec(NULL) returns a zero vector. The dimension of the word embedding vector for sentence S is 50 since we use the 50-dimensional Glove vector for each word. For the same reason, the dimension of the vector for a NULL sentence is set to 50, that is, a NULL sentence is represented by a 50-dimensional zero vector.

When the algorithm 6 for the creation of a feature vector is invoked with a training example, the saliency feature vector SFV is computed first and then RV is computed in the subsequent steps of the algorithm 7. When a summary sentence is selected by the model, the NULL sentence at the position P_1 in Summary-so-far is replaced by the selected summary sentence. Gradually, the positions in Summary-so-far are occupied by the summary sentences. For example, with an input example pattern $<< S_1, NULL, NULL, NULL, NULL >, S_2 >$, we mean that the sentence S1 is already in Summary-so-far, each unfilled position in Summary-so-far is filled with a NULL sentence and the sentence S2 will be checked whether it will be included in Summary-so-far or not. During training, such a pattern is labeled as "Positive" if S_2 is a sentence in a reference summary, otherwise, it is labeled as "Negative". However, to compute the relation vector RV(we also call it the diversity feature vector), S_2 is compared with S_1 and each NULL sentence in the above input example pattern. This results in the relation vector RV = [γ, 0, 0, 0, 0], where γ = cosine(WordEmdeddingVec(S_1), WordEmdeddingVec(S_2)) and a zero value in RV indicate that the cosine similarity between the embedding vector representing S_2 and the zero vector representing NULL sentence is 0.

With the two sets of features represented by SFV and RV as discussed above, we obtain an overall numerical representation of the feature vector FV for a training example which is fed to the learning model. We have considered nine (n = 9) hand-crafted saliency features for a sentence S and the five diversity features (m = 5) that correspond to the five different positions in Summary-so-far. Here, we have restricted the diversity features to five only because our objective is to generate a short summary from each document and this fixed format makes the diversity feature vector equal in length. Thus, the total dimension of the numerical vector

corresponding to each training example is (m+n)= (9+5) = 14. After converting each positive and each negative example to numerical vectors, the obtained feature vector FV looks like:
$(< x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8\ x_9\ f_1\ f_2\ f_3\ f_4\ f_5 >, Class)$, where $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\ x_9$ are the saliency feature values and f_1, f_2, f_3, f_4, f_5 are the diversity feature values and $Class$ can be either Yes or No.

Although the training dataset has been designed for a 6-sentence summary sentence task, the algorithms presented in this section can be easily ported to m-sentence summary generation task(m can vary from 1 to some user-specified value) with minor modifications.

3.2.2 Testing phase

During the testing phase, we create a summary incrementally by choosing sentences one by one from the document and submitting it to the learned model. Initially, when the first sentence is selected to create the summary from a test document, we generate the test example as follows:
[< NULL NULL NULL NULL NULL>, S_1], where S_1 is the first sentence of the test document. The test example is also converted to the numerical vector using the same procedure as in Algorithm 6. So, each test example corresponds to a feature vector: $< x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8\ x_9\ f_1\ f_2\ f_3\ f_4\ f_5 >$, which is submitted to the learned model. If the model says Yes, S_1 is added to the summary and the next test example becomes [< S_1 NULL NULL NULL NULL>,S_2], where S_2 is the second sentence of the test document. However, if this test example is labeled as No by the model, S_2 is not added. Thus, the learned model is invoked many times until the summary contains six sentences.

The detailed procedure of applying the model for summary generation is given in Figure 3.1. Although a six-sentence summary is generated for each test document using the proposed model, the first L words or b bytes are taken from each system-generated summary during summary evaluation. The value for L or b has been set according to guidelines for the respective reference datasets used in our experiments.

3.3 Experimental Results:

For our experiments, we used two different datasets-DUC dataset[3] and Daily Mail dataset[4] [155]. The DUC dataset comprises DUC2001 and DUC2002 datasets. The DUC2001 dataset is used as training data and the DUC 2002 dataset is kept for testing our model. In DUC 2001 (Document Understanding Conference 2001) and

[3]Document Understanding Conference: http://duc.nist.gov/
[4]https://github.com/JafferWilson/Process-Data-of-CNN-DailyMail

DUC 2002, the first task was to generate a summary of a length of 100 words from each document. The DUC2001 dataset contains 30 clusters, where each cluster contains a set of documents related to a topic. The DUC2001 dataset consists of a total of 309 English news articles. The DUC2002 dataset is available in the form of 59 news clusters containing 567 English news articles in total. Multiple reference summaries created by human judges are available for each article in both datasets. These reference summaries are used for the preparation of training sets and the evaluation of system summaries.

To prove the robustness and efficacy of our model, we have also tested our model on the second dataset called the Daily Mail dataset which is a large summarization dataset consisting of 170k articles in total. For our experiments on this dataset, it is randomly divided into three disjoint subsets: training set (90% of the whole document set), validation set(5% of the whole document set), and test set (5% of the entire document set). For each document in the Daily Mail dataset, there is a single summary with multiple sentences.

The number of documents, the average document length, the minimum document length, and the maximum document length for each dataset used in our experiments are shown in Table 3.2. In this case, the length of the document is the number of sentences contained in a document.

Table 3.2: Dataset descriptions

Dataset names	No. of documents	Average doc-len	Min doc-len	Max doc-len
DUC2001	309	110	32	780
DUC2002	567	108	26	701
Daily Mail	170K	34	12	175

Summary can be evaluated in two ways (1) using automatic evaluation tools, and (2) manual evaluation. Since manual evaluation of system-generated summaries is a laborious task, the common practice is to use automatic methods for summary evaluation. We have mainly used an automatic evaluation method for summary evaluation, but for only a small number of system-generated summaries, we have used a manual evaluation method which is discussed later in this section. For automatic summary evaluation, we have used the widely used automatic summary evaluation package called ROUGE [141] that measures summary quality based on the n-gram overlap between a summary generated by a system and a set of available reference summaries [40]. In addition to computing the word n-gram overlap, it also counts various kinds of other overlapping units between the system summary and the reference summaries.

We have used the latest version of the ROUGE package- ROUGE 1.5.5 for evaluating summaries. When the quality of a summary is evaluated using word n-gram overlap, the ROUGE toolkit reports a ROUGE–N score where N can be set to 1, 2, 3, and 4. The ROUGE toolkit is also used to measure skip bigram-based ROUGE scores. Among various types of ROUGE scores, the unigram-based ROUGE score (ROUGE-1) agrees most with human judgment[40]. Along with ROUGE-1 scores, many state-of-the-art summarization systems have been evaluated using ROUGE-2 (bigram-based) and ROUGE-SU4 (skip bigrams with skip distance up to 4 words [141]). So, we consider the ROUGE-1, ROUGE-2, and ROUGE-SU4 scores to evaluate our proposed summarization models. We run the ROUGE toolkit with stemming and without removing stop words. We set the summary length to 100 words by using the -l 100 option in the ROUGE toolkit, which takes the first 100 words from each system summary for evaluation. We report ROUGE recall scores to evaluate and compare our proposed neural summarization method with other existing summarization methods.

3.3.0.1 Performance of the proposed summarization models on DUC dataset:

For experimental purposes, we have developed two versions of our proposed model. In our work, we have used semantic sentence relations in two different contexts-(1) defining diversity features and (2) constructing a similarity graph which is used to compute a centrality score for a sentence. The centrality score is an important saliency feature that plays an important role in defining the saliency of a sentence. For finding semantic sentence relations, we have used word embedding-based sentence vectors and the cosine similarity measure. The two versions of our proposed model differ in the way sentence vectors are created using word embedding. We consider two methods for computing sentence vectors using word embedding. Equation 3.1 computes a sentence vector using the average rule which finds a sentence vector by taking the average of all Glove vectors for the words (stop-words removed) contained in the sentence. Another method for computing a sentence vector is to apply the product rule that finds the sentence vector by computing an element-wise product of all Glove vectors for the words(stop words removed) contained in the sentence. Based on these two methods for computing sentence vectors used to find sentence relations, we have developed two models: the first model uses the average rule, and the second model uses the product rule.

The performances of our developed two neural models on the DUC 2002 dataset have been shown in Table 3.3 where both the models are trained using the DUC2001 dataset. For best results, the following configurations for both models are used. The proposed model has four hidden layers and the number of hidden nodes in the first, second, third, and fourth hidden layers are seven, three, two, and two respectively. The ReLU activation function is used in the first three hidden layers, the sigmoid activation function is used at the fourth hidden layer, and the softmax activation function is used at the output layer. We have used Gradient Descent Optimizer for training the models. The learning rate is set to 0.05 and the number of epochs is set to 100.

We can see from Table 3.3, our proposed neural model with sentence representation using average rule outperforms the model with product rule over all three ROUGE metrics. The paired t-test shows that the performance differences between our best-performing model from the other model are all statistically significant (p-value<0.05).

Table 3.3: ROUGE scores for our proposed neural models when models are tested on DUC 2002 data

Proposed Neural Models	ROUGE-1	ROUGE-2	ROUGE-SU4
Model -1: Proposed neural model with sentence representation using **Average Rule**	**0.5145**	**0.2580**	**0.2730**
Model-2: Proposed neural model with sentence representation using Product Rule	0.4730	0.2154	0.2336

3.3.0.2 Performance of the proposed summarization models on the Daily Mail dataset

We have also tested our model on the Daily Mail dataset to prove its robustness. Since our neural model with sentence representation using the average rule performed best on the DUC dataset, we only applied this model to the Daily Mail dataset. This dataset has three parts: training set, validation set, and test set. For obtaining the best results on the validation set of the Daily Mail dataset, we have used the following configuration of the neural networks: four hidden layers and the number of hidden nodes in the first, second, third and fourth hidden layers are seven, three, two, and two, respectively. The ReLU activation function is used in the first three hidden layers, the sigmoid activation function is used in the fourth hidden layer, and the softmax activation function is used at the output layer. Here we have also used Gradient Descent Optimizer as an optimizer. The learning rate is set to 0.05 and the number of epochs is set to 100. The trained model is then tested on the test set for producing summaries which are collected for ROUGE evaluation. Since the best results are obtained with sentence representation using the Average Rule, we have shown the results obtained by this model in Table 3.4.

3.3.0.3 Comparison with existing summarization methods

Our system is compared with 14 summarization systems, of which some systems used supervised learning and deep learning-based approaches, some systems used

Table 3.4: ROUGE scores for our proposed neural models when models are tested on Daily Mail dataset

Model	ROUGE-1	ROUGE-2	ROUGE-SU4
Proposed Neural Model (Average Rule)	**0.4325**	**0.1740**	**0.1820**

unsupervised approaches, and some systems used graph-based approaches to extractive summarization. We have grouped below the state-of-the-art extractive summarization methods to which the results obtained by our proposed approach are compared.
• *Regression-based methods*: Two regression-based models presented in [156] are compared with our approach, one of them uses linear regression and the other model uses least median squared linear regression(LeastMedSq) for ranking sentences. A set of hand-crafted features is used to train the model.
• *Neural summarization models*: Three deep neural network models to which our proposed approach is compared are: [157],[77] and SummaRuNNer [151]. NetSum[157] uses neural network-based learning-to-rank algorithms to assign scores to sentences. The NN-SE system presented in [77] uses a recurrent convolutional network to extract sentences. In this model, CNN is used for sentence representation and the LSTM-based encoder-decoder model with an attention mechanism is used to generate an extractive summary. SummaRuNNer [151] is also an RNN-based summarization model that considers summary generation as a sequence learning task. This model starts to assign binary labels to sentences based on the current sentence and the labels of sentences encountered previously.
• *Graph-based methods*: UnifiedRank [158] uses a graph-based approach which is a novel approach for two summarization tasks: single document and multi-document summarization which are done simultaneously in a unified framework. This work investigates how one summarization task influences the other.
• *Cluster-based approaches*: COSUM [159] uses a K-means clustering algorithm for clustering sentences and a differential evolution-based optimization algorithm to select salience sentences from the clusters. The approach proposed in [160] is a cluster-based summarization approach that uses a sentence clustering algorithm based on differential evolution (DE). In this approach, a summary is created by assigning a score to each sentence in each cluster using a measure that considers the similarity of the sentence to other sentences within the cluster and the sentences outside the cluster. FEOM [161] is a summarization system that uses a sentence clustering-based approach to extractive text summarization. In this approach, a fuzzy evolutionary optimization-based method is used for sentence clustering and representative sentences are chosen from each cluster to create an extractive

summary.

• *Memetic algorithms*: MA-SingleDocSum [162] uses a memetic algorithm that combines the traditional evolutionary global search algorithm with the local search technique to find the best subset of sentences from the document to create an extractive summary. In this work, the fitness function for a possible solution is defined by a linear combination of a set of hand-crafted features. The model finds the best solution by maximizing this fitness function.
• *Conditional Random Fields(CRF)*: The CRF-based approach [163] treats the text summarization task as a sequence learning problem. In this approach, sentences in a document are labeled sequentially using 1 and 0.
• *Hidden Markov Models (HMM)*: QCS [65] is a summarization system that uses a hidden Markov model (HMM) that calculates the probability that each sentence is in the summary. To generate a single document extract for each document, the model chooses a subset of sentences with the highest probability.
• *Manifold Ranking* [58]: In this approach, the ranking score for sentences is obtained in the manifold-ranking process which assigns scores to the sentences based on the information richness of each sentence biased towards a given topic and information novelty which is measured by imposing diversity penalty for each sentence.
• *Peer System 28*: This system participated in the DUC 2002 summarization contest and performed the best on the single document summarization task. The peer summaries produced by this system are available with the DUC 2002 dataset. This system produces an extractive summary by extracting sentences using the HMM and the logistic regression model.

Table 3.5 presents the results obtained by our proposed best model and other state-of-the-art methods for the DUC 2002 dataset. From Table 3.5, it can be seen that our proposed neural approach outperforms all other methods in terms of ROUGE-1 and ROUGE-2 scores.

We have calculated the percentage improvement in the performance of our proposed approach over all other state-of-the-art methods using Equation 3.4 and we present the results in Table 3.6.

$$PI = \frac{M - N}{N} * 100 \tag{3.4}$$

where PI is denoted as Performance Improvement, M is noted as the score for the proposed approach, and N is defined as the score for other approaches.

As can be seen from Table 3.6, performance improvements achieved by our proposed approach compared to three other neural models namely Net Sum [157], NN-SE [77], SummaRuNNer [151] are respectively 14.43%, 8.54% and 13.33%, con-

Table 3.5: ROUGE comparison of our proposed neural approach with the existing methods on DUC 2002 data

Systems	Rouge-1	Rouge-2
Our Proposed Neural Model (Average Rule)	0.5145	0.2580
COSUM [159]	0.49083	0.23092
MA-SingleDocSum [162]	0.48280	0.22840
Peer system 28	0.4807	0.2288
FEOM [161]	0.46575	0.12490
Net Sum [157]	0.44963	0.11167
NN-SE [77]	0.47400	0.23200
SummaRuNNer [151]	0.45400	0.23900
Unified Rank [158]	0.48487	0.21462
Linear Regression [156]	0.49784	0.23924
LeastMedSq [156]	0.49824	0.23964
DE based Clustering Approach [160]	0.46694	0.12368
QCS [65]	0.44865	0.18766
CRF [163]	0.44006	0.10924
Manifold Ranking [58]	0.42325	0.10677

sidering ROUGE-1 scores. Comparison of our proposed neural approach with the latest work on neural text summarization [151] shows that our approach achieves an improvement of 13.33% and 7.95%, considering ROUGE-1 and ROUGE-2 scores. In summary, for the DUC 2002 dataset, our proposed approach achieves a minimum of 3.26% and 7.66% improvements over all existing methods in terms of ROUGE-1 and ROUGE-2 scores.

To show the efficacy of our proposed approach, we have tested our model on the Daily Mail dataset. Since the existing methods mentioned in Table 3.7 are not tested on the Daily Mail dataset and the results of the existing methods on the Daily Mail dataset are not available for comparison, we have compared the performance of our proposed approach on the Daily Mail dataset with LEAD-3 model described in [151], which simply takes the first three sentences of the document as a summary. This is called the lead baseline. A comparison of results is presented in Table 3.7. Here, the ROUGE score is calculated with a summary length of 275 bytes [151]. As can be seen in Table 3.7, our neural model outperforms the LEAD-3 model. After calculating improvement using Equation (3.4), we can see that our proposed model obtains a 7.58% and 17.40% improvement over the lead baseline in terms of the ROUGE-1 score and the ROUGE-2 score, respectively.

