A

Ph.D. Thesis

on

Extractive Automatic Text Summarization Based On Voting and Filtering Techniques

submitted for partial fulfillment for the degree of

Ph.D. in Computer Science & Engineering

Academic Session

(2015-2016)

Supervisor:
Dr. Dinesh Gopalani

Submitted By:
Yogesh Kumar Meena
(2012RCP9001)

Department of Computer Science & Engineering

MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY, JAIPUR (RAJ.)

Department of Computer Science & Engineering

Malaviya National Institute of Technology , Jaipur - 302017

Certificate

I hereby certify that the thesis entitled, "**Extractive Automatic Text Summarization Based On Voting and Filtering Techniques**" submitted by **Yogesh Kumar Meena (2012RCP9001)** to Malaviya National Institute of Technology, Jaipur for the award of the degree of **Doctor of Philosophy** in Computer Science & Engineering is a bonafide record of original research wok carried out by him under my supervision. It is further certified that:

1. The results contained in this thesis have not been submitted in part or in full, to any other University or Institute for the award of any degree or diploma.
2. Mr. Yogesh Kumar Meena has fulfilled the requirement for the submission of this thesis.

(Dr. Dinesh Gopalani)
Assistant Professor
Department of Computer Science & Engineering
MNIT, Jaipur

Place: Jaipur

Date: 09-October-2015

Declaration

I, Yogesh Kumar Meena, declare that I own the research work introduced in this thesis entitled, "Extractive Automatic Text Summarization Based On Voting and Filtering Techniques" and the research contents used in this thesis. I with this assure that:

- The research work produced in this thesis is for the partial fulfillment of the degree of "*Doctor of Philosophy*" at MNIT, Jaipur.
- I have stated all the major resources used for the help.
- Where I have used proper citation for the work proposed by other researchers and quoted the source. This entire thesis belongs to me with the some exception of such citations.
- Where I have taken references of previously published work of other researchers and this is always clearly attributed.
- The results contained in this thesis have not been previously submitted for a degree or any other qualification at MNIT or any other institution.

Signed: ______________________________

Date: ______________________________

Acknowledgements

It is a matter of great pleasure for me to register for the Ph.D. program in the Department of Computer Science and Engineering, Malaviya National Institute of Technology, Jaipur, India. I would like to express sincere thanks and appreciation to my Ph.D. supervisor Dr. Dinesh Gopalani. He has devoted valuable time to guide me. His kind suggestions and cooperation together with the motivation made it possible to complete my research work. He has provided me a good atmosphere for doing work peacefully as well as resourcefully.

I would like to acknowledge Dr. Mahesh Chandra Govil (HOD, Department of CSE, MNIT, Jaipur) and Dr. Girdhari Singh (Associate Professor, Department of CSE, MNIT, Jaipur) for their continuous technical/non-technical support.

I convey special thanks to my DREC committee members Dr. Neeta Nain (MNIT, Jaipur) and Dr. D. Boolchandani (MNIT, Jaipur) for their inspiration and encouragement that directed me toward the completion of this research.

I am also grateful to Dr. Mushtaq Ahmed (MNIT, Jaipur), Mr. Ramesh B. Battula (MNIT, Jaipur), Mr. Dinesh Kumar Tyagi (MNIT, Jaipur) and Mr. Devendra Kumar Mahawer (MNIT, Jaipur) for their continuous motivation and support during PhD.

From the depth of my heart, I deeply acknowledge all those person who has really motivated me to complete the thesis. I also express my apology that I couldn't mention them in person.

Finally, in the end, I would like to thank God, the Creator of the Earth, the Almighty. The Supreme in power and knowledge and the most merciful, Whom I acquired guidance and knowledge to do something beneficial for the humanity.

Abstract

As the information content over world wide web is increasing rapidly, it becomes very difficult for users to search any required information. While searching a particular information, if a short glimpse of a long text document is provided before hand, may save the precious time of internet users. Automatic text summarization (ATS) has made it possible to condense text data by removing irrelevant information while retaining the most important information in the summary. Typically it is desired that the generated summary should express the whole content in a minimum number of words without losing the essence of the text. ATS has numerous applications such as snippet in web search engines, electronic program guide in direct-to-home (DTH) television services. It can also be applied in the domains such as medical, legal, business, and News.

A number of taxonomies are presented in the literature by researchers to classify automatic text summarization systems from different perspectives. There are two broad approaches named as extractive automatic text summarization (EATS) and abstractive automatic text summarization (AATS). Extractive text summarization selects sentences using different sentence scoring techniques from the input text and usually it is easy and simple to implement as compared to abstractive text summarization. In AATS sentences are fused and reformed using natural language generations techniques. The requirement of domain specific knowledge sources makes if difficult to use the strengths of AATS effectively. Thus, the focus of research work carried out is on extractive text summarization approaches.