Table 3.6: Percentage improvement achieved by our proposed best model with respect to the existing methods

Systems	% improvement Rouge-1	% improvement Rouge-2
COSUM [159]	4.82	11.73
MA-SingleDocSum [162]	6.57	12.96
Peer system 28	7.03	12.76
FEOM [161]	10.47	106.57
NetSum [157]	14.43	131.04
NN-SE [77]	8.54	11.21
SummaRuNNer [151]	13.33	7.95
Unified Rank[158]	6.11	20.21
Linear Regression[156]	3.35	7.84
LeastMedSq[156]	3.26	7.66
DE based Clustering Approach [160]	10.19	108.60
QCS[65]	14.68	37.48
CRF [163]	16.92	136.18
Manifold Ranking [58]	21.56	141.64

Table 3.7: System comparison results on Daily Mail dataset. ROUGE score with summary length of 275 bytes

Systems	Rouge-1	Rouge-2
Proposed Neural Model (Average Rule)	**0.4325**	**0.1740**
LEAD-3	0.4020	0.1482

3.3.1 Manual evaluation of system generated summaries

We have considered 30 system-generated summaries for the 30 different documents randomly chosen from the DUC2002 dataset for manual evaluation. Since manual evaluation is laborious, we have restricted this evaluation to 30 summaries only. For each document, a system-generated summary and the corresponding reference summary were given to two human judges for manual testing. One judge was a student in the Master of Computer Science and Engineering course, and another judge was a Ph.D. research scholar. Each judge was asked to assign a score on a scale of 1 to 5 to each system-generated summary by comparing it with the corresponding reference summary. Score=1 indicates that the system-generated summary is very different from the reference summary and score = 5 indicates that they are completely similar.

The scores given by each judge for 30 system-generated summaries are averaged to obtain the average evaluation score assigned by each judge. Finally, to compute the overall manual evaluation score, the average evaluation scores assigned by the two different judges are further averaged. We have shown in Table 3.8 the average evaluation score assigned by each individual judge and the overall manual evaluation score.

Table 3.8: Human assessment by two judges

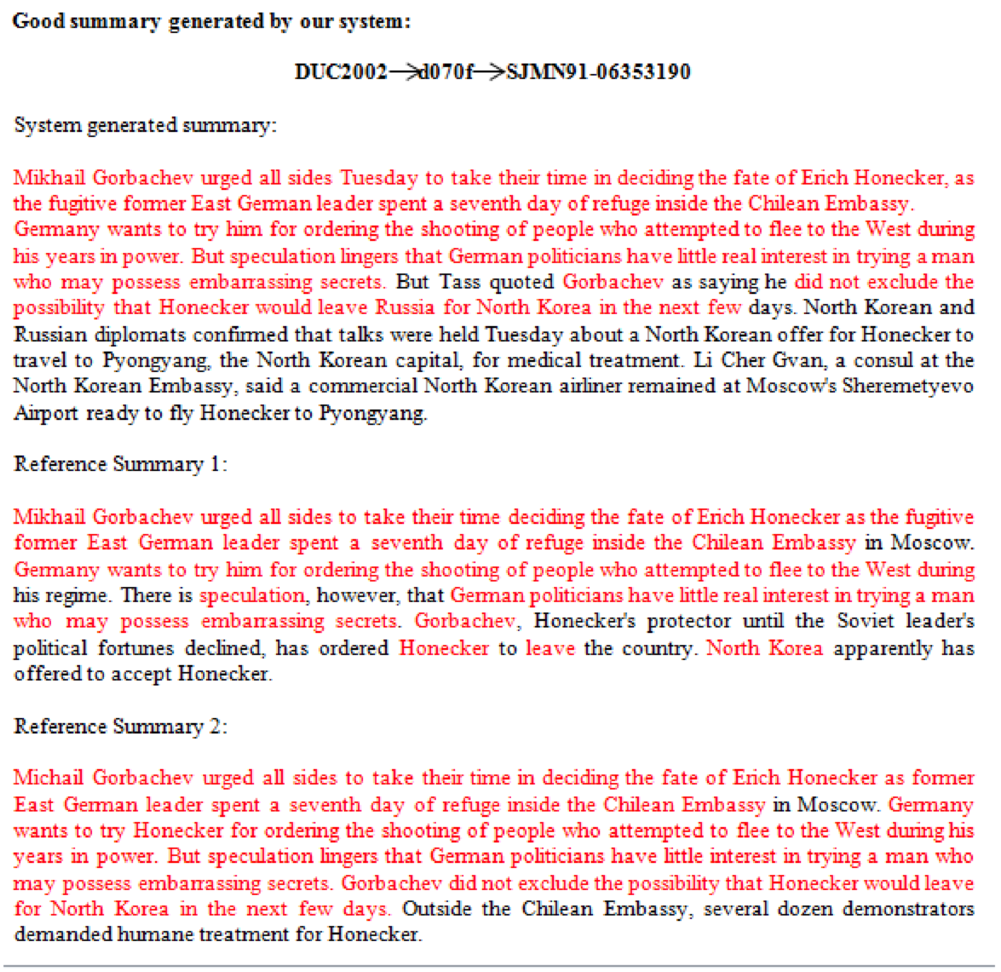
Average evaluation score assigned by Judge-1	Average evaluation score assigned by Judge-2	Overall manual evaluation score (Average of Judge-1 and Judge 2)
3.49	3.54	3.51

Table 3.8 shows that the overall manual evaluation score is 3.51, which indicates that the proposed neural text summarization method is effective in producing good extractive summaries.

3.3.2 Error analysis

We have shown in Figure 3.7 and Figure 3.8 respectively one good and one bad summary generated by our system. In the figures, the segments matching between the system summary and the reference summaries are colored red.

As we can see in Figure 3.7, the good summary generated by our proposed system has many sentence segments that match those appearing in both the reference summaries, reference summary 1 and reference summary 2 for the document id- SJMN91-06353190 in the folder - d070f of the DUC 2002 dataset. On the other hand, a bad system summary for the document id- LA121490-0148 under the folder d072f of the DUC2002 dataset is shown in Figure 3.8. It has sentence segments that appear either in reference summary 1 or in reference summary 2, but there are a few segments in this system-generated summary that are present in both reference summaries. This is why a low ROUGE score is found for this system-generated summary. This also demonstrates that the difference in views of human judges while creating the golden summary is also an important factor that affects the automatic summary evaluation process.



Good summary generated by our system:

DUC2002⟶d070f⟶SJMN91-06353190

System generated summary:

Mikhail Gorbachev urged all sides Tuesday to take their time in deciding the fate of Erich Honecker, as the fugitive former East German leader spent a seventh day of refuge inside the Chilean Embassy. Germany wants to try him for ordering the shooting of people who attempted to flee to the West during his years in power. But speculation lingers that German politicians have little real interest in trying a man who may possess embarrassing secrets. But Tass quoted Gorbachev as saying he did not exclude the possibility that Honecker would leave Russia for North Korea in the next few days. North Korean and Russian diplomats confirmed that talks were held Tuesday about a North Korean offer for Honecker to travel to Pyongyang, the North Korean capital, for medical treatment. Li Cher Gvan, a consul at the North Korean Embassy, said a commercial North Korean airliner remained at Moscow's Sheremetyevo Airport ready to fly Honecker to Pyongyang.

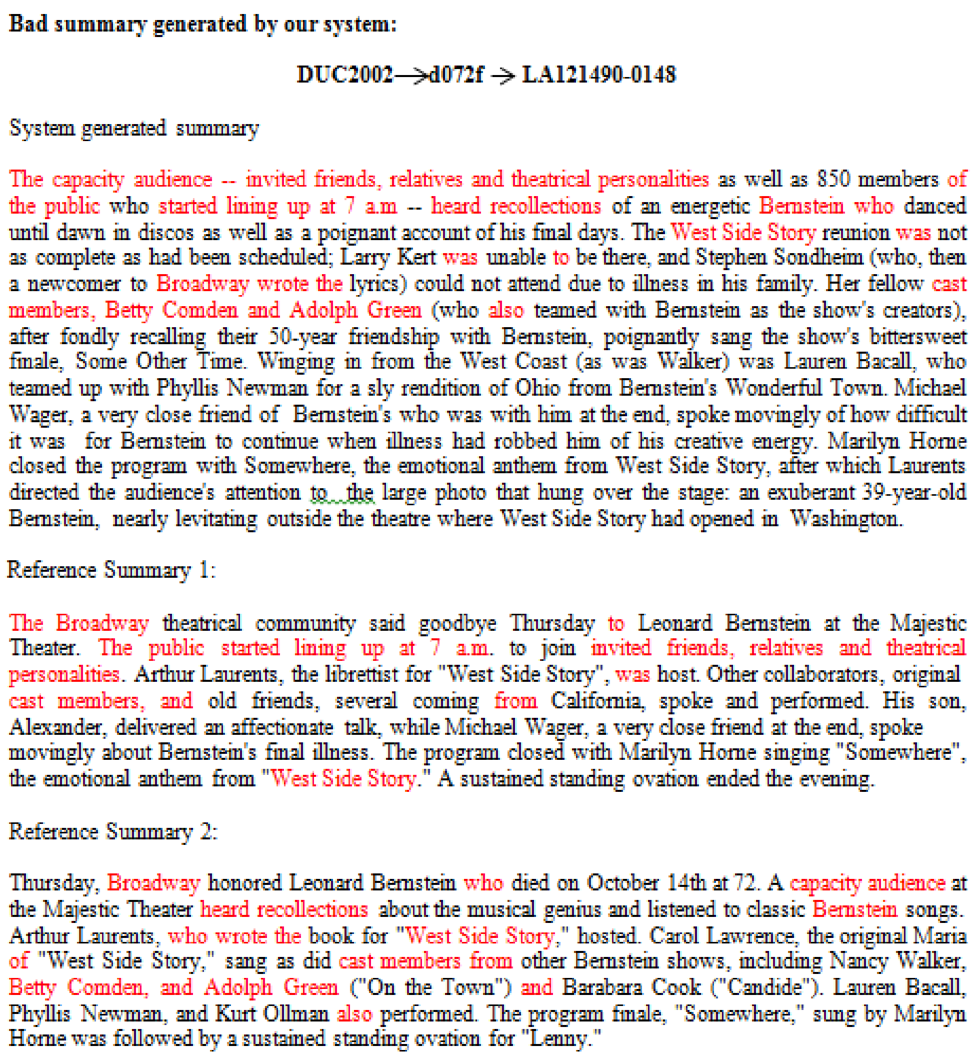
Reference Summary 1:

Mikhail Gorbachev urged all sides to take their time deciding the fate of Erich Honecker as the fugitive former East German leader spent a seventh day of refuge inside the Chilean Embassy in Moscow. Germany wants to try him for ordering the shooting of people who attempted to flee to the West during his regime. There is speculation, however, that German politicians have little real interest in trying a man who may possess embarrassing secrets. Gorbachev, Honecker's protector until the Soviet leader's political fortunes declined, has ordered Honecker to leave the country. North Korea apparently has offered to accept Honecker.

Reference Summary 2:

Michail Gorbachev urged all sides to take their time in deciding the fate of Erich Honecker as former East German leader spent a seventh day of refuge inside the Chilean Embassy in Moscow. Germany wants to try Honecker for ordering the shooting of people who attempted to flee to the West during his years in power. But speculation lingers that German politicians have little interest in trying a man who may possess embarrassing secrets. Gorbachev did not exclude the possibility that Honecker would leave for North Korea in the next few days. Outside the Chilean Embassy, several dozen demonstrators demanded humane treatment for Honecker.

Figure 3.7: An example of a good system-generated summary and two reference summaries for document SJMN91-06353190 under folder d070f of the DUC2002 dataset. Sentence segments that are common between the system summary and reference summaries are indicated in red color



Bad summary generated by our system:

DUC2002→d072f → LA121490-0148

System generated summary

The capacity audience -- invited friends, relatives and theatrical personalities as well as 850 members of the public who started lining up at 7 a.m -- heard recollections of an energetic Bernstein who danced until dawn in discos as well as a poignant account of his final days. The West Side Story reunion was not as complete as had been scheduled; Larry Kert was unable to be there, and Stephen Sondheim (who, then a newcomer to Broadway wrote the lyrics) could not attend due to illness in his family. Her fellow cast members, Betty Comden and Adolph Green (who also teamed with Bernstein as the show's creators), after fondly recalling their 50-year friendship with Bernstein, poignantly sang the show's bittersweet finale, Some Other Time. Winging in from the West Coast (as was Walker) was Lauren Bacall, who teamed up with Phyllis Newman for a sly rendition of Ohio from Bernstein's Wonderful Town. Michael Wager, a very close friend of Bernstein's who was with him at the end, spoke movingly of how difficult it was for Bernstein to continue when illness had robbed him of his creative energy. Marilyn Horne closed the program with Somewhere, the emotional anthem from West Side Story, after which Laurents directed the audience's attention to the large photo that hung over the stage: an exuberant 39-year-old Bernstein, nearly levitating outside the theatre where West Side Story had opened in Washington.

Reference Summary 1:

The Broadway theatrical community said goodbye Thursday to Leonard Bernstein at the Majestic Theater. The public started lining up at 7 a.m. to join invited friends, relatives and theatrical personalities. Arthur Laurents, the librettist for "West Side Story", was host. Other collaborators, original cast members, and old friends, several coming from California, spoke and performed. His son, Alexander, delivered an affectionate talk, while Michael Wager, a very close friend at the end, spoke movingly about Bernstein's final illness. The program closed with Marilyn Horne singing "Somewhere", the emotional anthem from "West Side Story." A sustained standing ovation ended the evening.

Reference Summary 2:

Thursday, Broadway honored Leonard Bernstein who died on October 14th at 72. A capacity audience at the Majestic Theater heard recollections about the musical genius and listened to classic Bernstein songs. Arthur Laurents, who wrote the book for "West Side Story," hosted. Carol Lawrence, the original Maria of "West Side Story," sang as did cast members from other Bernstein shows, including Nancy Walker, Betty Comden, and Adolph Green ("On the Town") and Barabara Cook ("Candide"). Lauren Bacall, Phyllis Newman, and Kurt Ollman also performed. The program finale, "Somewhere," sung by Marilyn Horne was followed by a sustained standing ovation for "Lenny."

Figure 3.8: An example of a bad system generated summary and two reference summaries for document LA121490-0148 under the folder d072f of the DUC2002 dataset. Sentence segments that are common between the system summary and reference summaries are indicated in red color.

3.4 Chapter Summary

In this chapter, a novel neural approach to single document text summarization is presented. In this approach, a deep feedforward neural network is trained to select a summary sentence in a way that trades off two factors, the saliency of the sentence, and how its inclusion in the summary improves summary diversity. Word embedding-based sentence representation and the cosine similarity measure have been used to find semantic sentence relations, which are used to compute diversity features as well as some saliency features. Training data preparation is an important part of this work. The suitable training data are carefully prepared to achieve the goal. Experimental results show that our proposed neural model with sentence representation using average word vectors performs better than many existing methods for the DUC 2002 dataset when we consider Rouge-1 and Rouge-2 measures, respectively, for summary evaluation.

4 Extractive Multi-Document Summarization Method Using Enhanced Similarity Measure

4.1 Introduction

Based on the input size, text summarization can be categorized as single document and multi-document text summarization. In case of single-document text summarization, only one document is taken as input to summarize, whereas multi-document text summarization aims to create summary by taking multiple related documents as input. For multi-document summarization, since the input consists of multiple related documents coming from various sources (e.g. news articles), redundancy is a critical issue, and we need to address this issue for creating multi-document summary. Multi-document text summarization captures both similarities and differences among collection of documents while generating a summary. The efficacy of an extractive multidocument summarization system is heavily dependent on the sentence similarity metric used to eliminate redundant sentences from the summary.

Many early approaches in the field of multidocument extractive text summarization include a centroid-based approach and graph-based approaches that used sentence similarity measures for redundancy removal and/or graph construction. Radev et al. used [15] a modified measure of cosine similarity that computes the similarity of a sentence to the pseudo centroid which is a set of significant words selected from the input. They used the geometric representation of sentences and the centroid in vectors but did not consider the semantic meaning of words. A graph-based approach developed by [15] considered a matrix of connections weights derived from cosine similarity between sentences (nodes) and used centrality-based salience to create a summary from multiple documents.

The methods proposed in [15] [164] [165] [166] [40] also used text summarization methods that involve the use of graphs. In these approaches, sentences obtained after breaking the input documents are considered as nodes in a graph. The weight of a connection between any pair of sentences is assigned by the similarity value between them. Assessing the significance of a sentence is done through a graph-based method that relies on both global and local information from the entire graph.

Our survey reveals that most existing graph-based methods predominantly employ the standard cosine similarity measure to construct the similarity graph. In addition to this, some other existing extractive multi-document summarization systems including centroid based system mentioned earlier also employ sentence similarity to either reduce redundancy, construct a graph, or both. However, the traditional cosine similarity measure uses sentence vectors that are obtained using the bag-of-words model that considers each term independently. This similarity measure cannot capture the actual similarity in many situations. For example, if two sentences have dissimilar terms but share one highly frequent term common to them, the similarity value might be high. If two sentences have similar meanings, but different word choices may have a lower cosine similarity value. If sentence lengths vary significantly, cosine similarity may not adequately reflect the actual similarity value. Longer sentences may have inherently lower cosine similarity scores, leading to biased results. Therefore, the traditional cosine similarity measure used in many early text summarization researches cannot capture the contextual and semantic relationships between terms. Motivated by this fact, we propose an enhanced sentence similarity measure that can capture semantic relationship between words and hypothesize that enhancing the similarity measure can lead to an improvement in the performance of graph-based summarization. The proposed similarity measure is a hybrid sentence similarity measure that combines a lexical similarity measure and a BERT-based semantic similarity measure.

In the first part of the chapter, we describe the proposed methodology. In this part, we discuss in detail the proposed similarity measure and the multi-document summarization method that uses the proposed similarity measure. In the second part of the chapter, we evaluate the performance of the summarization method to prove the effectiveness of the proposed similarity measure. In Section 4.3.2, a comparison of the summarization method with existing summarization methods is also discussed. Finally, a chapter summary is presented in Section 4.4.

4.2 Proposed Methodology

Erkan et al. proposed in [15] a graph-based degree centrality for computing sentence importance. In this approach, the input document set is converted into a collection of sentences, and the sentence collection is represented as a graph where each node corresponds to a sentence, and an edge between any two nodes is established if the lexical cosine similarity between the corresponding two sentences crosses a predefined cut-off value. In the degree-centrality-based approach, the degree of a node is considered as the salience score of the corresponding sentence. During summary generation, sentences are ranked based on their salience scores, and the top sentence is selected first in the summary. The next sentence is chosen from the ranked list and is included in the summary if it is sufficiently not similar to the previously selected sentences. In this work, sentence comparison is done using a lexical similarity measure for redundancy removal. One of the main drawbacks of the approach proposed by Erkan et al.[15] is that it does not use semantic similarity measures in graph construction and redundancy removal. To overcome this drawback, we propose a degree centrality method that uses a hybrid sentence similarity measure that combines the lexical sentence similarity used by Erkan et al.[15] and the BERT-based semantic similarity. The proposed hybrid sentence similarity measure is used in graph construction as well as redundancy removal. The redundancy plays a crucial role in multi-document summarization. In the subsequent subsections, we will present the first two similarity measures: Lexical and semantic.

4.2.1 Lexical Similarity

In this method, a sentence is represented as a TF-IDF vector whose length is equal to the vocabulary size, and the similarity between two sentences is computed as the cosine of the vectors for the corresponding two sentences. We call it lexical similarity because it breaks sentences into bags of words. The similarity between two sentences $S1$ and $S2$, represented by vector representations using $TF * IDF$ values, is calculated using the cosine similarity formula [15]:

$$LexSim(s_1, s_2) = \frac{\sum_{j=1}^{n} w_{1j} \cdot w_{2j}}{\sqrt{\sum_{j=1}^{n}(w_{1j})^2} \cdot \sqrt{\sum_{j=1}^{n}(w_{2j})^2}} \tag{4.1}$$

where S_1=$(w_{11}, w_{12}, w_{13}.........w_{1n})$ and
S_2=$(w_{21}, w_{22}, w_{23}.........w_{2n})$ which are TF-IDF based vector representation pf the sentences.
$w_{ij} = TF_{ij} * IDF_{ij}$
The $TF * IDF$ value for a word, denoted as w_{ij}, is calculated as the product of Term Frequency (TF) and Inverse Document Frequency (IDF). The TF is calculated using formula [15] [166]:

$$TF_{ij} = n(i, j) \tag{4.2}$$

Where $n(i, j)$ is the number of occurrences of the word i in sentence j.
The IDF is calculated as:

$$IDF_i = \log\left(\frac{N}{n_i}\right) \tag{4.3}$$

Where N is the total number of documents in a corpus and n_i is the number of documents containing the word i. We have set a threshold value of 3 for filtering out the noisy words. Words whose TF * IDF value is > 3 are considered when computing a sentence vector. Here, this similarity is noted as $LexSim$.

4.2.2 BERT based semantic similarity measure

BERT(Bidirectional Encoder Representations from Transformers) is based on the transformer architecture, which was introduced by [167]. Transformers have become a foundational architecture for various NLP tasks due to their effectiveness in capturing long-range dependencies in sequences. Unlike earlier sequence models processing text in one direction, BERT employs bidirectional processing, taking into account context from both left and right directions. We have obtained a sentence vector using the BERT encoder. The sentence vector obtained by the BERT encoder is 768 dimensional. The similarity between two sentences is computed using the cosine of the corresponding BERT vectors. Equation 4.4 is used to compute the semantic similarity based on BERT.

$$SemanticSim_{bert} = cosine(SVi_{bert}, SVj_{bert}) \tag{4.4}$$

In the above equation, SVi_bert and SVj_bert are two 768-dimensional vectors obtained from the BERT encoder for sentences i and j.

Equation 4.4 is a semantic similarity measure because we use a BERT encoder for sentence representation, which is a mapping of a sentence to a high-level abstract space.

4.2.3 Hybrid Similarity Measure

Although the measure of lexical similarity given in Equation 4.1 takes into account the relative importance of the terms in the input, it considers the bag-of-words model for sentence representation. Hence, the sentence vector becomes sparse. On the other hand, the semantic similarity measure given in 4.4 finds how much two sentences are contextually and semantically similar. In some cases, it is observed that the semantic similarity output is very high, although human finds that the concerned sentences have low semantic similarity. It may happen when sentences do not contain sufficient information, or one sentence is long and another sentence is short.

To generate a more precise measure of sentence similarity, we hybridize both similarity metrics, 1) lexical similarity and 2) BERT-based semantic similarity. We have performed hybridization using a blending parameter α shown in Equation 4.5. α is tuned to find the appropriate weights for the similarity measures which are combined.

$$Sim_{ij} = \alpha * Lexsim(i, j) + (1 - \alpha) * SemanticSim_{bert}(i, j) \tag{4.5}$$

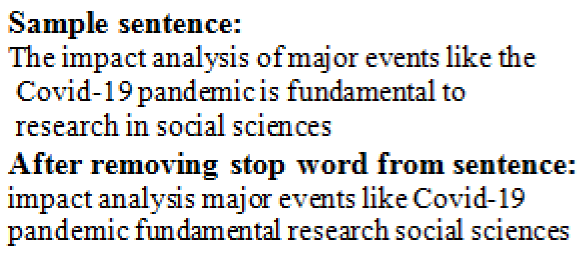
In Equation 4.5, Sim_{ij} = Similarity among i_{th} sentence and j_{th}sentence.
α=Blending parameter

4.2.4 Summarization method

The proposed summarization system undergoes several key stages: Pre-processing, Graph formation, Centrality computation, and summary creation.

4.2.4.1 Pre-processing

In this phase, sentences are delimited using a sentence tokenizer of the NLTK toolkit. The sentences are then broken into a collection of words. We have removed the stop words from the dataset. Stop words are common words that frequently occur in the dataset but are unimportant. NLTK toolkit was used to remove stop words from sentences. Figure 4.1 shows a sample sentence after the removal of stop words.



Sample sentence:
The impact analysis of major events like the Covid-19 pandemic is fundamental to research in social sciences
After removing stop word from sentence:
impact analysis major events like Covid-19 pandemic fundamental research social sciences

Figure 4.1: Removal of stop word

4.2.4.2 Graph Construction and Centrality Calculation

To compute degree centrality, we need to construct a graph in which each sentence corresponds to a node and establish an edge between two nodes if the hybrid similarity between the corresponding sentences is greater than a threshold. In our experiment, we get the best result by setting the threshold value to 0.6.

The graph is represented as an adjacency matrix that we call the sentence similarity matrix in which each cell (i,j) contains the value of the hybrid similarity (equation4.5) between sentence i and sentence j. Figure 4.2 illustrates a similarity graph with the pairs of sentences with similarities that exceed 0.1, 0.2, and 0.3 respectively. In the graph, $D_i s_j$ refers to the j-th sentence in the i-th document, and the different types of lines connecting nodes are used to indicate varying similarity values. The degree centrality of a sentence is defined as the degree of the corresponding node within the similarity graph. The degree centrality score of a sentence is regarded as the degree of the corresponding node in the similarity graph. Given that centrality represents the number of edges incident on a node, these scores vary from 1 to the total number of sentences in the set of documents. Consequently, normalization is necessary to constrain this value within a range of 0 to 1. To achieve this, we employ the Min-Max procedure as follows.

$$c_{score} = \frac{d - min}{max - min} \tag{4.6}$$

Where: c_{score} represents the normalized centrality score of the sentence, d denotes the degree of the sentence representing a node in the similarity graph), *min* stands for the minimum degree value in the graph, and *max* signifies the maximum degree value in the graph.

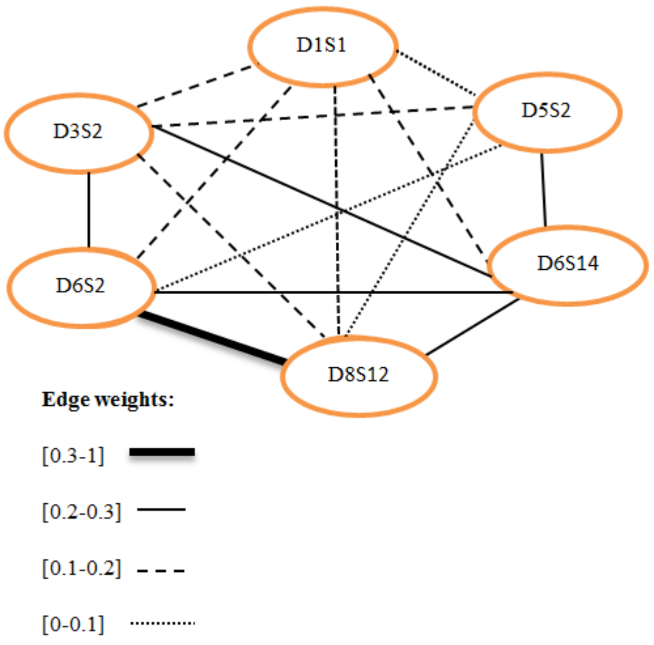


Figure 4.2: Weighted similarity graph

4.2.5 Sentence ranking and summary generation

To generate the summary from a set of documents, sentences are ranked based on their scores, where the score of a sentence is a linear combination of the normalized centrality score and the positional score of the sentence. We consider the position score because it is proven to be an effective feature in text summarization [168] [128].

The overall score for each sentence is derived through the linear combination of centrality score (c_{score}) and the position score. The positional score is calculated using Equation4.7. Sentences are ranked in decreasing order of their combined scores. To generate the nonredundant summary, the top-ranked sentence is selected first, and then the next sentence from the ranked list is selected in the summary if it is sufficiently dissimilar to the sentences already selected in the summary. This process is continued until the desired summary length is reached. We use a similarity threshold value to decide whether two sentences are similar or not. This threshold value is also tuned to find its optimal value.

$$positional - score = \frac{1}{\sqrt{i}} \tag{4.7}$$

where i is the position of a sentence in the document

4.3 Evaluation and Results:

We have evaluated our approach using an automatic summary evaluation package ROUGE which is widely used by many researchers. ROUGE [141] measures n-gram overlap between a system-generated summary and reference summaries [40]. ROUGE counts various kinds of unit that overlap between the system summary and the reference summaries. We have used the latest version of the ROUGE package named ROUGE 1.5.5 to evaluate the system summaries. The ROUGE toolkit reports various ROUGE–N scores, for example, ROUGE-1, ROUGE-2, etc. Along with ROUGE-1 scores, many state-of-the-art summarization systems have been evaluated using ROUGE-2 (bigram-based) and ROUGE-SU4 (skip bigrams with skip distance up to 4 words [141]. So, we consider the ROUGE-1, ROUGE-2, and ROUGE-SU4 scores to evaluate our proposed summarization models. We set the summary length to 665 bytes(100 words) as per DUC 2004 guidelines since we tested the proposed model on the DUC 2004 dataset. We use ROUGE-F score scores to evaluate and compare our proposed summarization method with other existing summarization methods.

4.3.1 Results:

We have used the DUC2004 dataset[1] for the evaluation of the proposed summarization model. The data set consists of 50 folders. Each folder contains approximately

[1]https://duc.nist.gov/duc2004/

10 documents which are news articles sourced from both the New York Times and Associated Press Wire services. Each article is accompanied by four reference summaries; The system summaries are compared with reference summaries using the ROUGE package for evaluating the model's performance.

Since the blending parameter α was used to find the weights for the similarity measures which are combined to have a hybrid similarity measure, it has an impact on the summarization output. To find the optimal value of α, we have varied the blending parameter α to obtain the best performance of the proposed model. The results with different values of α are shown in Figure 4.3.

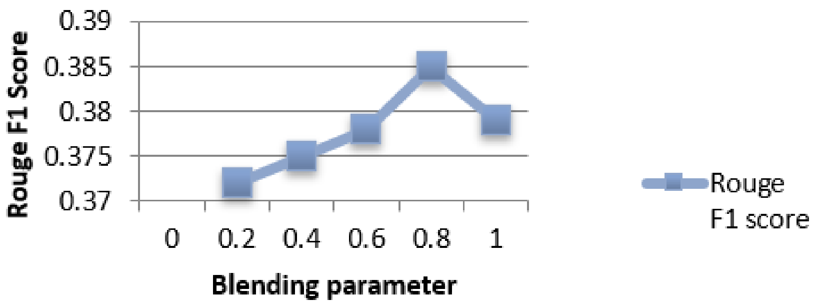


Figure 4.3: Impact on summarization performance while varying the values of the blending parameter,α

Table 4.1: The performance of the proposed summarization method(degree + position) with hybrid similarity measure on DUC2004 data. R-1 implies Rouge-1 F1 score, R-L implies Rouge-L F1 score and R-SU4 indicates Rouge-SU4 F1 score

Alpha(α)	**R-1**	**R-L**	**R-Su4**
0.4	0.3756	0.3461	0.1645
0.6	0.3789	0.3477	0.1659
0.8	**0.3855**	**0.3591**	**0.1797**

As we can see in Figure 4.3, the best result is obtained when the value of α is set to 0.8. To generate a non-redundant summary, we consider a similarity threshold as mentioned in the summary generation subjection. To assess the influence of this similarity threshold value used to minimize redundancy in the summary, we have varied it while keeping the blending parameter α fixed to the optimal value of 0.8. As shown in Figure 4.4, the best result is obtained when we set the similarity threshold at 0.6.

We have shown the performance of the proposed model in Table 4.1. Since the proposed work is an extension of the degree centrality method proposed by

Erkan et al. (2004) [15] who did not use the positional feature in sentence ranking, we have evaluated the proposed method without positional information and the results obtained are given in Table 4.2. In both tables, the best scores are indicated in bold font.

Table 4.2: The performance of the proposed summarization method (degree only) on DUC2004 data. R-1 implies Rouge-1 F1 score, R-L implies Rouge-L F1 score and R-SU4 indicates Rouge-SU4 F1 score

Alpha(α)	**R-1**	**R-L**	**R-Su4**
0.4	0.3607	0.3302	0.1459
0.6	0.3619	0.3316	0.1463
0.8	**0.3719**	**0.3348**	**0.1649**

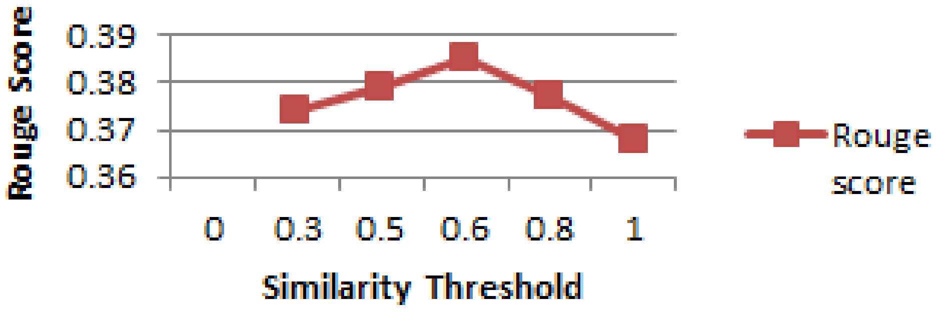


Figure 4.4: Similarity threshold vs Rouge F1 score

4.3.2 Comparisons with Existing methods

Since the proposed work is an extension of the degree centrality method proposed by Erkan et al. (2004) [15] who did not use positional feature in sentence ranking, we have evaluated our proposed method without positional information and compared the results obtained with those of the degree centrality method proposed by Erkan et al. [15]. In Table 4.3, for comparison, we have presented the results obtained by three models: the proposed method with only the degree centrality score, the proposed method (degree centrality score + positional score) and the degree centrality method with lexical similarity as proposed in the paper of Erkan et al. [15]. Since we have used the ROUGE F1 score for system evaluation whereas Erkan et al. used the ROUGE-1 recall score for system evaluation, we have implemented the degree centrality method with the lexical similarity measure as proposed in [15]. In table 4.3, we have shown the comparison of the proposed method with the degree centrality-based method proposed by Erkan et al. It is evident from table 4.3 that our proposed method with degree centrality feature performs significantly better than that of the degree centrality-based method proposed by Erkan et al. who used only lexical similarity to construct graph and redundancy removal whereas we used a hybrid similarity measure. This shows that the proposed hybrid similarity measure is effective. The table also shows that the performance of the proposed method (degree + position) improves 3.65% (=(((0.3855- 0.3719)/0.3719)*100)) over the proposed method with the degree feature only.