The major problems in automatic text summarization are redundancy, coverage, coherence, cohesion, and anaphora. Researchers are continuously trying to resolve these issues but till date they are not been able to succeed to a great extent. In literature researchers proposed a number of extractive text summarization approaches using different methodologies such as statistical approaches, graph based approaches, machine learning approaches, evolutionary approaches and lexical chain based approaches. Statistical and graph based approaches are affected by the issues of coherence and cohesion. Issues with machine learning and evolutionary methods are of the requirement of labeled corpus and generalization. Lexical chain based methods are underutilized due to practical semantic issues such as the semantic threshold. However, statistical and graph based features do not require such extra knowledge. Most of statistical and graph based extractive summarization systems proposed in the recent literature are not able to ensemble different scoring methods in an efficient way. As ensembling different scoring approaches degrades the quality of generated text summaries. Further, all the statistical and graph based features are not analyzed properly to make feature combinations.

In the proposed work, an extensive set of statistical and graph based features is used to create feature combinations. The impact of these feature combinations is analyzed using ROUGE evaluation measures. The effect of stemming and size of the text document is also analyzed. The outcome of this work is used to generate best feature combinations that may be utilized further for the multiple level sentence scoring process. Later, best feature combinations found in impact analysis of feature combinations are used for sentence filtering process. For efficient use of feature combinations, different permutations of best feature combinations are applied for sentence filtering. Sentence filtering is further extended with voting schemes that are originally proposed for expert search systems. A new methodology for initial sentence scoring is proposed and be used further for initial ranking of the sentences of the given text. Experimental results using ROUGE evaluation measures have shown that the proposed methods perform better as compared to single feature algorithms, other state of the art statistical methods and professional tools such as MS Word and Copernic.

Contents

List of Figures

List of Tables

List of Symbols

Symbol	**Description**
D	The given text document
N	The total number of sentences in the given text document D
t	The given term
tf	Term frequency of a given term
isf	Inverse sentence frequency of a given term
d_t	The number of total distinct terms
c_t	The number of total common terms
n_t	Total terms in the whole document
S_i	Given sentence in D
V_i	Voting method

List of Acronyms

Abbreviation	**Description**
ATS	Automatic text summarization
EATS	Extractive automatic text summarization
AATS	Abstractive automatic text summarization
TFS	Term frequency score
TF-ISF	Term frequency-inverse sentence frequency score
CWS	Cue word score
GS	Gain score
WCS	Word Co-occurrence score
BPS	Busy path score
AS	Aggregate similarity score
TR	TextRank score
LR	LexRank score
TS	Title similarity score
SL	Sentence location score
NDS	Numerical data score
CS	Cosine similarity core
IWS	Interection with sentences score
SE	Sentence entropy score
FSPS	Frequential sum of probability score
HDS	Hamming distance score
HWS	Hamming weight score
SHW	Sum of Hamming weights of terms by frequency score
TTS	Total terms in the sentence score
SSCS	Sentence to sentence cohesion score
SCCS	Sentence to centroid cohesion score
DSTS	Depth of the sentence in the tree score
BFP	Best filtering permutation
BVM	Best voting method

Chapter 1

Introduction

Day by day the information content on world wide web is exponentially increasing along with the increase in the number of web users. In current scenario volume of information available on world wide web exceeded the textual information available in the printed form in libraries. The rapid growth of online information services makes it difficult to retrieve the relevant information quickly. While searching a particular information content, many a times information system users realize that the extracted information using currently available popular tools, that is present in the form of text, is not relevant to their information need at all, even after reading the whole list of text documents. They perhaps only waste their valuable time in reading the irrelevant text document. This problem can be solved if users can be provided with a summary of the given text document. However, due to a large volume of available text data, that too dynamic in nature, it is very cumbersome for human experts to summarize all the documents manually. This issue leads to the requirement of an automated system that summarize the given text document automatically. This system that condenses the text document automatically and preserving its overall information content using a computer is termed as Automatic Text Summarization (ATS) system. There are numerous applications of automatic text summarization such as a snippet in information retrieval systems, news headlines as replacement to full story of news, electronic program guide in television systems. ATS is widely used for various domains such as business, news, legal and medical domains.

The researchers have classified text summarization systems [1–7] on the basis of three main perspectives namely input, purpose and output. The third aspect which is most popular among these (i.e. output) considers summarization systems as either extractive or abstractive, and is mostly used by researchers. In extractive automatic text summarization (EATS), a subset of sentences from the original text document is selected for the final summary. Whereas abstractive automatic text summarization (AATS), sentences are fused and regenerated using natural language resources or rules. Abstractive text summarization requires deep knowledge resources, lexical/language resources, parsers and language generators. Because of these resource requirements, it is practically infeasible to use abstractive methods for automatic text summarization. Therefore, researches mainly focused on extractive text summarization instead of abstractive text summarization. The research work carried out is also focused on extractive automatic text summarization.

A typical extractive text summarization process completes in three steps namely pre-processing, sentence scoring, and summary generation. The first step in text summarization is preprocessing the input text document. Preprocessing mainly includes the sentence segmentation, stemming, stopword removal and special symbol removal. In the step which is sentence scoring, a score is assigned to each sentence of the document based on certain specific criteria. In literature various features are defined for sentence scoring such as term frequency, numerical data inclusion, sentence location, etc. After calculating each sentence score, rank is assigned to each sentence based on these scores. The sentences with the higher score are considered more important than the sentences with lower scores and are assigned high ranks consequently. After assigning a rank to each sentence, in third step, top ranked sentences are selected to generate the summary. Total number of sentences in the summary depends on the required length of it.