Table 4.3: Comparison with existing models that used Degree centrality score only.

Model	Rouge-1 F1 score
Our proposed model (degree + position)	0.3855
Our proposed model(degree centrality score only)	0.3719
The Degree centrality based approach proposed in [15]	0.3685

We conducted a comparison of the proposed summarization method with those that participated in DUC 2004 Task 2. The ROUGE scores for the systems that participated in the DUC 2004 Task 2 have been taken from the paper by Sarkar et al. (2015) [169]. The summaries generated by the participating teams during the DUC 2004 contest were made available by the conference authority on their website [2]. The leading teams that participated in DUC 2004 are identified by peer codes 65, 104, and 35. Among these, the team assigned peer code 65 emerged as the top-performing system in DUC 2004.

Comparisons of the proposed model with the systems that participated in the DUC 2004 Task 2 are shown in Table 4.4. In this table, we have also compared the proposed method with three existing multidocument summarization systems (MDS), two systems proposed by Sarkar et al. in 2015 [169] and 2022 [170], and another centroid-based system called Mead developed by Radev et al. [15]. A short description of these three existing models is as follows.

- The method proposed by Sarkar et. al in 2015 [169] is most similar to our proposed method because they used degree centrality and positional information. However, our method differs in the similarity measure used for constructing a graph and redundancy removal. Sarkar et. al(2015) used a hybrid similarity measure that combines two lexical similarity measures

[2]http://duc.nist.gov

whereas our method uses a hybrid similarity measure that combines a lexical similarity measure and a BERT-based semantic similarity measure.

- The method proposed by Sarkar et. al in 2022 [170] used semantic term relations for finding the term weights. The sentence score is a linear combination of the term weight-based score and the positional score. To find term relations, they used cosine similarity between the word embedding-based representation vectors for the terms.

- Method developed by Radev et. al. [15] was incorporated in an MDS system called Mead that combines centroid score with positional feature and sentence length feature. In this work, the terms whose TF*IDF weights were greater than a predefined threshold value were considered centroid. The similarity of a sentence with the centroid is taken as the sentence score which is further combined with the positional score for finding the overall sentence score.

Table 4.4: Comparison with the systems that participated in the DUC 2004 Task 2 and three existing MDS systems

Model	Rouge-1 F1 score
Our proposed method(degree + position) with hybrid similarity ($\alpha = 0.8$)	**0.3855**
The method proposed in [170]	0.3840
A Graph-based system with a hybrid similarity measure proposed in [169]	0.3820
DUC Sys65	0.3795
DUC Sys35	0.3757
MEAD baseline[15]	0.3737
Sys104	0.3712
DUC Coverage baseline	0.3454
DUC Lead baseline	0.3210

As we can see in Table 4.4, the proposed method performs better than all existing methods presented in the table.

4.4 Chapter Summary

In this chapter, we propose a hybrid similarity measure to improve graph-based extractive multidocument summarization. The proposed similarity measure blends a lexical similarity measure with a BERT-based semantic similarity measure. We use this similarity measure to build a similarity graph where a node corresponds to a sentence, and the weight of an edge between two nodes is the value of the hybrid similarity between the corresponding sentences. We set a threshold value for the edge weight. If the edge weight is not greater than a predefined value, the edge is removed from the graph. The degree centrality of a node is considered as the score of the corresponding sentence, and the sentences are ranked based on their scores. Finally, an extractive multi-document summary is generated using MMR(Maximum Marginal Relevance) method. The proposed method has been tested on benchmark English dataset named DUC2004 dataset and the results are compared with several state-of-the-art multi document summarization systems. Experimental results reveal that the summarization method with the proposed hybrid similarity measure is effective for extractive multi-document summarization.

5 Abstractive Text Summarization Using Deep Learning–based Sentence Fusion

5.1 Introduction

Multi-document abstractive text summarization aims to generate a gist focusing on salient concepts in the set of related documents. Abstracting is such a human ability that it is difficult to model with a machine. Thus, automatic abstracting is tough and is considered one of the most difficult tasks in natural language processing (NLP). Over the past few years, with the advancement of deep learning technology in the natural language processing (NLP) domain, abstractive text summarization also has made significant progress using a text-to-text generative model. One of the drawbacks of the text-to-text generative model is its input length restriction, which limits the amount of text that can be processed at once. This can pose challenges when summarizing multiple text documents as a whole. The human process of document summarization involves summarizing documents by sentence fusion. Sentence fusion combines two or more sentences to create an abstract sentence. Sentence fusion is useful to convert an extractive summary to an abstractive summary. The extractive summary contains a set of salient sentences selected from multiple related documents. Redundancy creates problems while creating an extractive summary because it contains sentences whose segments or phrases are redundant. Sentence fusion helps remove redundancy by fusing sentences into a single abstract sentence. This converts an extractive summary to an abstractive summary.

The abstractive summarization process generates a human-like summary that may contain new words that are not present in the input. When human creates a summary, they focus on salient sentences which are often fused to create a concise summary. The earlier approach to abstract generation [117] in text summarization used local and global trimming of the extracted sentences to produce an abstractive summary. Barzilay and McKeown (2005) [23] proposed a parse and fuse approach for sentence fusion to produce an abstractive summary. The target of this work was to create a grammatically correct non-redundant sentence by fusing multiple sentences. Many other researchers have used compression rules to remove sentence constituents to compress sentences. They have used anaphoric constraints[171] or integer linear programming [172] to maintain cross-sentence coherence. The text generation-based approach [164] [120] [121] is another kind of approach that selects textual units from the input and uses a surface realization step to generate new sentences.

In neural network-based approaches[122] [173], authors have used sentence extraction and sentence compression by rewriting using a complex neural network architecture. Recently, the transformer-based approach has shown unprecedented success in the domain of text generation. A transformer-based model used an encoder-decoder architecture [174][175] where the encoder accepts input sentences and the decoder generates an abstract based on the input sentences. One of the problems with the transformer-based model is that it has an input length restriction that truncates the input to some limit. This leads to difficulty in processing a longer text document as a whole. To combat this situation, sentence selection and sentence fusion can be useful. In this method, extracted sentences can be carefully paired and fused to obtain an abstract sentence. Thus, sentence fusion is useful for summarizing long documents using the transformer-based model. With inspiration from the success of neural networks in machine translation, the attention-based encoder-decoder framework has become an increasingly popular method for abstractive summarization. This approach allows the model to dynamically access relevant information from the input during the generation of the output sequence, as the decoder's hidden states revisit the input and focus on important details. The initial methods for generating abstracts[117] in text summarization involved trimming sentences at the local and global level to produce an abstractive summary. Neural network-based methods [122] [173], have employed intricate architectures to perform sentence extraction and compression using rewriting techniques. The use of transformer-based models has recently achieved unparalleled success in text generation tasks. The transformer-based model adopts an encoder-decoder structure [174][175], with the encoder processing input sentences and the decoder generating an abstract based on them. One of the drawbacks of this model is its input length restriction, which limits the amount of text that can be processed at once. This can pose challenges when summarizing a longer document as a whole. This situation becomes worse when multi-document abstraction is done using such a generative deep-learning model because the input size for multi-document abstractive summarization is very large and the input contains a collection of related documents.

In this chapter, we propose an abstractive summarization approach that uses an unsupervised extractive summarization model to select a small number of significant sentences from the input and then passes the extracted sentences to a deep learning-based sentence fusion model that produces a number of fused summary sentences. Finally, a subset of the fused sentences is selected using a criterion that maximizes the information content of the final summary. KL divergence (Kullback-Leibler divergence) based method is used to select some informative non-redundant sentences among a list of fused sentences to produce the final summary.

In this summarization method, a fused sentence is selected in a summary if the term distribution in the summary after adding this fused sentence mostly agrees with the term distribution in the source documents. This is based on the hypothesis that the best summary is a subset of sentences that has the closest term distribution to that in the input documents. A transformer-based text-to-text sentence fusion model is trained over a large dataset which is automatically created using the multi-news dataset [1].

In the first part of this chapter, we have discussed in detail the architecture of the text-to-text transformer model. In the second part of this chapter, training data preparation is discussed. The third part of this chapter focuses on experimental results and evaluation metrics. Comparisons with existing approaches are given in the last part of the chapter.

5.2 Proposed abstractive multi-document summarization approach

The proposed approach has several steps: (1) Generating an extract from the input document set using an unsupervised multi-document text summarization method, (2) fusing the pairs of sentences into a single sentence using a text-to-text generative model, and (3) using the KL divergence (Kullback-Leibler divergence)-based method to select a few informative nonredundant sentences among a list of fused sentences to produce the final summary. The steps of our proposed system are illustrated in Figure 5.1.

5.2.1 Unsupervised Text Summarization Model

Since the input to the proposed summarization system is a set of related documents, the input size is very large. The text-to-text generative model has an input length restriction. To deal with this situation, we have an unsupervised multi-document summarization system for generating first a draft extract which is then passed to the text-to-text generative model for generating the final summary.
We have used the Lexrank and Sumbasic method to create an extractive summary.

[1] $https://huggingface.co/datasets/multi_news/viewer/default$

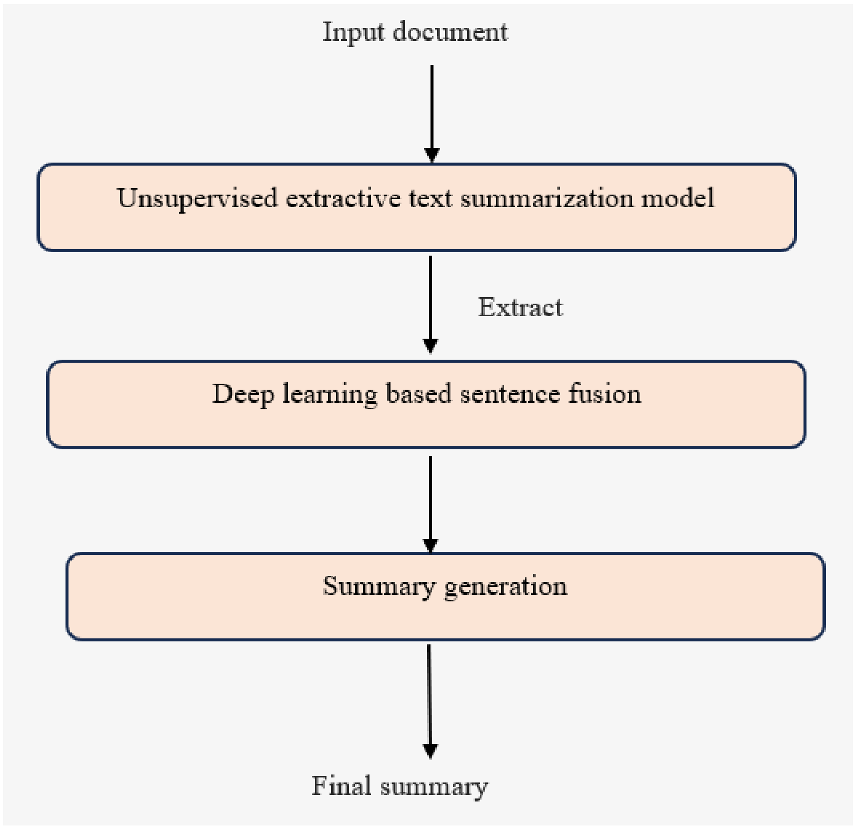

Input document
Unsupervised extractive text summarization model
Extract
Deep learning based sentence fusion
Summary generation
Final summary

Figure 5.1: Steps of our abstractive summarization approach

LexRank is a sentence ranking approach using a graph-based method. The details of this method can be found in [15]. In this method, the input is represented as a graph where the sentences correspond to the nodes, and the edge between the two nodes represents the cosine similarity value between the two nodes. Finally, Google's page rank algorithm is applied to the graph to rank sentences, and the top K (K is specified by the user)sentences are chosen as an extract. ”Sumbasic” was developed by [149] which considers term frequency as the saliency of a term. Here, the probability of a word is calculated based on its frequency, and each sentence is assigned a score equal to the average probability of the words contained in the sentence. The sentence with the highest score is selected first in the summary. Before selecting the next sentence, the words present in the already selected sentence are penalized by multiplying their probability values by themselves, and the sentences are reranked using the newly calculated probability values. After re-ranking the sentences, the sentence with the highest score is selected as the second sentence of the summary. This process is continued until the summary of the desired length is obtained.

We have chosen the LexRank and Sumbasic model from the text summarization library called sumy [2].

5.2.2 Deep Learning Based Sentence Fusion

The sentences in the extract created by the LexRank and Sumbasic algorithm are paired and each pair of sentences is fed to a deep learning-based sentence fusion model that fuses two related sentences to form a fused abstract sentence. The sentences are paired in such a way that the members of a pair are highly related to each other. We apply the following steps to produce suitable sentence pairs submitted to the sentence fusion model.

Algorithm 8 Steps for producing sentence pairs to be submitted to the sentence fusion model

```
1: i=1
2: Select the first sentence $X_i$ from the extract and compare $X_i$ with each other sentences in the extract and
   choose the sentence $X_j$ which has the highest cosine similarity with $X_i$ and form a sentence pair $<X_i, X_j>$.
3: Delete two sentences $X_i$ and $X_j$ from the extract. i = i+1. If i < length of the extract go to step 2, else Go to
   step 4.
4: End
```

As we mentioned in the above algorithm, cosine similarity is used to compute the relationship between two sentences. To apply the cosine similarity measure, the sentences require to be represented as vectors. In this case, we have used BERT(Bidirectional Encoder Representations from Transformers) embedding to obtain the vector for a sentence. cosine similarity is the cosine of the angle between two vectors, which means the dot product of two vectors divided by the product of their lengths. The cosine similarity is calculated using the equation 5.2, where A and B are two sentence vectors.

After obtaining the pair of sentences, they are passed one by one to the deep learning-based sentence fusion model, which is basically a text-to-text generative model [3]. The training procedure for this deep model is discussed in Section 5.2.4.

After finding the fused sentences, a subset of them is selected to form the final summary. We assume that the term frequency distribution in the final summary should be as close as possible to the term frequency distribution in the input. To ensure this, when the fused sentences are added to the final summary, a Kullback-Leibler divergence-based criterion is checked. The details of this process are discussed in the next subsection.

[2]https://github.com/miso-belica/sumy

[3]https://huggingface.co/huggingface-course/mt5-small-finetuned-amazon-en-es

5.2.3 Summary Generation

In this process, a subset of fused sentences is selected to create a summary. The target is to select a subset of fused sentences such that the KL divergence (Kullback-Leibler divergence) between this subset and the input is the minimum. KL divergence measures how much the term distribution in the summary differs from that in the input. We hypothesize that the best summary is the one whose term distribution is closest to that in the input document set.

The summary is created by selecting the fused sentences one by one. Here, the question is: which sentence will be selected first in the summary? The sentence whose inclusion in the summary minimizes the KL divergence between the summary and the input is first added to the summary. From the remaining fused sentences, we choose the second summary sentence in a similar way. Kullback-Leibler divergence (often abbreviated as KL divergence or simply KL) is a measure of how much two probability distributions are different from each other. The KL divergence between two probability distributions P and Q is defined as:

$$KL(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)} \tag{5.1}$$

where x is a possible outcome of the random variable, P(x) is the probability of x in the distribution P, and Q(x) is the probability of x under the distribution Q. The logarithm is usually taken to base 2, so that the units of KL are bits.

One way to interpret the KL divergence is as the expected value of the logarithmic difference between P and Q, weighted by P. In other words, it measures the average amount of surprise (in bits) when observing a sample from P instead of Q. In our case, P is the probability distribution of the terms in the summary and Q is the probability distribution of the terms in the set of input documents. The steps of the summary generation process are as follows.

Algorithm 9 Steps of the summary generation process

```
Input: A list of fused sentences, $FS = (fs_1, fs_2, .., fs_k)$, where k= number of fused sentences. SUMM=( ).
Output: Non redundant abstractive Summary
 1: Compute the probability distribution of the terms in the input, $P_{input} = (p_1, p_2, .., p_n)$, where n is the
    number of distinct terms in the input (vocabulary size).
 2: for each sentence $fs_i$ in FS do
 3:     SUMM = SUMM U $fs_i$ (Simulate adding the sentence)
 4:     Compute the probability distribution of terms in SUMM, $P_{SUMM}$.
 5:     $KL_i$ = compute KL divergence($P_{input}$,$P_{SUMM}$)
 6:     SUMM = SUMM - $fs_i$ (Remove the sentence from the summary)
 7: end for
 8: $j = argmin_i(KL_i)$
 9: SUMM = SUMM U $FS_j$. Delete $FS_j$ from FS. If FS is not empty, go to step 2, else go to step 10.
10: End
```

5.2.4 Training Sentence fusion Model

A text-to-text transformer is used for sentence fusion. We have used the huggingface mT5 model[4], a multilingual version of T5. mT5 has been used for many text-to-text generation tasks [174] mT5 is a multilingual text-to-text transformer that was pre-trained in 101 languages. The generic architecture of a transformer model is shown in Fig. 5.2. A transformer model has an encoder stack and a decoder stack. Although most transformer models follow the encoder-decoder architecture [174], there are some exceptions like the GPT family of models.

The encoder maps a sequence (x_1,x_2,..., x_n) to a continuous representation z = (z_1,z_2,..., z_n) [174]. The decoder decodes Z to produce the output (y_1,y_2,..., y_m) with an element at a time. It is an auto-regressive model. In subsequent subsections, we have discussed the following primary building blocks of a transformer-based text-to-text generation model.

- **Embedding Layer**
- **Positional Embedding Layer**
- **Encoder**
- **Decoder**

5.2.4.1 Embedding Layer:

The first pair of input sentences is passed to the embedding layer by converting it into a vector. The embedding layer maps the input tokens to the sequence of vectors. The purpose of this layer is to capture the semantic information from the input words. Neural networks can understand numbers, hence mapping of each input word to a vector with continuous values is done to represent that word.

5.2.4.2 Positional Embedding Layer:

Positional information plays an important role in learning information about the order of words in input. In the case of recurrent neural network(RNN), the ordering of the input word is considered. Sequentially input is parsed word by word through this layer. But the encoder in the transformer has no recurrence like recurrent neural networks, so it is necessary to incorporate positional information via this layer. As it is not captured by the embedding layer, hence it is appended with the output of the embedding layer.

[4]https://huggingface.co/huggingface-course/mt5-small-finetuned-amazon-en-es

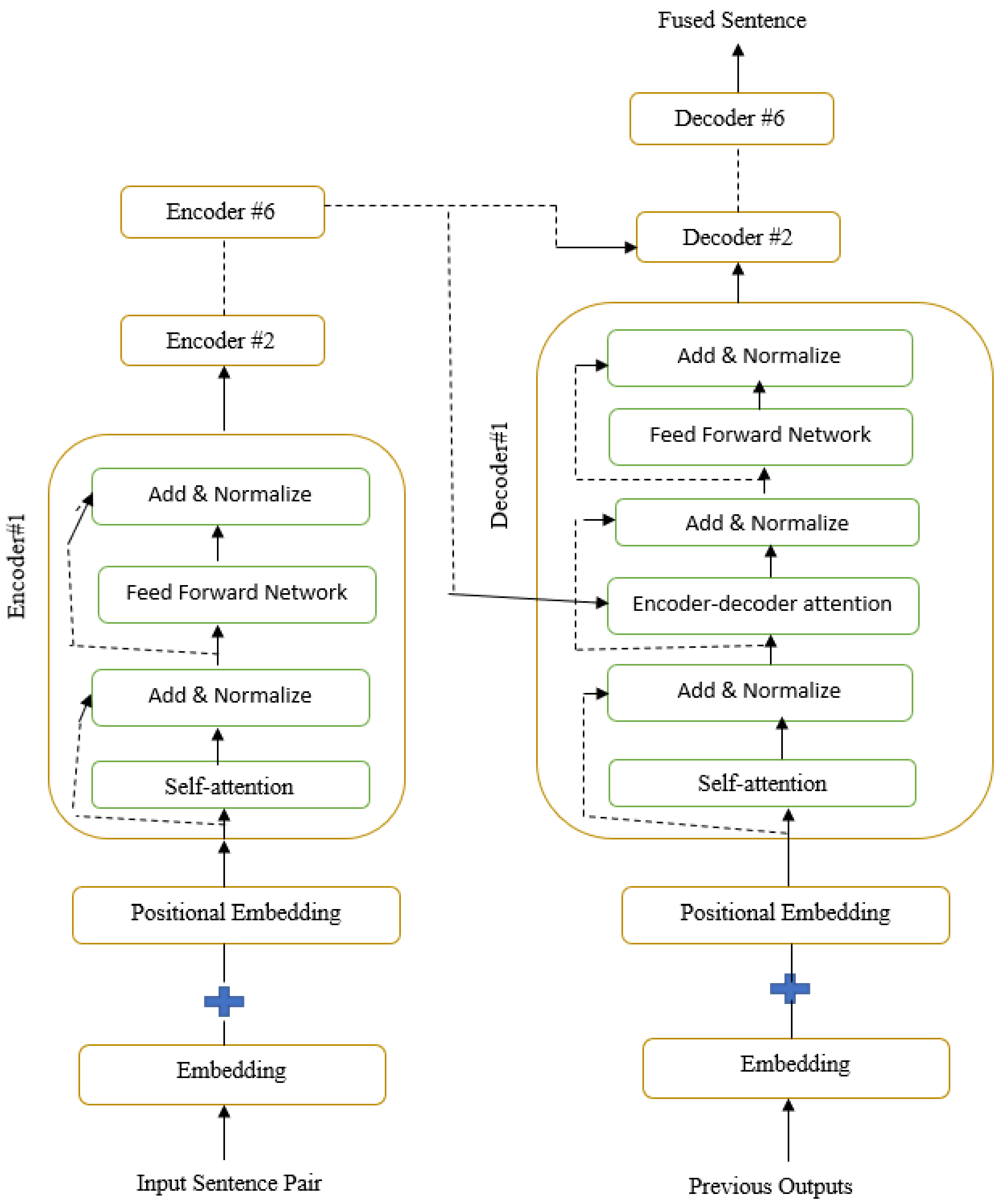


Figure 5.2: Architecture of deep learning based sentence fusion model

5.2.4.3 Encoder:

The transformer model consists of six encoders. The encoders are connected to each other. Each encoder has the following components.

- Self-attention: The output of the positional embedding layer is passed to the self-attention layer. This attention is known as intra-attention, as it considers the position of the input sequence to compute the representation of that sequence. The self-attention layer compares the input sequence numbers with each other, and the corresponding position of the output sequence is modified. Both the output and the input have the same dimension and sequence length.

- Add & Normalize: The output of the self-attention layer and the feedforward network layer is passed to this layer. "Add and normalize" is a two-step process. The "Add" step is the residual connection. It is used to solve the problem of vanishing gradients. The output of a layer is added with the input in the add step. The "Normalize" step is for layer normalization. The residual connection is incorporated with two sublayers after layer normalization. Layer normalization is defined as LayerNorm(x+Sublayer(x)) Where, Sublayer(x) denotes function of sub-layer.

- Feed Forward Network: The output of the add and normalize layer is passed to this layer. The purpose of this layer is to transform the attention vector in such a form that it is acceptable to the next encoder or decoder. Attention vectors are accepted once at a time in this layer. These attention vectors are independent of each other. Hence, parallelization can be applied here. The output is processed by this layer from an attention layer to better fit the input for the next attention layer. During summary generation, humans pay attention to the salient parts of the long document. Here, the attention function can be defined as the mapping of a query and a set of key-value pairs to an output, where the keys are the query, and the output are all vectors. The output is a weighted sum of the values, where the weight is computed using a compatibility function between the query and the corresponding key. Multihead attention permits the model to share information among different sub-spaces. In the encoder-decoder attention layers, the previous decoder layer and memory layer produce queries, and the encoder generates the values. This allows the decoder to pay attention to all positions in the input.

5.2.4.4 Decoder:

There are six decoders in the transformer. The output of the last encoder is passed to the second decoder and the encoder-decoder layer of the first decoder. Decoder also consists of a self-attention layer, add and ″normalize″ layer, and feed-forward network layer. Along with these a new layer is added here which is named ″encoder-decoder attention″.

- Encoder-Decoder Attention: It generates multihead attention over the encoder output. Here also a residual connection is used around two sub-layer. An improved version of self-attention is used to prevent attention to subsequent positions by positions. An encoder-decoder Transformer is built up with the encoder, where input sequence is given, and the decoder, which produce the output sequence. A "fully visible" attention mask is used by the encoder. It allows a self-attention mechanism for paying attention to each input of its output. In the decoder, the self-attention mechanism uses a masking pattern such that the future is unseen to the model during the generation of output.

5.2.4.5 Model implementation:

An input sentence pair and a reference fused sentence make up each training example for the model. Two sentences are joined as an input sentence pair by inserting the symbol ## between them. The input sentence pair is given to the encoder via embedding and then positional embedding layers, as seen in Fig. 5.2. The appropriate fused sentences are regarded as golden fused sentences during training and are used to calculate the loss. Our training data, which consists of 127897 training instances, is used to train the model. Since the transformer-based model requires a large amount of data set, we automatically created training instances. Section 5.2.5 discusses the specific steps involved in creating training data. The trained model is saved, and sentence pairs made from the extract produced by the LexRank and Sumbasic algorithms stated in the first part of the methodology section are fused using the saved trained model.

5.2.5 Training dataset preparation for sentence fusion model

For sentence fusion, the transformer model needs a lot of training data. It is laborious to manually generate a lot of training data. To create the dataset for our sentence fusion task, we used an automated process. In this part, the data generation process is explained. An input sentence pair and an output fused sentence make up each training instance. For the purpose of creating training data, we used the Multinews summarization footnote [5]. There are 56216 pairs of document summary pairs in the Multinews dataset. Each summary in this dataset is basically an abstract. A sentence in an abstract is presumed to be connected to several sentences in the original text. We locate the two source sentences that are most similar to a given sentence in the training abstract. The sentence chosen from the abstract is thought of as the equivalent fused sentence, and these two source sentences are thought of as the input sentence pair. As a result, the training examples are automatically constructed, each consisting of a pair of source sentences and the corresponding abstract sentence (considered as a fused sentence of the source pair). In Algorithm 10, the concept of automatic instance generation is presented. We used BERT embedding to represent sentences as vectors, and the cosine similarity metric was used to determine how semantically related the sentences are to each other. The sentence vector obtained by BERT is 768-dimensional. The cosine of the angle between two vectors, that is, the dot product of two vectors divided by the product of their lengths, is what is termed cosine similarity. Equation 5.2 is used to determine the cosine similarity between two sentence vectors A and B.

$$cosine - similarity = \frac{A.B}{||A||||B||} \tag{5.2}$$

The hyperparameters used during training the model are shown in Table 5.1.

Table 5.1: Hyperparameters used in model training

Parameter Name	Value
optimizer name	AdamWeightDecay
Train Loss	3.1301
Validation Loss	2.6937
No. Of Epoch	4

[5] $https://huggingface.co/datasets/multi_news/viewer/default$

Algorithm 10 Training data set creation

```
Input: Document summary pairs.
Output: A collection of instances where each instance consists of a sentence pair and the corresponding fused sentence.
1: for each document-summary pair do
2:     Break the input document and the corresponding summary into sentences. Obtain sentence vector using BERT.
3:     for each sentence S in an abstract do
4:         Compute the cosine similarity of S with each sentence in the source document.
5:         Rank the source sentences based on cosine similarity values from the highest to the lowest order.
6:         Choose the highest similar two sentences from the source document as the sentence pair and consider the sentence S as the fused sentence corresponding to the sentence pair.
7:     end for
8: end for
```

5.3 Experimental Results :

5.3.1 Evaluation metric:

We have evaluated the summaries produced by the proposed approach using the ROUGE metrics.

5.3.1.1 ROUGE:

ROUGE [141] measures n-gram overlap between a system-generated summary and the reference summaries [40]. ROUGE counts various kinds of unit that overlap between the system summary and the reference summaries. We have used the latest version of the ROUGE package - ROUGE 1.5.5 - for evaluating the system summaries. The ROUGE toolkit reports various ROUGE–N scores, for example, ROUGE-1, ROUGE-2, etc. Along with ROUGE-1 scores, many state-of-the-art summarization systems have been evaluated using ROUGE-2 (bigram-based) and ROUGE-SU4 (skip bigrams with skip distance up to 4 words [141]). So, we consider the ROUGE-1, ROUGE-2, and ROUGE-SU4 scores to evaluate our proposed summarization models. We set the summary length to 665 bytes as per DUC 2004 guidelines since we tested the proposed model on the DUC 2004 dataset. We use ROUGE-F score scores to evaluate and compare our proposed neural summarization method with other existing summarization methods.

5.3.2 Results:

We have used the DUC2004 dataset[6] for the evaluation of the proposed summarization model. The data set consists of 50 folders. Each folder contains approximately 10 documents which are news articles sourced from both the New York Times and Associated Press Wire services. Each article is accompanied by four

[6]https://duc.nist.gov/duc2004/

Table 5.2: System evaluation on test data

	R-1	R-2	R-SU4
Sumbasic + Sentence fusion model	**0.394**	**0.158**	**0.197**
Sumbasic	0.30216	0.126	0.159
Lexrank + Sentence fusion model	0.387	0.138	0.179
Lexrank	0.359	0.074	0.08

reference summaries that have been written by humans. Human summaries are basically abstract summaries. The system summaries are compared with reference summaries using the ROUGE method to evaluate the model performance.

We have developed two models where one model differs from another model in the type of extractive summarizer used: (1) the Sumbasic extractive summarizer + deep learning-based sentence fusion and (2) the Lexrank extractive summarizer + deep learning-based sentence fusion.

The performance of these two developed models on test data is shown in Table 5.2. In the table, the result of the best model is shown in bold. We can observe from this table that the performance of the proposed model with the Sumbasic extractive summarizer is better than that with the Lexrank extractive summarizer. This result tells us that the performance of the proposed summarization approach depends on the type of extractive summarizer used.

5.3.3 Comparison with existing approaches:

To show the efficacy of our proposed model, we have compared the performance of our model with some existing extractive and abstractive summarization approaches. Although the proposed approach produces an abstractive summary, we have compared the proposed approach with some extractive summarization approaches because these are proven to be effective methods for multi-document summarization. A brief description of the existing extractive approaches to which our proposed model is discussed in the following.

- **Centroid-based** The centroid-based summarization method, introduced in[10], is a technique that involves calculating a pseudo-sentence of the document, called the centroid. The centroid is formed by selecting the most important words in the document on the basis of their TF-IDF scores. A predefined threshold is set for the TF-IDF score, and any word with a score above this threshold is included in the centroid. The score of each sentence is computed on the basis of a set of features. These features include cosine similarity to the centroid, sentence position in the document, and cosine similarity to the first sentence. These features are combined to compute the final score for each sentence. The sentences are then ranked according to their scores, and the top-ranked sentences are elected to form a summary.

- **FreqSum**[165] This approach focuses on how frequency affects the summarization process by investigating three frequency-related factors: (1) the frequency of content words, (2) the use of composition functions to estimate the importance of a sentence based on word frequency, and (3) the adjustment of frequency weights based on the context. It is shown that this approach can achieve performance comparable to that of state-of-the-art summarization systems.

- **CLASSY04**[166] An ″oracle″ score introduced by authors is based on the probability distribution of unigrams in human summaries. They use this score to evaluate the quality of the generated extracts and compare them to human-written summaries. The results show that using the Oracle score, the generated extracts score better on average than the human summaries when evaluated using the ROUGE metric. They introduced an approximation to the oracle score and devised a method to identify a subset of terms that would likely be included in the reference summary. These terms are called signature terms. Using the signature terms, an extractive summary is generated from a set of documents.

The comparison of the proposed approach with some state-of-the-art extractive summarization approaches is shown in Table 5.3.

Table 5.3: Comparison of system performance with existing extractive summarization approaches

ROUGE Score			
Approaches	R-1	R-2	R-SU4
Sumbasic + Sentence fusion model	**0.394**	**0.158**	**0.197**
Centroid[10]	0.364	0.079	0.12
FreqSum[165]	0.353	0.081	0.1
CLASSY04[166]	0.376	0.089	0.15

From the above table (Table 5.3) we can conclude that our approach outperforms all extractive summarization approaches in terms of both R-1 and R-2 by a large margin.

We have also compared our model's performance with some state-of-the-art abstractive text summarization models. A brief description of these abstractive summarization models is discussed below.