Major problems in automatic text summarization are redundancy, coverage, coherence, cohesion, and anaphora. Researchers are continuously trying to resolve these issues but till date they are not been able to succeed to a great extent. In final summary, selection of similar type of sentence is termed as the problem of redundancy. If information rich sentences that cover most of the topics present in the text are not selected then, the issue is termed as coverage issue. If meaningless sentences are selected in the final summary then, the issue is termed as coherence issue. In cohesion issue, the lexical similarity of contents between the selected sentences for the summary becomes very less. In anaphora problem, sentences that contain references to objects whose information is not present in the other sentences, are selected for the final summary.

Work in the area of EATS started in the early 1950s, however as of now; no such system is available that can generate summaries as efficient as experts. In early stage, pioneered features such as term frequency [8], sentence location [9], cue words and title similarity [10] were proposed for text summarization process. As the time progressed, new features were added for the text summarization process like tf-isf, gain, etc. The basic methodology behind these summarization processes is to assign score to each of the sentences present in the text by using either a single feature or linear sum of features. However, applying features alone do not improve the quality of text summaries significantly, as each feature processes sentences in a different aspect. In literature many feature based statistical, graph based, lexical, machine learning and evolutionary methods for extractive summarization are proposed by researchers.

The key component of the statistical approach of extractive text summarization is assigning weights to words or sentences based on certain statistics such as frequency of appearance of words in the text. Statistical approaches are simple to use as no knowledge source is required in these approaches. However problems with these approaches are similar to extractive text summarization approaches such as problem of ambiguous references i.e. anaphors (such as pronouns which refers to some words that appears earlier in the text) and cataphors (ambiguous words which signals to word that appears later in the text).

In graph based approaches, sentence scores are generated based on the relationship among the sentences. Although graph based approaches work well as graph based approaches do not rely only on the local context of a text unit (vertex), rather take into account information recursively drawn from the entire text (graph) [11]. However graph based approaches also have the issue of ambiguous references. In addition to this issue, graph based approaches also have an issue of the requirement of multiple iterations as these approaches require multiple iterations to converge.

Later machine learning methods and evolutionary methods were proposed with the increase number of features. These methods, used to identify a suitable set of features and their applicable weights [12]. On the trained data the performance, most of the times, is optimal; however applying the same process to unknown data may result in the generation of poor quality summaries. This issue of generation of poor quality summaries for unknown data is known as the generalization issue. Although machine learning algorithms allow to test performance with a high number of features in an easy way, however at the same time, all these approaches require labeled corpus of sentences. This requirement of labeled corpus becomes a critical issue while summarization because it is difficult to mark summary sentences similar to abstracts, manually. Also these methods are not able to resolve the issue of coherence and cohesiveness.

Semantic methods such as feature based semantic similarity and lexical chains were proposed to alleviate the issue of coherence and cohesiveness. In lexical chain based approaches, lexical chains are constructed after the pre-processing step. There after in the next step, scoring of the chains is conducted. For this purpose various matrices are applied such as chain length, chain distribution in the text, text span covered by the chain, density, graph topology (diameter of the graph of the words) and number of repetitions [13]. The main issue with lexical chain based methods is the requirement of knowledge sources on hand such as WordNet to find semantically related words and chains, and secondly interpretation of semantic information such as semantic similarity threshold. These practical issues limits and puts constraints on the usefulness of semantic methods.

In order to improve the performance of extractive text summarization systems, the information-rich sentences needs to be selected to produce the final summary. Hence sentence selection is the most important step of extractive text summarization process. For the purpose a numerical measure of usefulness may be assigned to each of the sentences in the given text. Information rich sentences are then selected according to the specified heuristics. Aforesaid weight assigning process by using a specific heuristic is termed as sentence scoring. For the final summary, sentences are selected on the basis of their scores computed using different scoring methods. Therefore, the overall performance of extractive text summarization system mainly depends on the sentence scoring methods employed to score the sentences. Several other factors like pre-processing, also influences the quality of text summaries. This thesis mainly focuses on the development of two different methodologies for sentence scoring.

In this thesis, the impact of feature combinations is analyzed on the quality of text summaries. Best feature combinations are then selected by using ROUGE [14] evaluation measures. These combinations are further used for multi level processing. Sentence filtering is applied for multi level processing using the best feature combinations. Lastly, the voting techniques are integrated with sentence filtering approach to further improve the quality of generated text summaries.

1.1 Sentence Scoring

Sentence scoring is the process of awarding scores to each of the sentences given in the text document. In case of statistical methods, scores are awarded either on the basis of a single feature or by using linear combination of multiple features.

According to the classification of features as word level, sentence level and graph level; scoring methods are also classified subsequently. In word level features, sentences are scored depending upon individual words/phrases present in the given sentence. For example in term frequency, sentence score is computed as the sum of the frequencies of each term present in the given sentence. In graph based features, sentences are scored as per their relative strengths compared to other sentences, using graph based algorithms. In sentence level scoring, sentences are scored by as per their individual properties related to sentence features such as sentence location.

1.2 Research Gaps and Motivations

Extractive text summarization methods are further classified into surface level statistical methods, graph based methods, machine learning methods, evolutionary methods and lexical chain based methods. All these methods ultimately assigns a score to each of the sentences in the given text in a different way. Therefore, sentence scoring has become the most prominent part of overall text summarization process. Different writing styles of text makes it more difficult to use a single rule or heuristic for sentence scoring. Thus, a number of heuristics are required for sentence scoring to generate efficient summaries. Based on the extensive literature survey on extractive automatic text summarization methods, following are the motivations of the research work carried out in this thesis.

The first motivation for the proposed sentence scoring method is derived from the fact that often by applying more heuristics, the quality of generated text summaries may improve. Most of other statistical methods use the linear sum of features for sentence scoring that eventually leads to incohesive text. Graph based methods overcome the issue of cohesiveness to some extent by using interrelationship between sentences. However, graph based methods fails in case if the sentences are not linked to each other. Machine learning methods and evolutionary methods compute the weight of each of the feature used for sentence scoring. However, weighting each of the features and then computing sentence scores, restricts the strength of each of the features and is known as feature weighing issue. Another problem with machine learning methods is the requirement of manually annotated corpus, which mark each sentence as either summary sentence or non-summary sentence. This sentence annotation is a difficult task as a number of different factors affect marking of the given sentences as summary or non-summary sentences. Another issue with machine learning methods and evolutionary methods is that the created model/fitness criterion performs well on the trained data, whereas performs relatively poor while applied on unknown text data, and is known as generalization issue. Lexical chain based methods depend heavily on semantic threshold values used for extracting lexical chains from the text. Because of all aforesaid issues present in machine learning methods, evolutionary methods and lexical chain based methods, leads to focus on statistical and graph based methods for sentence scoring. Therefore, different combinations of features are used to investigate their impact on quality of text summaries.

Sentence scoring methods use sentence selection mechanisms for generating text summaries. These selection methods aggregate the scores of each sentence using a specific scoring method. However, this sometimes generate less effective summaries due to the selection of less important sentences using the specific scoring method. Most of the researchers focused only on sentence selection methodologies. Instead of sentence selection if sentence rejection process is used, this might improve the performance since it is not considered generally by researchers . This is the motivation behind the use of sentence rejection method termed as sentence filtering.

The generated text summaries using specific scoring process should be cohesive while preserving its important contents. Graph based methods help in selecting sentences that are highly correlated with other sentences. However, ensembling different approaches of text summarization results in the production of poor quality summaries. Most of the specific summarization methods have few feature in common and may end up with a negative impact while ensembling. This aggregation impact motivates to use supports that are termed as votes for each sentence to be selected instead of ensembling. This might further improve the quality of generated text summaries. Thus it is imperative to investigate the hypothesis.

1.3 Objectives

The main objectives of this thesis are given below:

1. To analyze the impact of sentence scoring using feature combinations generated from prominent statistical and graph based features, on the quality of generated text summaries. The aim is to find the best feature combinations which perform better than all other combinations.

2. To improve the performance by using best feature combinations with sentence rejection, which is termed as sentence filtering.

3. To extend sentence filtering, voting mechanisms are applied to further improve the quality of text summaries.

1.4 Contributions

The major contributions of this thesis are as follows:

1. Considering the fact that a single feature can not optimize the quality of text summary, feature combinations are experimented. In order to analyze the impact of feature combinations, an extensive set of statistical and graph based features is used that consists of twenty-four different features. Prominent features are selected using ROUGE-1 [14] for generating feature combinations. As the number of possible combinations using twenty-four feature values is very large. Furthermore, best combinations are extracted which are present in the list of first forty combinations for all evaluation measures (ROUGE-1, ROUGE-2, ROUGE-L, and ROUGE-W). Along with the impact of feature algorithm combinations, the impact of stemming and size of the text document is also taken under consideration and analyzed here.

2. Different feature combinations select different sets of sentences for the final summary. Ensembling results in overall quality degradation, of the generated summary. As a solution, different permutations of best feature combinations are used for sentence filtering. Multi level processing (interest) is applied for sentence scoring. These levels are actually the elements of respective feature combinations permutation.

3. Sentence filtering is further extended by providing voting schemes that are originally proposed for expert search systems. A new methodology for initial scoring is proposed and is further used for initial ranking. The best voting methods consist of three elements that are specific voting scheme for voting, specific set of features for initial ranking and specific feature combination for sentence filtering.

In this thesis, main focus is on the improvement of the quality of extractive text summaries using statistical and graph based features. It avoids the requirement of extra knowledge and complex training procedures which are the main drawbacks of lexical chain based methods and machine learning based methods, respectively. The proposed approaches identify the best possible set of features, their combinations, filtering permutations and voting schemes. The presence of graph based features and voting schemes ensures that important thematic sentences are scored higher. Because these techniques use sentence level inter-relationship which is present in the text document itself. The effectiveness of proposed approaches is tested using ROUGE (ROUGE-1, ROUGE-2, ROUGE-L and ROGE W) [14] evaluation measures. In addition, the impact of pre-processing and size of the text document is analyzed on the quality of text summaries.