- **ILPSumm**[167] This is a multi-document summarization technique that aims to generate a concise summary by clustering similar sentences, generating multiple paths, and selecting the most informative and readable sentences. After creating sentence clusters, a word graph is created. This structure allows to generation multiple paths that connect sentences within each cluster. The K-shortest path algorithm helps to find diverse and informative paths. From the set of shortest paths obtained, its aim is to select sentences that maximize the information content and readability of the final summary using integer linear programming (ILP). The ILP model formulates the selection process as an optimization problem, with the objective of maximizing information content while considering readability constraints.

- **ParaFuse doc**[168] This approach generates an abstractive summary using the paraphrastic sentence fusion model which executes sentence fusion and paraphrasing by skip-gram model. It emphasizes the information content of the generated abstract.

- **Align Fuse**[170] It creates an multidocument abstract using a graph-based approach. Sentences within clusters are represented as nodes in a graph, ensuring alignment between related sentences. The process of selecting summary sentences involves solving an Integer Linear Programming (ILP) problem, where the objective function and constraints are carefully crafted to capture the informativeness and linguistic quality of the sentences. To further enhance the summary's overall quality, an intensification function is employed to filter out lower-quality sentence fusions.

The comparison of the proposed approach with the state-of-the-art abstractive summarization approaches is shown in Table 5.4. The results are shown in Table 5.4.

Table 5.4: Comparison of system performance with existing abstractive summarization approaches

ROUGE Score		
Approaches	R-1	R-2
Sumbasic + Sentence fusion model	**0.394**	**0.158**
ILPSumm[167]	0.364	0.079
ParaFuse doc[168]	0.353	0.081
Align Fuse(sim =0)[170]	0.401	0.117
Align Fuse(sim =0.8)[170]	0.393	0.122

From the above table(Table 5.4) we can conclude that our approach outperforms all existing approaches in terms of R-2 scores.

5.4 Chapter Summary :

In this chapter, an abstractive multidocument summarization approach is presented. The proposed approach uses first an unsupervised summarization method for extractive summary generation and then a deep learning-based sentence fusion model to fuse related suitable sentence pairs for abstract generation. The proposed abstractive model outperforms several state-of-the-art extractive and abstractive summarization systems. We have considered two extractive summarizers:(1) the Sumbasic extractive summarizer and (2) the Lexrank extractive summarizer to create an extract. The experimental results reveal that the performance of the proposed abstractive model depends on the nature of extractive summarizers connected to the sentence fusion model.

6 Conclusions and Future Scope of Research

In this chapter, we conclude the thesis. Initially, we recapitulate the previous chapters. Subsequently, we draw conclusions and provide a preview of future avenues.

6.1 Summary

In the thesis, we investigate some techniques to improve the performance of extractive and abstractive text summarization. We have proposed various novel approaches for performing extractive and abstractive text summarization. Our investigations are summarized as follows.

- **Unsupervised Extractive Text Summarization techniques:** In Chapter 2, we propose several unsupervised techniques for performing single document extractive text summarization. In this chapter, we propose three single document extractive summarization methods (1) a method that combines saliency and novelty score for sentence ranking, (2) LSA based method for single document text summarization, and (3) a spectral clustering based single document extractive text summarization method. The proposed methods have been evaluated using a Bengali text summarization dataset which is developed by collecting the news documents from the popular daily Bengali news paper, Anandabazar Patrika, and manually creating a reference summary for each document. Our experiments reveal that each of the proposed methods is effective in generating summary from a single text document. We observe that among the proposed three methods, the spectral clustering based summarization method performs the best.

- **Supervised Extractive Text Summarization Using a Neural Network:** In Chapter 3, we propose a supervised machine learning-based extractive summarization model that uses neural networks to automatically model saliency and diversity in the extracted summary produced. This learning model learns to determine whether a sentence is summary worthy or not, based on the sentence-level features and the relations of the sentence under considerations with the sentences already selected in the summary in the previous step. Thus, it incrementally selects sentences in the summary by balancing saliency and diversity in the summary. A set of hand-made features is designed to measure the saliency of an individual sentence. The change in the summary diversity due to the inclusion of a sentence in the summary is defined by a set of diversity features which are basically the semantic relations between the sentence and other sentences previously selected in the summary. The proposed model has been tested on benchmark summarization datasets, the DUC2002 and Daily Mail(DM) datasets. The result obtained demonstrates that the proposed neural summarization model performs better than several state-of-the-art summarization systems.

- **Extractive Multi-Document Summarization:** In chapter 4, we propose a hybrid similarity measure for improving graph based extractive multi document summarization. The proposed similarity measure blends a lexical similarity measure with a BERT-based semantic similarity measure . We use this similarity measure for building a similarity graph where a node corresponds to a sentence and the weight of an edge between any two nodes is the value of the hybrid similarity between the corresponding sentences. We set a threshold value for the edge weight. If the edge weight is not greater than a predefined value, the edge is removed from the graph. The degree centrality of a node is considered as the score of the corresponding sentence, and the sentences are ranked based on their scores. Finally, an extractive multi-document summary is generated using MMR(Maximum Marginal Relevance) method. The proposed method has been tested on benchmark English dataset named DUC2004 dataset and the results are compared with several state-of-the-art multi document summarization systems. Experimental results reveal that the summarization method with the proposed hybrid similarity measure is effective for extractive multidocument summarization.

- **Abstractive Text Summarization:** In this chapter, we propose a novel method for abstractive multidocument text summarization. The proposed method first creates an extract using an unsupervised multi-document text summarization method and deep learning-based sentence fusion for generating an abstract from multiple related documents. We use a text-to-text transformer model, called MT5 model for fusing sentence pairs chosen from the extract generated by the unsupervised model. Finally, to generate an abstract, we use the KL divergence (Kullback-Leibler divergence)-based method to select a certain number of informative nonredundant sentences from the list of fused sentences produced by the deep learning model. The KL divergence method is a measure of how much one probability distribution differs from a reference probability distribution. During abstract generation, when fused sentences are selected one by one, the KL divergence method is used to check whether the selection of a fused sentence in the abstract improves information content and reduces redundancy. The proposed model is tested on a benchmark English dataset named DUC2004 dataset, and the results are compared with several state-of-the-art extractive and abstractive multidocument summarization systems. Experimental results reveal that the proposed model outperforms many state-of-the-art extractive and abstractive summarization systems.

6.2 Contribution of the Thesis

The main contributions of the thesis have been discussed in the chapters in question. In this section, we summarize the contribution of the thesis in Table 6.1.

Table 6.1: A summary of the contributions of the thesis

Chapter	Problem	Contribution
Chapter 2	To what extent can the unsupervised text summarization method solve the single document text summarization problem?	We have explored the following three unsupervised techniques for extractive text summarization and compared their performances: • A sentence ranking-based summarization method that combines saliency score and novelty score for sentence scoring. The sentence scoring technique linearly combines the saliency and novelty score. The novelty score of a sentence is the number of novel terms contained in the sentence.

Continued on next page

Table 6.1 – continued from previous Table

Chapter	Problem	Contribution
		We call a word a novel word for a sentence if the sentence contains the first appearance of the word. When combining the saliency score and the novelty score, a weighted combination of them is taken, and the weights are adjusted on our dataset to show whether both scores are needed or not. • A latent semantic indexing-based summarization method that uses singular value decomposition (SVD) of a term-by-sentence matrix created from a document to map each sentence to a latent semantic space and calculates the score of the sentence by the magnitude of the sentence vector after multiplying its components by the corresponding singular values.

Continued on next page

Table 6.1 – continued from previous Table

Chapter	Problem	Contribution
		• A spectral clustering-based summarization method that uses word-embedded sentence representation and a spectral clustering algorithm to identify various topics covered in a document and generate an extractive summary by selecting salient sentences from the identified topics.

Continued on next page

Table 6.1 – continued from previous Table

Chapter	Problem	Contribution
Chapter 3	How can both saliency and diversity in the generated summary be balanced using artificial neural network?	We have explored a neural network-based model that can learn to include a summary-worthy sentence in a summary by balancing the saliency and the diversity in the summary. A set of hand-crafted sentence-level features is designed to measure the saliency of a sentence. To address the diversity issue, a set of diversity features is defined on the basis of the semantic relations between the sentence and other sentences previously selected in the summary. To measure the semantic relation between sentences, word embedding-based sentence representation and cosine similarity measure are used. The neural network is designed such a way that it selects the summary worthy sentences one by one in the summary. The trained model classifies a sentence as summary worthy if the sentence is sufficiently salient and diverse with respect to the sentences previously selected in the summary.

Continued on next page

Table 6.1 – continued from previous Table

Chapter	Problem	Contribution
Chapter 4	Can the hybrid similarity measure combining lexical and BERT based semantic similarity improve multi-document summarization performance?	We have explored a combination of semantic and lexical similarity measure for developing a hybrid similarity measure that can improve graph-based multidocument extractive text summarization. For lexical similarity measure, we have used cosine similarity between sentence vectors obtained by the bag-of-words tokenization and TF*IDF based vector space model. For semantic similarity measure, we have used cosine similarity between sentence vectors obtained using a pre-trained BERT encoder. A weighted combination of semantic and lexical similarity measures has been considered and weights are tuned to determine whether both measures have contributions to the hybridization process.

Continued on next page

Table 6.1 – continued from previous Table

Chapter	Problem	Contribution
Chapter 5	Can deep learning based sentence fusion improve abstractive multi-document text summarization performance?	We have explored a text-to-text transformer model for sentence fusion, which takes a pair of related sentences and generates a single sentence by merging the information from the input pair. We have used this sentence-fusion model to fuse pairs of suitable sentences chosen from an extract. We have also explored a KL divergence-based method to create an abstract from the list of fused sentences. In the process of generating the abstract,the KL divergence method is used to assess whether the selection of a fused sentence in the summary enhances the information content and reduces redundancy in the summary.

6.3 Limitation of the Thesis

Although we have discussed the limitations of the proposed methods in the chapters concerned, in this section we summarize them.

- The extractive summarization methods proposed in Chapter 2 depend on sentence ranking. During summary generation, these methods select sentences one by one to form a summary using the MMR (maximum Marginal Relevance)[176] reranking method that reduces redundancy and improves the diversity of information in the summary. Thus each method proposed in Chapter 2 follows two-step process (1) sentence ranking and (2) improving diversity in the summary by reducing redundancy. The MMR (Maximum Marginal Relevance)[176] method is a greedy method that is sub-optimal in nature. Although the LSI based method and spectral clustering based method proposed in Chapter 2 deal with semantics, to some extent they work only on the input document containing hundreds of sentences which are insufficient for capturing semantic properly.

- Although the neural model proposed in chapter 3 which is trained to balance saliency and diversity simultaneously. For defining the diversity features we have used sentence vector where a sentence vector is average of GLOVE vectors of the words contained in the sentence. Although this approach improves summarization performance, more powerful sentence representation could be used for computing diversity feature values because sentence representation using averaging word vectors becomes less powerful when the sentence size is long. Since the model is trained in a supervised model, it needs a huge amount of training data(a collection of document-reference summary pairs) which is not always easy to collect for a news domain.

- In Chapter 4, to improve multidocument summarization performance, we propose a hybrid similarity measure that combines lexical and semantic similarity measures. For computing semantic similarity, the sentences are represented using BERT based sentence embedding. Since the hybrid similarity measure is used, it takes more time than the individual similarity measures. To reduce time, we have used the BERTbase(small) model though the BERT large model could produce more powerful sentence representation and give accurate semantic similarity value.

- In Chapter 5, we have performed abstractive text summarization. We have presented an abstractive multi-document summarization approach that uses an unsupervised summarization method for extractive summary generation and a deep learning-based sentence fusion model to fuse related sentence pairs for an abstract generation. As model's performance depends on the type of extractive summarizers used, if the extractive summarizer performs poorly, then the abstractive summarization model will not be able to produce quality output summary. Although the proposed model outperforms some state-of-the-art summarization systems in terms of ROUGE scores, in some cases the sentence fusion model produces an ungrammatical output that affects summary quality.

6.4 Conclusion

Finding an effective technique for extractive and abstractive text summarization is the main objective of the thesis. We have explored various unsupervised and neural network-based techniques for extractive single-document text summarization. We also investigated a hybrid similarity measure that combines lexical and semantic similarity to improve the summarization of graph-based extractive multidocument summarization. For abstract summarization, we investigated the deep learning-based sentence fusion and the KL divergence-based method for redundancy reduction. The summarization techniques proposed in this thesis include

- An unsupervised extractive method that linearly combines saliency and novelty for sentence scoring and single-document summarization.
- An unsupervised extractive method that uses LSA (Latent Semantic Analysis) for sentence scoring and single-document summarization.
- An unsupervised extractive method that uses a spectral clustering algorithm to segment the input document into multiple clusters and choose representative sentences from the clusters to form a summary.
- A neural network-based supervised method that can generate a single document extractive summary from a document by balancing saliency and diversity in the summary.
- An extractive multi-document summarization method that uses a hybrid similarity measure proposed in this thesis.
- An abstractive multi-document summarization method that generates an abstract from multiple related documents using sentence fusion.

All of the summarization methods mentioned above have been tested on benchmark datasets. We have used the popular automatic evaluation package ROUGE(Recall oriented Understanding for Gisting Evaluation) for summary evaluation. Finally, we conclude with the following observations.

- In Chapter 2, we have proposed three single document extractive summarization methods (1) a method that combines saliency and novelty score for sentence ranking, (2) LSA based method for single document text summarization, and (3) a spectral clustering based single document extractive text summarization method. The proposed methods have been evaluated using a Bengali text summarization dataset developed by us. Each of the proposed method has shown better performance on Bengali summarization dataset. Among the three methods proposed by us, the spectral clustering-based summarization method performs the best. However, one of the problems of our proposed approach is that the saliency and diversity of the generated summary are not well balanced. To address this, we apply the neural network model. Our proposed neural model is effective enough to handle this issue. In our next chapter, we have discussed the proposed neural extractive text summarization approach.

- In Chapter 3, a neural approach to extractive single document summarization has been presented. In this approach, deep feed forward neural network is used for selecting a summary sentence based on two factors - its saliency and how its inclusion in the summary improves summary diversity. Word embedding-based sentence representation and the cosine similarity measure have been used to find semantic sentence relations, which are used to compute diversity features. Experimental results show that our proposed neural model performs better than many existing methods for the DUC 2002 dataset when we consider the Rouge-1 and Rouge-2 scores, respectively, for summary evaluation.

- In Chapter 4, we propose a hybrid similarity measure for improving graph based extractive multi document summarization. The proposed similarity measure blends a lexical similarity measure with a BERT-based semantic similarity measure . We use this similarity measure for building a similarity graph where a node corresponds to a sentence and the weight of an edge between any two nodes is the value of the hybrid similarity between the corresponding sentences. We set a threshold value for the edge weight. If the edge weight is not greater than a predefined value, the edge is deleted from the graph. The degree centrality of a node is considered as the score of the corresponding sentence, and sentences are ranked based on their scores. Finally, an extractive multi-document summary is generated using MMR(Maximum Marginal Relevance) method. The proposed method has been tested on benchmark English dataset named DUC2004 dataset and the results are compared with several state-of-the-art multi document summarization systems. Experimental results reveal that the summarization method with the proposed hybrid similarity is effective for extractive multi-document summarization.

- From chapter 2 to chapter 4, we propose several extractive summarization techniques for single document and multi document summarization. In Chapter 5, we focus on abstractive text summarization. In this chapter, we propose a multi-document abstractive text summarization method that uses first an unsupervised summarization method for extractive summary generation and then a deep learning-based sentence fusion model to fuse related sentence pairs for an abstract generation. For implementing the sentence fusion model, we have used a text-to-text generative model named mT5. After producing fused sentences by the generative model, the final summary is generated by selecting fused sentences using KL-divergence-based method that maximizes information in the final abstractive summary. The proposed model is tested on benchmark English dataset named DUC2004 dataset, and the results are compared with several state-of-the-art extractive and abstractive summarization systems. The experimental result reveals that the proposed abstractive model outperforms many state-of-the-art extractive and abstractive summarization systems.

6.4.1 Scope of the Future Research

Although we have made every effort to ensure that this thesis is as comprehensive as possible, there is still room for improvement. Some future directions are summarized in this section.

- The feature-based extractive summarization method proposed in Chapter 2 is effective when a sufficient amount of training data is not available. In this method, we have used some important features for sentence scoring. However, the proposed method can be further enhanced by including some other hand-made features or linguistic features. The performance of the sentence clustering-based summarization method discussed in Chapter 2 can be improved by employing more advanced semantic techniques for sentence clustering and integrating them into the text summarization workflow.

- In Chapter 3, we propose a method in which sentence selection and redundancy control in the summary is done simultaneously by a trained multi-layer feedforward neural networks. To detect sentence-level redundancy, the word embedding-based sentence representation is used and for sentence representation, glove vectors are used. In the future, we will look for employing better sentence representation and sequence learning to improve summary quality.