1.5 Thesis Structure

The remainder of the thesis is partitioned into three sections i.e. the introductory chapters, contributory chapters and the concluding chapter. The introductory chapters (Chapter 2 and Chapter 3) demonstrates the background required for understanding the basic concepts of extractive automatic text summarization. Features used for the proposed impact analysis and further extensions are discussed in Chapter 3. The contributory chapters include three chapters, each presenting the proposed work. Chapter 4 presents the impact analysis of feature combinations using prominent features. These combinations are termed as best feature combinations. Sentence filtering methodology using best feature combinations is presented in Chapter 5. It also presents how sentence filtering can be used with other existing methods. Chapter 6 provides the voting based summarization systems along with sentence filtering. It also introduces modified initial ranking procedure along with different voting schemes. Finally, Chapter 7 concludes the research work of this thesis and provides future directions.

Chapter 2

Automatic Text Summarization

Automatic text summarization (ATS) is a process of condensing text documents while preserving the information contents using a computer system. As text data on the internet is growing rapidly, it is practically impossible to summarize it manually. Automatic text summarization has made it possible to condense text data by removing insignificant information and keeping the most important information in the summary. Therefore, the main purpose of automatic text summarization is to generate a summary of a single text document or bunch of text documents using a computer machine. The generated summary shall express whole content in a minimum number of words without losing its information content.

In this chapter classification of text summarization systems from the different perspectives is discussed. There are two broad approaches named as extractive and abstractive text summarization. Extractive text summarization selects sentences using different sentence scoring approaches as it is from the input text that is easy and simple to implement. In abstractive text summarization, sentences are fused and reformed using natural language generations techniques. Thus, the focus of research work carried out is on extractive text summarization approaches. Further, relevant work done till date by researchers for extractive text summarization using different approaches such as statistical approaches, graph based approaches, machine learning approaches, evolutionary approaches and lexical chain based approached is discussed. After that, work done in the area of abstractive text summarization and applications of text summarization systems is discussed.

2.1 Classification of Text Summarization Systems

Different taxonomies are proposed by researchers for the classification of text summarization systems and is shown in Figure 2.1. Sparck Jones [1] in 1999 proposed a taxonomy in which text summarization systems are classified based on three factors: input, purpose and output. In this taxonomy, input factors are mainly document structure, domain, specialization level, restriction on language, scale, media, genre, unit and language [7]. Purpose factors for text summarization systems include situation, audience, and usage. The output factors consider content, format, style, production process, surrogation, length to classify text summarization systems [7]. Hovy and Lin [2] also proposed the similar taxonomy in 1999. The difference is that Hovy and Lin's taxonomy considered specific factors concerning the coherence and the subjectivity level of the summary. Mani and Maybury [3] in 1999 proposed another taxonomy that classifies text summarization systems based on the level of processing. According to this taxonomy, text summarization systems can be classified into three categories: surface level text summarization systems, entity level text summarization systems, and discourse level text summarization systems. Richard Tucker [15] also proposed a taxonomy in 2000 which classifies text summarization systems based on the four main directions: summarizing from attentional networks, sentence by sentence, from informational content, and from discourse structure. Thus, considering all the taxonomies, text summarization systems can be classified based on the following factors:

Input: Input to a text summarization system may be a single document, or there may be multiple documents. Based on this, text summarization systems can be classified into two categories: Single-Document text summarization systems and Muti-Document text summarization systems.

Purpose: Based on the purpose of text summarization, text summarization systems can be classified into various categories such as Generic, Personalized, Query-focused, Update and Sentiment-based [6].

Output: There are various output factors based on which text summarization systems can be classified. Two of the output factors are the style of the output and production process of the output. Based on the style of output, text summarization systems can be classified into four categories: informative(contains information about all of the topics in the source text), indicative(provides a brief survey of the existing topics in the source text), aggregative(contains information that is not presented in the source text) and critical(provides an additional review

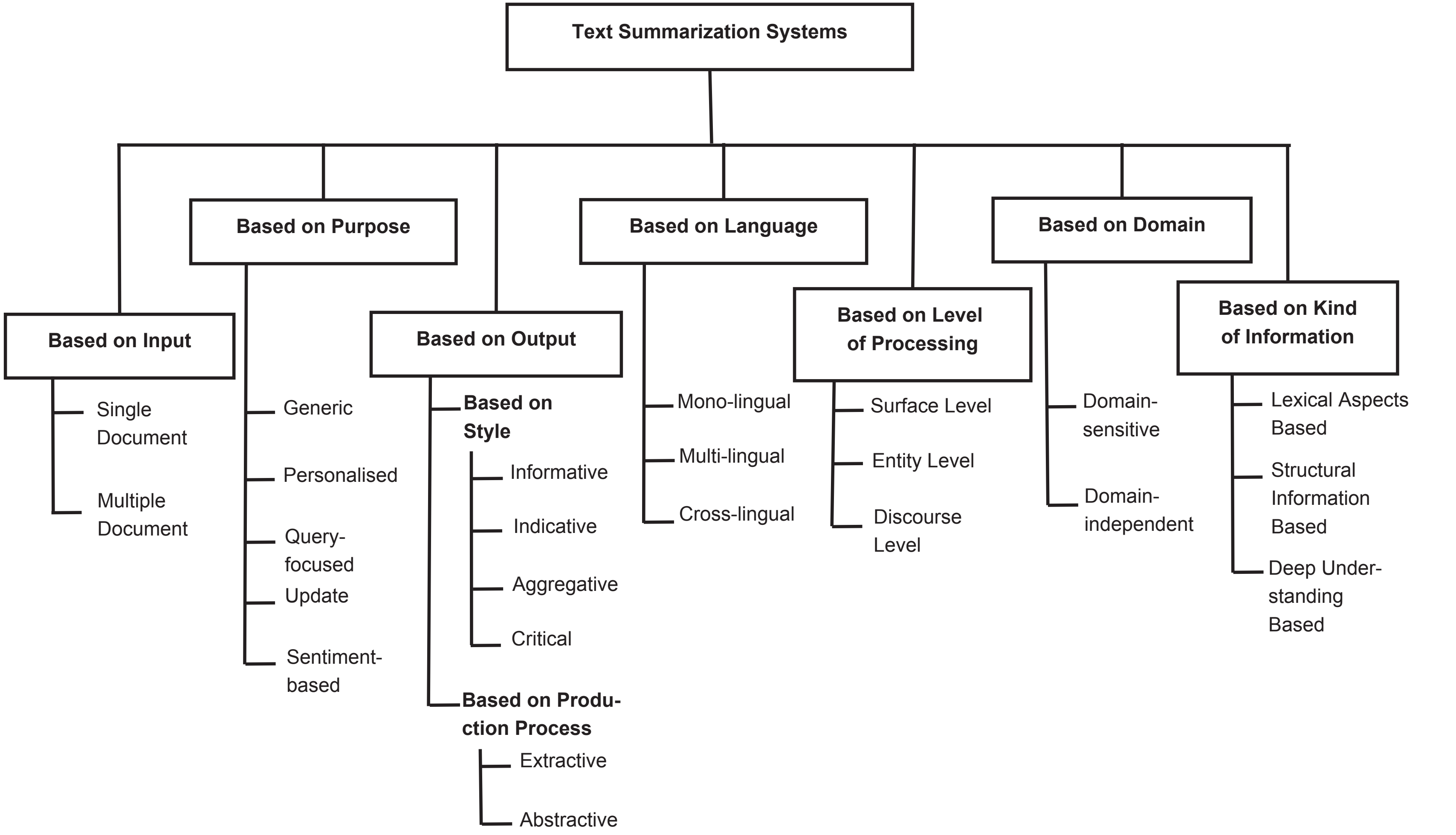


Figure 2.1: Taxonomy of text summarization systems.

of specifications of the summarized text) [7]. Based on the production process of the output, text summarization systems can be classified into two categories: Extractive text summarization systems (EATS) and Abstractive text summarization systems (AATS). In EATS, the summary is generated by extracting fragments of text from the source text. While in AATS, the summary is generated by understanding the meaning of the source text and using some natural language generation techniques.

Language: Based on language of input source text and output summary, text summarization systems can be classified into three categories: Mono-lingual text summarization systems(single language used for input text and output summary), Multi-lingual text summarization systems(input text and output summary are in the same language, but systems can generate summaries in various languages) and Cross-lingual text summarization systems(able to generate output summaries in a different language from their inputs) [7].

Level of Processing: Based on level of processing of the source text, text summarization systems are categorized into three categories: surface level text summarization systems(tends to represent the information that is achievable from shallow features existing in the text), entity level text summarization systems(tends to build an internal representation of the text by modeling text entities and their relations), and discourse level text summarization systems(uses global structure of the text to summarize).

Domain: Based on the domain of input source text, text summarization systems can be classified into two categories: Domain-sensitive text summarization systems and Domain Independent/General Purpose text summarization systems. Domain-sensitive text summarization systems are those which can summarize just those texts that belong to a predetermined domain such as SUMMONS [16] that is specialized to produce summaries in the terrorism domain [7]. However, general-purpose systems are those which can extract information in a specific domain such as Meta Summarizer [17].

Kind of Information: Based on the kind of information with which text summarization systems deal, there are three categories of text summarization systems: lexical aspects based text summarization systems, structural information based text summarization systems and deep understanding based text summarization systems. In lexical aspects based text summarization systems, information associated with words is exploited. The main idea in these approaches is that repeated information is a good indicator of importance. Structural information based text summarization systems are those that try to get information from texts as structured entities [7]. Deep understanding based text summarization systems try to achieve an understanding of the text.

As discussed already abstractive text summarization systems require deep understanding and knowledge sources to produce quality summaries. Lack of these resources and complexities involved in implementation makes it difficult for researchers to use the advantages of abstractive systems. Therefore, the primary focus of researchers is on extractive text summarization systems that are easy, simple to implement and require surface level information. Researchers focused on different extractive approaches mainly classified as statistical methods, machine learning methods, lexical chain methods, graph based methods and discourse based methods as given in various survey papers [4–7].

2.2 Extractive Text Summarization Methods

Sentences are selected without any changes, as it is from the input text document in case of extractive automatic text summarization. A number of different ways are proposed by researchers to select the sentences using extractive text summarization methods such as statistical methods, graph based methods, machine learning methods, evolutionary method and lexical chain based methods, which are discussed in the following section.

2.2.1 Extractive Generic Text Summarization

A typical extractive text summarization process completes in three steps namely pre-processing, sentence scoring and summary generation. The details of process flow are discussed in the following subsection. Whole generic text summarization process flow consisting of three steps is shown in Figure 2.2.