- In Chapter 4, we propose a hybrid similarity measure which is used to improve a graph-based multi-document summarization. The hybrid similarity measure combines the lexical similarity and the BERT-based semantic similarity. Due to resource constraints, We have used the BERTbase model to compute similarity. In the future, we would like to use the BERTlarge model or other large language models to calculate semantic similarity.

- In Chapter 5, we propose a deep sentence fusion for abstractive summary generation. To generate an abstract from multiple related documents, the proposed method, a deep learning-based sentence fusion model that fuses sentence pairs selected from an extract. This model can be extended to employing multi-sentence fusion in generating an abstract from an extract.

Bibliography

[1] D. Evans, J. Klavans, and K. McKeown, "Columbia newsblaster: Multilingual news summarization on the web," 01 2004.

[2] H. P. Luhn, "The automatic creation of literature abstracts," *IBM Journal of Research and Development*, vol. 2, no. 2, pp. 159–165, 1958.

[3] H. P. Edmundson, "New methods in automatic extracting," *J. ACM*, vol. 16, no. 2, p. 264–285, apr 1969. [Online]. Available: https://doi.org/10.1145/321510.321519

[4] P. B. Baxendale, "Machine-made index for technical literature—an experiment," *IBM Journal of Research and Development*, vol. 2, no. 4, pp. 354–361, 1958.

[5] C.-Y. Lin and E. Hovy, "Identifying topics by position," in *Proceedings of the Fifth Conference on Applied Natural Language Processing*, ser. ANLC '97. USA: Association for Computational Linguistics, 1997, p. 283–290. [Online]. Available: https://doi.org/10.3115/974557.974599

[6] K. Sarkar, "An approach to summarizing bengali news documents," in *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, ser. ICACCI '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 857–862. [Online]. Available: https://doi.org/10.1145/2345396.2345535

[7] K. Sarkar, "Bengali text summarization by sentence extraction," 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:16657813

[8] R. Katragadda, P. Pingali, and V. Varma, "Sentence position revisited: A robust light-weight update summarization 'baseline' algorithm," 2009. [Online]. Available: https://api.semanticscholar.org/CorpusID:11989149

[9] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. USA: Prentice-Hall, Inc., 1988.

[10] D. R. Radev, H. Jing, M. Styś, and D. Tam, "Centroid-based summarization of multiple documents," *Information Processing and Management*, vol. 40, no. 6, pp. 919–938, 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306457303000955

[11] I. Mani, D. House, G. Klein, L. Hirschman, T. Firmin, and B. Sundheim, "The TIPSTER SUMMAC text summarization evaluation," in *Ninth Conference of the European Chapter of the Association for Computational Linguistics*, H. S. Thompson and A. Lascarides, Eds. Bergen, Norway: Association

for Computational Linguistics, Jun. 1999, pp. 77–85. [Online]. Available: https://aclanthology.org/E99-1011

[12] M. T. Maybury, "Generating summaries from event data," *Information Processing and Management*, vol. 31, no. 5, pp. 735–751, 1995, summarizing Text. [Online]. Available: https://www.sciencedirect.com/science/article/pii/030645739500025C

[13] K. Sarkar, "Syntactic sentence compression: Facilitating web browsing on mobile devices," in *2008 International Conference on Information Technology*, 2008, pp. 283–286.

[14] P. B. Baxendale, "Machine-made index for technical literature—an experiment," *IBM Journal of Research and Development*, vol. 2, no. 4, pp. 354–361, 1958.

[15] G. Erkan and D. R. Radev, "Lexrank: Graph-based lexical centrality as salience in text summarization," *J. Artif. Int. Res.*, vol. 22, no. 1, p. 457–479, dec 2004.

[16] J. Christensen, Mausam, S. Soderland, and O. Etzioni, "Towards coherent multi-document summarization," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, Jun. 2013, pp. 1163–1173. [Online]. Available: https://aclanthology.org/N13-1136

[17] I. Mani, E. Bloedorn, and B. Gates, "Using cohesion and coherence models for text summarization," 01 1998.

[18] I. Mani, B. Gates, and E. Bloedorn, "Improving summaries by revising them," 05 2002.

[19] J. Morris and G. Hirst, "Lexical cohesion computed by thesaural relations as an indicator of the structure of text," *Computational Linguistics*, vol. 17, no. 1, pp. 21–48, 1991. [Online]. Available: https://aclanthology.org/J91-1002

[20] J. C. Otterbacher, D. R. Radev, and A. Luo, "Revisions that improve cohesion in multi-document summaries: A preliminary study," in *Proceedings of the ACL-02 Workshop on Automatic Summarization - Volume 4*, ser. AS '02. USA: Association for Computational Linguistics, 2002, p. 27–36. [Online]. Available: https://doi.org/10.3115/1118162.1118166

[21] H. Iii and D. Marcu, "A noisy-channel model for document compression," 07 2002.

[22] Z. Cao, F. Wei, L. Dong, S. Li, and M. Zhou, "Ranking with recursive neural networks and its application to multi-document summarization," in *AAAI Conference on Artificial Intelligence*, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:10675728

[23] R. Barzilay and K. R. McKeown, "Sentence fusion for multidocument news summarization," *Computational Linguistics*, vol. 31, no. 3, pp. 297–328, 2005. [Online]. Available: https://aclanthology.org/J05-3002

[24] R. Barzilay, K. R. McKeown, and M. Elhadad, "Information fusion in the context of multi-document summarization," in *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*. College Park, Maryland, USA: Association for Computational Linguistics, Jun. 1999, pp. 550–557. [Online]. Available: https://aclanthology.org/P99-1071

[25] R. Barzilay, N. Elhadad, and K. R. McKeown, "Sentence ordering in multidocument summarization," ser. HLT '01. USA: Association for Computational Linguistics, 2001, p. 1–7. [Online]. Available: https://doi.org/10.3115/1072133.1072217

[26] D. Bollegala, N. Okazaki, and M. Ishizuka, "A bottom-up approach to sentence ordering for multi-document summarization," in *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*. Sydney, Australia: Association for Computational Linguistics, Jul. 2006, pp. 385–392. [Online]. Available: https://aclanthology.org/P06-1049

[27] M. Lapata, "Probabilistic text structuring: Experiments with sentence ordering," in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ser. ACL '03. USA: Association for Computational Linguistics, 2003, p. 545–552. [Online]. Available: https://doi.org/10.3115/1075096.1075165

[28] W. G. Lehnert, "Plot units and narrative summarization," *Cognitive Science*, vol. 5, no. 4, pp. 293–331, 1981. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S036402138180016X

[29] D. Linguistique and U. Montrral, "Synthesizing weather forecasts from formatfed data," 01 2003.

[30] C. D. Paice, "Constructing literature abstracts by computer: Techniques and prospects," *Information Processing and Management*, vol. 26, no. 1, pp. 171–186, 1990, special Issue: Natural Language Processing and

Information Retrieval. [Online]. Available: https://www.sciencedirect.com/science/article/pii/030645739090014S

[31] M. Hoey, "Patterns of lexis in text," 1991. [Online]. Available: https://api.semanticscholar.org/CorpusID:54142231

[32] B. Boguraev, "Salience-based content characterisafion of text documents," 1997. [Online]. Available: https://api.semanticscholar.org/CorpusID:1592006

[33] C. Cigir, M. Kutlu, and I. Cicekli, "Generic text summarization for turkish," in *2009 24th International Symposium on Computer and Information Sciences*, 2009, pp. 224–229.

[34] E. Hovy and C.-Y. Lin, "Automated text summarization and the summarist system," in *Proceedings of a Workshop on Held at Baltimore, Maryland: October 13-15, 1998*, ser. TIPSTER '98. USA: Association for Computational Linguistics, 1998, p. 197–214. [Online]. Available: https://doi.org/10.3115/1119089.1119121

[35] K. Sarkar and S. Bandyopadhyay, "Generating headline summary from a document set," 02 2005, pp. 649–652.

[36] M. Banko, V. Mittal, and M. Witbrock, "Headline generation based on statistical translation," 10 2002.

[37] D. M. Zajic, B. Dorr, R. M. Schwartz, and B. schwartz, "Automatic headline generation for newspaper stories," 2002. [Online]. Available: https://api.semanticscholar.org/CorpusID:12252356

[38] K. McKeown and R. Barzilay, "Information fusion for multidocument summarization: paraphrasing and generation," 2003. [Online]. Available: https://api.semanticscholar.org/CorpusID:59723731

[39] R. Barzilay and M. Elhadad, "Using lexical chains for text summarization," *Advances in Automatic Text Summarization*, 01 2000.

[40] X. Wan and J. Yang, "Improved affinity graph based multi-document summarization," in *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*. New York City, USA: Association for Computational Linguistics, Jun. 2006, pp. 181–184. [Online]. Available: https://aclanthology.org/N06-2046

[41] G. DeJong, "An overview of the frump system introduction," 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:187973332

[42] K. Sarkar, "Using domain knowledge for text summarization in medical domain," *International Journal of Recent Trends in Engineering*, vol. 1, pp. 200–205, 2009.

[43] H. Jing and K. R. McKeown, "Cut and paste based text summarization," in *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, ser. NAACL 2000. USA: Association for Computational Linguistics, 2000, p. 178–185.

[44] X. Wan and J. Yang, "Multi-document summarization using cluster-based link analysis," in *SIGIR-08*, 2008, p. 299–306.

[45] Y. Jin, *Multi-objective machine learning*. Springer Science & Business Media, 2006, vol. 16.

[46] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A K-Means clustering algorithm," *Applied Statistics*, vol. 28, no. 1, pp. 100–108, 1979. [Online]. Available: http://dx.doi.org/10.2307/2346830

[47] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '07. USA: Society for Industrial and Applied Mathematics, 2007, p. 1027–1035.

[48] D. Hamad and P. Biela, "Introduction to spectral clustering," in *Introduction to spectral clustering*, 05 2008, pp. 1 – 6.

[49] J. Tilton, "Image segmentation by region growing and spectral clustering with a natural convergence criterion," in *IGARSS '98. Sensing and Managing the Environment. 1998 IEEE International Geoscience and Remote Sensing. Symposium Proceedings. (Cat. No.98CH36174)*, vol. 4, 1998, pp. 1766–1768 vol.4.

[50] O. Sidi, O. van Kaick, Y. Kleiman, H. Zhang, and D. Cohen-Or, "Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering," *ACM Trans. Graph.*, vol. 30, no. 6, p. 1–10, dec 2011. [Online]. Available: https://doi.org/10.1145/2070781.2024160

[51] F. Lauer and C. Schnörr, "Spectral clustering of linear subspaces for motion segmentation," in *2009 IEEE 12th International Conference on Computer Vision*, 2009, pp. 678–685.

[52] A. Gupta, M. Kaur, A. Bajaj, and A. Khanna, "Entailment and spectral clustering based single and multiple document summarization," *International Journal of Intelligent Systems and Applications*, vol. 11, pp. 39–51, 04 2019.

[53] M. Osborne, "Using maximum entropy for sentence extraction," in *Proceedings of the ACL-02 Workshop on Automatic Summarization*. Phildadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002, pp. 1–8. [Online]. Available: https://aclanthology.org/W02-0401

[54] E. Günes and R. Dragomir R., "Lexrank: Graph-based lexical centrality as salience in text summarization," *J. Artif. Int. Res.*, vol. 22, no. 1, p. 457–479, Dec. 2004.

[55] R. Mihalcea, "Graph-based ranking algorithms for sentence extraction, applied to text summarization," in *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions*, ser. ACLdemo '04. USA: Association for Computational Linguistics, 2004, p. 20–es. [Online]. Available: https://doi.org/10.3115/1219044.1219064

[56] S. R. Chowdhury, K. Sarkar, and S. Dam, "An approach to generic bengali text summarization using latent semantic analysis," in *2017 International Conference on Information Technology (ICIT)*, 2017, pp. 11–16.

[57] K. Sarkar, "Automatic single document text summarization using key concepts in documents," *Journal of Information Processing Systems*, vol. 9, 12 2013.

[58] X. Wan, J. Yang, and J. Xiao, "Manifold-ranking based topic-focused multi-document summarization," in *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, ser. IJCAI'07. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007, p. 2903–2908.

[59] J. Kupiec, J. O. Pedersen, and F. R. Chen, "A trainable document summarizer," in *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1995. [Online]. Available: https://api.semanticscholar.org/CorpusID:5775833

[60] K. Sarkar, M. Nasipuri, and S. Ghose, "Using machine learning for medical document summarization," 2011.

[61] C. Aone, M. E. Okurowski, and J. Gorlinsky, "Trainable, scalable summarization using robust nlp and machine learning," in *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ser. ACL '98/COLING '98. USA: Association for Computational Linguistics, 1998, p. 62–66. [Online]. Available: https://doi.org/10.3115/980845.980856

[62] K. Sarkar, "Automatic keyphrase extraction from medical documents," vol. 5909, 12 2009, pp. 273–278.

[63] C. Shah and A. Jivani, *An Automatic Text Summarization on Naive Bayes Classifier Using Latent Semantic Analysis*, 01 2019, pp. 171–180.

[64] D. O'leary, J. Conroy, J. Schlesinger, and M. Okurowski, "Using hmm and logistic regression to generate extract summaries," 09 2002.

[65] D. M. Dunlavy, D. P. O'Leary, J. M. Conroy, and J. D. Schlesinger, "Qcs: A system for querying, clustering and summarizing documents," *Information Processing and Management*, vol. 43, no. 6, pp. 1588–1605, 2007, text Summarization. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306457307000246

[66] M. Fuentes Fort, E. Alfonseca, and H. Rodríguez, "Support vector machines for query-focused summarization trained and evaluated on pyramid data." 01 2007.

[67] L. M. Manevitz and M. Yousef, "One-class svms for document classification," *J. Mach. Learn. Res.*, vol. 2, p. 139–154, mar 2002.

[68] D. Shen, J.-T. Sun, H. Li, Q. Yang, and Z. Chen, "Document summarization using conditional random fields," in *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, ser. IJCAI'07. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007, p. 2862–2867.

[69] K. Kaikhah, "Automatic text summarization with neural networks," in *2004 2nd International IEEE Conference on 'Intelligent Systems'. Proceedings (IEEE Cat. No.04EX791)*, vol. 1, 2004, pp. 40–44 Vol.1.

[70] A. Qaroush, I. Abu Farha, W. Ghanem, M. Washaha, and E. Maali, "An efficient single document arabic text summarization using a combination of statistical and semantic features," *Journal of King Saud University - Computer and Information Sciences*, vol. 33, no. 6, pp. 677–692, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157818310498

[71] M. Kågebäck, O. Mogren, N. Tahmasebi, and D. Dubhashi, "Extractive summarization using continuous vector space models," 04 2014.

[72] S. R., "Recursive deep learning for natural language processing and computer vision," 2014.

[73] M. Yousefi-Azar and L. Hamey, "Text summarization using unsupervised deep learning," *Expert Systems with Applications*, vol. 68, pp. 93–105, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417416305486

[74] R. Nallapati, F. Zhai, and B. Zhou, "Summarunner: A recurrent neural network based sequence model for extractive summarization of documents," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, ser. AAAI'17. AAAI Press, 2017, p. 3075–3081.

[75] R. K. Roul, J. K. Sahoo, and R. Goel, "Deep learning in the domain of multi-document text summarization," in *Pattern Recognition and Machine Intelligence*, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:35322137

[76] Y. Liu, S. Zhong, and W. Li, "Query-oriented multi-document summarization via unsupervised deep learning," *Expert Systems with Applications*, vol. 2, 06 2015.

[77] J. Cheng and M. Lapata, "Neural summarization by extracting sentences and words," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 484–494. [Online]. Available: https://aclanthology.org/P16-1046

[78] K. Filippova, "Multi-sentence compression: Finding shortest paths in word graphs," in *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Beijing, China: Coling 2010 Organizing Committee, Aug. 2010, pp. 322–330. [Online]. Available: https://aclanthology.org/C10-1037

[79] K. Knight and D. Marcu, "Summarization beyond sentence extraction: A probabilistic approach to sentence compression," *Artificial Intelligence*, vol. 139, no. 1, pp. 91–107, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370202002229

[80] T. Fevry and J. Phang, "Unsupervised sentence compression using denoising auto-encoders," 01 2018, pp. 413–422.