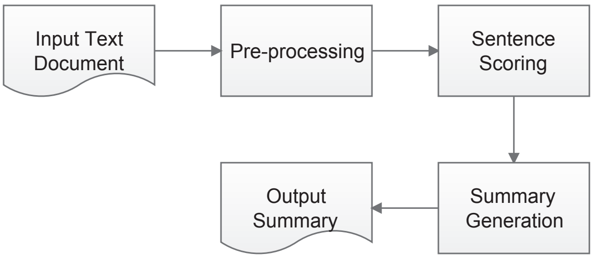


Figure 2.2: Generic text summarization process.

1. **Preprocessing:** First step in text summarization is preprocessing the input text document. Preprocessing mainly includes the following steps sentence segmentation, stemming, stopword removal and special symbol removal.

   (a) **Sentence Segmentation:** Sentence are segmented using sentence delimiting symbols such as the period ('.') and question mark ('?').

   (b) **Stemming:** Since there may be different forms of the same word in the document, so instead of considering them as different words, these words are considered as a same word. For this purpose, stemming is applied to the text document to convert each word to its root form. For example, root form of both "computed" and "computing" is "compute".

   (c) **Stopword Removal:** In stopword removal, common words such as "is, am, are, was, the, here, etc." are removed from the input text as these are most frequent words and does not contribute to the importance of sentences.

   (d) **Special Symbol Removal:** After removing stopwords from the stemmed text, special symbols such as "double quotes, comma, :, ;, (,), [,], %" are removed, as these also do not contribute to the importance of sentences.

   For example, consider a text document containing eight sentences as shown in Table 2.1.

   Table 2.1: Input text document before applying pre-processing.

S_1	At around 8 in the morning, there was a blast in the middle of the market.
S_2	According to the eyewitnesses, the bomb was placed in an abandoned shop.
S_3	Around 10 people have died and 25 are injured.
S_4	The government has formed an SIT to investigate the matter.
S_5	The government has also declared a compensation of 1 lakh to the next of kin of the dead.
S_6	The district magistrate Ramesh Kumar condemned the whole incident and said that those who have done this will we punished.
S_7	No terrorist organization has yet taken the responsibility of the blast.
S_8	However there is speculation of Indian Muzahiddin being behind the attack.

   After sentence segmentation of text document, next step is stemming. Table 2.2 shows the text document after stemming process.

Table 2.2: Input text document after applying stemming.

S_1	at around 8 in the morn , there was a blast in the middl of the market .
S_2	accord to the eyewit , the bomb was place in an abandon shop .
S_3	around 10 peopl have die and 25 are injur .
S_4	the govern has form an sit to investig the matter .
S_5	the govern has also declar a compens of 1 lakh to the next of kin of the dead .
S_6	the district magistr ramesh kumar condemn the whole incid and said that those who have done this will we punish .
S_7	no terrorist organ has yet taken the respons of the blast .
S_8	howev there is specul of indian muzahiddin be behind the attack .

After applying stemming to the input text document, stopwords and special symbols are removed from the stemmed document. Table 2.3 shows the stemmed text document after stopwords, and special symbols removal i.e. the text document after applying all the three steps of preprocessing.

Table 2.3: Stemmed text document after stopword, special symbol removal.

S_1	8 morn blast middl market
S_2	accord eyewit bomb place abandon shop
S_3	10 peopl die 25 injur
S_4	govern form sit investig matter
S_5	govern declar compens 1 lakh kin dead
S_6	district megistr ramesh kumar condemn incid punish
S_7	terrorist organ respons blast
S_8	howev specul indian muzahiddin attack

2. **Sentence Scoring:** After preprocessing, the next step in generic text summarization is sentence scoring. In this step, a score is assigned to each sentence of the document based on certain specific criteria. Various features are defined for sentence scoring such as term frequency, numerical data inclusion, sentence location, etc. After calculating each sentence score, rank is assigned to each sentence based on these scores of the sentences. The sentences with the higher score are considered as more important than the sentences with lower scores. Ranks are assigned to each sentence in the given text according to their scores.

3. **Summary Generation:** After assigning a rank to each sentence, the final step in generic text summarization is summary generation. In this step,

top ranked sentences are selected to generate the summary. The number of sentences to be selected depends on the required length of the summary.

2.2.2 Statistical Approaches

The key component of a statistical approach of extractive text summarization is assigning weights to words or sentences based on certain statistics such as frequency of appearance of words in the text. Pioneered statistical methods in summarized form are presented in Table 2.4.

Table 2.4: Summary of important statistical approaches of extractive automatic text summarization.

Author, Year	**Features**	**Theme**
Luhn [8], 1958	Term Frequency	Sentences are scored by significant terms available in the particular sentence
Baxendale [9], 1958	Sentence Location	Sentences are scored based on their position in the document
Edmundson [10], 1969	Term Frequency, Sentence Location, Title Similarity, Cue Method	In cue method, dictionary corpus of cue words is used to score the important sentences and for title similarity, a title glossary is used
Brandow et al [18], 1995	TF-ISF	It gives importance to the terms that are more frequent in the particular sentence and less frequent in the other sentences
Lin [19], 1999	Numerical Data Inclusion, Query Signature, Proper Name, Pronoun, IR signature	Different combinations of given features are tried and it is analyzed that IR Signature is an important feature
Mori [20], 2002	Gain	It was used to remove the impact of IDF that gives score to less important terms as well
Nobata et al [21], 2002	Term Frequency, Sentence Location, Title Similarity, Cue Method	It includes named entity based information extraction approach along with given features
Liu et al [22], 2009	Word Co-occurrence	It gives higher score to the sentences which contain frequently co-occurring terms