[81] J. Zhou and A. Rush, "Simple unsupervised summarization by contextual matching," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 5101–5106. [Online]. Available: https://aclanthology.org/P19-1503

[82] C. Baziotis, I. Androutsopoulos, I. Konstas, and A. Potamianos, "SEQˆ3: Differentiable sequence-to-sequence-to-sequence autoencoder for unsupervised abstractive sentence compression," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 673–681. [Online]. Available: https://aclanthology.org/N19-1071

[83] S. Dohare, V. Gupta, and H. Karnick, "Unsupervised semantic abstractive summarization," in *Proceedings of ACL 2018, Student Research Workshop*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 74–83. [Online]. Available: https://aclanthology.org/P18-3011

[84] P. Laban, A. Hsi, J. Canny, and M. A. Hearst, "The summary loop: Learning to write abstractive summaries without examples," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 5135–5150. [Online]. Available: https://aclanthology.org/2020.acl-main.460

[85] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[86] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:160025533

[87] Y. Wang and H.-Y. Lee, "Learning to encode text as human-readable summaries using generative adversarial networks," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 4187–4195. [Online]. Available: https://aclanthology.org/D18-1451

[88] E. Chu and P. Liu, "MeanSum: A neural model for unsupervised multi-document abstractive summarization," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 1223–1232. [Online]. Available: https://proceedings.mlr.press/v97/chu19b.html

[89] A. Bražinskas, M. Lapata, and I. Titov, "Unsupervised opinion summarization as copycat-review generation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online:

Association for Computational Linguistics, Jul. 2020, pp. 5151–5169. [Online]. Available: https://aclanthology.org/2020.acl-main.461

[90] M. Isonuma, J. Mori, and I. Sakata, "Unsupervised neural single-document summarization of reviews via learning latent discourse structure and its ranking," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 2142–2152. [Online]. Available: https://aclanthology.org/P19-1206

[91] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu, "Mass: Masked sequence to sequence pre-training for language generation," in *International Conference on Machine Learning*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:146808476

[92] P. Liu, Y.-A. Chung, and J. Ren, "Summae: Zero-shot abstractive text summarization using length-agnostic auto-encoders," 10 2019.

[93] P. B. Baxendale, "Man-made index for technical literature an experiment," *IBM Journal of Research and Development*, vol. 2, pp. 354–361, 1958.

[94] H. P. Edmundson, "New methods in automatic extracting," *Journal of the Association for Computing Machinery*, vol. 16, pp. 264–285, 1969.

[95] K. Sarkar, M. Nasipuri, and S. Ghose, "Using machine learning for medical document summarization," *International Journal of Database Theory and Application*, pp. 31–49, 2011.

[96] K. R. McKeown and D. R. Radev, "Generating summaries of multiple news articles," in *In Proceedings of the 18th Annual International ACM SIGIR Conference on Re-search and Development in Information Retrieval*. Seattle, 1995, pp. 74–82.

[97] K. Sarkar, K. Saraf, and A. Ghosh, "Improving graph based multidocument text summarization using an enhanced sentence similarity measure," in *2015 IEEE 2nd International Conference on Recent Trends in Information Systems (ReTIS)*, I. Kolkata, Ed., 2015, pp. 359–365.

[98] R. Ferreira *et al.*, "Assessing sentence scoring techniques for extractive text summarization," vol. 2013.

[99] R. Shardan, *Implementation and evaluation of evolutionary connectionist approaches to automated text summarization*. V. Kulkarni., 2010.

[100] M. N. Azadani, N. Ghadiri, and E. Davoodijam, "Graph-based biomedical text summarization: an itemset mining and sentence clustering approach," *J Biomed Inform*, vol. 84, pp. 42–58, 2018.

[101] D. Kadam. Comparative Study Of Hindi Text Summarization Techniques: Genetic Algorithm And Neural Network, 2015.

[102] M. Gambhir and V. Gupta, "Recent automatic text summarization techniques: a survey," *Artif Intell Rev*, vol. 47, pp. 1–66, 2017.

[103] V. Gupta, "Hybrid algorithm for multilingual summarization of hindi and punjabi documents," 2013, pp. 717–727.

[104] V. Gupta and G. Lehal, "Survey of text summarization extractive techniques," *Journal of Emerging Technologies in Web Intelligence*, vol. 2, no. 10, pp. 258–268, 2010.

[105] A. Abuobieda *et al.*, "Text summarization features selection method using pseudo genetic-based model," pp. 193–197, 2012.

[106] M. A. Fattah and F. Ren, "Ga, mr, ffnn, pnn and gmm based models for automatic text summarization," *Comput Speech Lang*, vol. 23, pp. 126–144, 2009.

[107] K. Nandhini and S. R. Balasundaram, "Improving readability through extractive summarization for learners with reading difficulties," *Egypt Inform J*, vol. 14, pp. 195–204, 2013.

[108] R. Mihalcea and P. Tarau, "A language independent algorithm for single and multiple document summarization," *In: Proc. IJCNLP*, vol. 24, p. 2005, 2005.

[109] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Comput Networks ISDN Syst*, vol. 30, pp. 107–117, 1998.

[110] D. Parveen, H. M. Ramsl, and M. Strube, "Topical coherence for graph-based extractive summarization," in *Proceedings of theconference on empirical methods in natural language processing*, 2015, pp. 1949–1954.

[111] L. Plaza, M. Stevenson, and A. Diaz, "Resolving ambiguity in biomedical text to improve summarization," *Inf Process Manage*, vol. 48, pp. 755–766, 2012.

[112] S. Xiong and D. Ji., "Query-focused multi-document summarization using hypergraph-based ranking," *Inf Process Manage;*, vol. 52, pp. 670–681, 2016.

[113] E. Canhasi and I. Kononenko, "Weighted archetypal analysis of the multi-element graph for query-focused multi-document summarization," *Expert Syst Appl*, vol. 41, p. 535–543, 2014.

[114] G. Salton, "et al," *Automatic text structuring and summarization. Inf Process Manage;*, vol. 33, p. 193–207.

[115] O. Medelyan, "Computing lexical chains with graph clustering," *In: Proceedings of the ACLstudent research workshop*, p. 85–90, 2007.

[116] Y. n Chen *et al.*, *Spoken lecture summarization by random walk over a graph constructed with automatically extracted key terms.* In twelfth annual conference of the international speech communication association, 2011.

[117] K. Sarkar, "Syntactic trimming of extracted sentences for improving extractive multi-document summarization," *Journal of Computing*, vol. 2, pp. 177–184, 01 2010.

[118] K. Sarkar, "Improving multi-document text summarization performance using local and global trimming," in *Proceedings of the First International Conference on Intelligent Human Computer Interaction*, U. S. Tiwary, T. J. Siddiqui, M. Radhakrishna, and M. D. Tiwari, Eds. New Delhi: Springer India, 2009, pp. 272–282.

[119] K. Filippova, "Dependency graph based sentence fusion and compression," 04 2010.

[120] L. Bing, P. Li, Y. Liao, W. Lam, W. Guo, and R. Passonneau, "Abstractive multi-document summarization via phrase selection and merging," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 1587–1597. [Online]. Available: https://aclanthology.org/P15-1153

[121] V. Chenal and J. C. K. Cheung, "Predicting sentential semantic compatibility for aggregation in text-to-text generation," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, Dec. 2016, pp. 1061–1070. [Online]. Available: https://aclanthology.org/C16-1101

[122] Y.-C. Chen and M. Bansal, "Fast abstractive summarization with reinforce-selected sentence rewriting," 01 2018, pp. 675–686.

[123] A. See, P. Liu, and C. Manning, "Get to the point: Summarization with pointer-generator networks," 01 2017, pp. 1073–1083.

[124] K. Song, L. Zhao, and F. Liu, "Structure-infused copy mechanisms for abstractive summarization," 06 2018.

[125] 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," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 7871–7880. [Online]. Available: https://aclanthology.org/2020.acl-main.703

[126] R. L. Donaway, K. W. Drummey, and L. A. Mather, "A comparison of rankings produced by summarization evaluation measures," in *Proceedings of the 2000 NAACL-ANLPWorkshop on Automatic Summarization - Volume 4*, ser. NAACL-ANLP-AutoSum '00. USA: Association for Computational Linguistics, 2000, p. 69–78. [Online]. Available: https://doi.org/10.3115/1117575.1117583

[127] J. Carletta, "Assessing agreement on classification tasks: The kappa statistic," *Comput. Linguist.*, vol. 22, no. 2, p. 249–254, jun 1996.

[128] D. R. Radev, H. Jing, and M. Budzikowska, "Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies," in *NAACL-ANLP 2000 Workshop: Automatic Summarization*, 2000. [Online]. Available: https://aclanthology.org/W00-0403

[129] D. Harman and D. M., "Proceedings of the document understanding conference (duc-01)," 2001.

[130] U. Hahn and D. Harman, "Proceedings of the document understanding conference (duc-02)," 2002.

[131] C.-Y. Lin, "Summary evaluation environment," 2001.

[132] I. Mani, G. Klein, L. Hirschman, T. Firmin, and B. Sundheim, "The tipster summac text summarization evaluation," 05 2002.

[133] T. Firmin and M. J. C. 1999, "An evaluation of automatic text summarization systems," pp. 325–336, 1999.

[134] H. Jing, R. Barzilay, K. McKeown, and M. Elhadad, "Summarization evaluation methods: Experiments and analysis," *AAAI Symposium on Intelligent Summarization*, 03 2000.

[135] R. Donaway, K. Drummey, and L. Mather, “A comparison of rankings produced by summarization evaluation measures,” 12 2002.

[136] D. Marcu, “From discourse structures to text summaries,” 12 2002.

[137] D. Marcu, “The automatic construction of large-scale corpora for summarization research,” in *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR ’99. New York, NY, USA: Association for Computing Machinery, 1999, p. 137–144. [Online]. Available: https://doi.org/10.1145/312624.312668

[138] J. Stewart, M. Kantrowitz, V. Mittal, and J. Carbonell, “Summarizing text documents: Sentence selection and evaluation metrics,” 01 2003.

[139] K. Papineni, S. Roukos, T. Ward, and W. J. Zhu, “Bleu: a method for automatic evaluation of machine translation,” 10 2002.

[140] C.-Y. Lin and E. Hovy, “Manual and automatic evaluation of summaries,” in *Proceedings of the ACL-02 Workshop on Automatic Summarization*. Phildadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002, pp. 45–51. [Online]. Available: https://aclanthology.org/W02-0406

[141] C.-Y. Lin, “ROUGE: A package for automatic evaluation of summaries,” in *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81. [Online]. Available: https://aclanthology.org/W04-1013

[142] K. Sarkar, “Sentence clustering-based summarization of multiple text documents,” *TECHNIA – International Journal of Computing Science and Co mmunication Technologies*, vol. 2, pp. 325–335, 01 2009.

[143] J. Carbonell and J. Goldstein, “The use of mmr, diversity-based reranking for reordering documents and producing summaries,” in *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR ’98. New York, NY, USA: Association for Computing Machinery, 1998, p. 335–336. [Online]. Available: https://doi.org/10.1145/290941.291025

[144] M. F. Porter, “An algorithm for suffix stripping,” *Program*, vol. 40, pp. 211–218, 1997.

[145] D. R. Radev, H. Jing, M. Styś, and D. Tam, “Centroid-based summarization of multiple documents,” *Information Processing &*

Management, vol. 40, no. 6, pp. 919–938, 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306457303000955

[146] K. Sarkar, "Centroid-based summarization of multiple documents," *TECHNIA–International Journal of Computing Science and Communication Technologies*, vol. 2, 2009.

[147] U. Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, pp. 395–416, 01 2007.

[148] E. Günes and R. Dragomir R., "Lexrank: Graph-based lexical centrality as salience in text summarization," *J. Artif. Int. Res.*, vol. 22, no. 1, p. 457–479, Dec. 2004.

[149] N. Ani and V. Lucy, "The impact of frequency on summarization," *Computer Science*, 01 2005.

[150] M. Rada and T. Paul, "TextRank: Bringing order into text," in *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 404–411.

[151] R. Nallapati, F. Zhai, and B. Zhou, "Summarunner: A recurrent neural network based sequence model for extractive summarization of documents," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, ser. AAAI'17. AAAI Press, 2017, p. 3075–3081.

[152] K. M. Hermann, T. Kočiský, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, p. 1693–1701.

[153] H. Moen, J. Heimonen, L.-M. Peltonen, A. Airola, T. Pahikkala, V. Terävä, R. Danielsson-Ojala, T. Salakoski, and S. Salanterä, "On evaluation of automatically generated clinical discharge summaries," vol. 1251, 05 2014.

[154] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. [Online]. Available: https://aclanthology.org/D14-1162

[155] K. M. Hermann, T. Kočiský, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," in

Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, p. 1693–1701.

[156] H. Oliveira, R. Lins, R. Lima, and F. Freitas, "A regression-based approach using integer linear programming for single-document summarization," 11 2017, pp. 270–277.

[157] K. Svore, L. Vanderwende, and C. Burges, "Enhancing single-document summarization by combining RankNet and third-party sources," in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic: Association for Computational Linguistics, Jun. 2007, pp. 448–457. [Online]. Available: https://aclanthology.org/D07-1047

[158] X. Wan, "Towards a unified approach to simultaneous single-document and multi-document summarizations," in *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Beijing, China: Coling 2010 Organizing Committee, Aug. 2010, pp. 1137–1145. [Online]. Available: https://aclanthology.org/C10-1128

[159] R. M. Alguliyev, R. M. Aliguliyev, N. R. Isazade, A. Abdi, and N. Idris, "Cosum: Text summarization based on clustering and optimization," *Expert Systems*, vol. 36, no. 1, p. e12340, 2019.

[160] R. M. Aliguliyev, "A new sentence similarity measure and sentence based extractive technique for automatic text summarization," *Expert Systems with Applications*, vol. 36, no. 4, pp. 7764–7772, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417408008737

[161] W. Song, L. Cheon Choi, S. Cheol Park, and X. Feng Ding, "Fuzzy evolutionary optimization modeling and its applications to unsupervised categorization and extractive summarization," *Expert Syst. Appl.*, vol. 38, no. 8, p. 9112–9121, aug 2011. [Online]. Available: https://doi.org/10.1016/j.eswa.2010.12.102

[162] M. Mendoza, S. Bonilla, C. Noguera, C. Cobos, and E. León, "Extractive single-document summarization based on genetic operators and guided local search," *Expert Systems with Applications*, vol. 41, no. 9, pp. 4158–4169, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417413010312

[163] D. Shen, J.-T. Sun, H. Li, Q. Yang, and Z. Chen, “Document summarization using conditional random fields,” 01 2007, pp. 2862–2867.

[164] K. Liao, L. Lebanoff, and F. Liu, “Abstract meaning representation for multi-document summarization,” *ArXiv*, vol. abs/1806.05655, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:49210924

[165] A. Nenkova, L. Vanderwende, and K. McKeown, “A compositional context sensitive multi-document summarizer: Exploring the factors that influence summarization,” in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR ’06. New York, NY, USA: Association for Computing Machinery, 2006, p. 573–580. [Online]. Available: https://doi.org/10.1145/1148170.1148269

[166] J. M. Conroy, J. D. Schlesinger, and D. P. O’Leary, “Topic-focused multi-document summarization using an approximate oracle score,” in *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, ser. COLING-ACL ’06. USA: Association for Computational Linguistics, 2006, p. 152–159.

[167] S. Banerjee, P. Mitra, and K. Sugiyama, “Multi-document abstractive summarization using ilp based multi-sentence compression,” 09 2016.

[168] M. T. Nayeem, T. Fuad, and Y. Chali, “Abstractive unsupervised multi-document summarization using paraphrastic sentence fusion,” 08 2018.

[169] K. Sarkar, K. Saraf, and A. Ghosh, “Improving graph based multidocument text summarization using an enhanced sentence similarity measure,” *2015 IEEE 2nd International Conference on Recent Trends in Information Systems (ReTIS)*, pp. 359–365, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:16105394

[170] R. Agarwal and N. Chatterjee, “Improvements in multi-document abstractive summarization using multi sentence compression with word graph and node alignment,” *Expert Systems with Applications*, vol. 190, p. 116154, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417421014792

[171] G. Durrett, T. Berg-Kirkpatrick, and D. Klein, “Learning-based single-document summarization with compression and anaphoricity constraints,” 03 2016, pp. 1998–2008.

[172] A. Martins and N. Smith, “Summarization with a joint model for sentence extraction and compression,” *Association for Computational Linguistics*, pp. 1–9, 01 2009.

[173] A. Mendes, S. Narayan, S. Miranda, Z. Marinho, A. F. T. Martins, and S. B. Cohen, "Jointly extracting and compressing documents with summary state representations," *ArXiv*, vol. abs/1904.02020, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:102352252

[174] E. Zolotareva, T. Misikir Tashu, and T. Horvath, "Abstractive text summarization using transfer learning," 08 2020.

[175] B. Ay, F. Ertam, G. Fidan, and G. Aydin, "Turkish abstractive text document summarization using text to text transfer transformer," *Alexandria Engineering Journal*, vol. 68, pp. 1–13, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S111001682300008X

[176] J. Goldstein and J. Carbonell, "Summarization: (1) using mmr for diversity - based reranking and (2) evaluating summaries," in *Proceedings of a Workshop on Held at Baltimore, Maryland: October 13-15, 1998*, ser. TIPSTER '98. USA: Association for Computational Linguistics, 1998, p. 181–195. [Online]. Available: https://doi.org/10.3115/1119089.1119120