Luhn in 1958 [8] proposed term frequency as the feature to score sentences to create auto abstracts for technical literature. He used IBM 704 data processing machine for the purpose of text scanning. Stopwords that are also termed as common words are removed in advance before extracting high frequency terms. Sentences were scored as the sum of the score of all terms present in the particular sentence. Top scoring sentences are reported as summary sentences. Although the abstracts created using this procedure are indicative in nature, still it has the high degree of reliability, stability and consistency. However, Luhn himself argued that authors styles of writing may cause selection of inferior sentences in final summary.

Later Baxendale in 1958 [9] proposed sentence location as a feature to score sentences. After examining 200 paragraphs, he found that in 85% paragraphs first sentence of the paragraph is significant to be considered as the topic sentence. In only 7% paragraphs, the topic sentence is last one. Many complex machine learning text summarization methods used this feature from that time onwards. However, using location feature again as the only feature for extracting summary may not produce the optimal summary.

In 1969 Edmundson [10] proposed two additional features title similarity and cue method along with features proposed by Luhn in 1958 [8] and Baxendale in 1958 [9]. In cue method, he used a dictionary corpus of cue words to score/weight the important sentences. For title similarity, a title glossary was created which contains the words present in the title, subtitle, and headings. Stopwords are removed from this glossary. Sentences that contain more title glossary words are scored higher. The rest of the other two features were used in the same way as proposed by Luhn in 1958 [8] and Baxendale in 1958 [9]. However, experimental results [10] indicated that term frequency is dominated by the rest of the other three features. Considering all four features, the complexity of this approach was high at that time due to slow processing. The overall impact of cue word was more as compared to other features. These four features proposed are extensively used by future researchers making them pioneered features.

Brandow et al. in 1995 proposed a sentence selection method using term weighting [18]. They used tf-idf [23] score of terms to score the sentences. The intuition behind this was to capture signature words that are less frequent generally. Experiments were conducted on 41 news documents for the purpose of condensation and compared with lead (sentence location) based method. It was found that tf-idf based scoring achieves 90% acceptability compared to 74.4% of lead (sentence location) based [9] scoring.

Later in 1999 Lin [19] tried to use decision trees and suggested many different features instead of relying on a single feature. In his paper, he described experiments with the SUMMARIST system. He evaluated his summaries against human generated summaries, some of the features that he used were numerical data inclusion (checks for presence of numerical values in the sentence), query signature (checks for presence of words from the query in the sentence), proper name (checks for presence of proper names in the sentence), pronoun presence, information retrieval (IR) signature (checks for presence of signature words or important words in the sentence). He tried many different combinations of these features. After that he analyzes that, IR signature is an important feature.

Mori in 2002 [20] used term information gain to remove the impact of IDF that assign score to less important terms as well. It was originally proposed for multi-document summarization. Hierarchical clustering was used for making clusters of similar sentences using term information gain. It was observed by experiments that this method is very effective in the case of retrieved documents using a query. However, using this feature alone is less effective depending upon writing style of authors.

Nobata et al. [21] proposed a method that includes named entity based information extraction approach along with four features named term frequency, cue word, title similarity and sentence location [10]. DUC 2001 dataset was used for experiments, and it was observed that the proposed approach performs better as compared to baseline-lead based and other systems of DUC 2001, specifically for cohesion. Liu et al. 2009 [22] proposed an approach of weighting terms using word co-occurrence. They used top frequent terms for generic summarization to compare them using n-gram overlap to other sentences. This technique gives the higher score to the sentences that contain frequently co-occurring terms.

Statistical approaches are simple to use as no knowledge source is used in these approaches. Problems with these approaches are only those which are with all extractive text summarization approaches such as problem of ambiguous references i.e. anaphora (such as pronouns which refers to some words that appears earlier in the text) and cataphora (ambiguous words which signals to word that appears later in the text).

2.2.3 Graph Based Approaches

In graph based approaches, sentence scores are generated based on the relationship among the sentences. First the text document is preprocessed and segmented into sentences. Each sentence of the document represents the node of the graph and edge between these nodes represents the similarity between the sentences. The edges can be weighted or unweighted. In unweighted approach if two sentences have terms in common then there is an edge that connects the nodes. In the weighted approach, the weights of the edges are computed by simply measuring the common terms between sentences or by using cosine similarity. The sentences with the highest similarity to the other sentences are selected to include in the summary. Important methods that use graph based approaches for text summarization are summarized in Table 2.5. Salton in 1997 [24] proposed bushy path and aggregate similarity to extract the summary from text documents. These two features used graph characteristics for summarization process. Intra-document links between passages are generated using these two features. These intra-document links ultimately identify linked sentences as thematic sentences that are finally selected as summary sentences. 50 documents were used for evaluation purpose. Observed results were satisfactory because two humans can not generate the identical summaries. No comparisons were made with standard methods. These features again have their own merits and demerits. Busy path performance is affected by the issue of cohesion as it takes care of coverage whereas aggregate similarity is affected by coverage problem.

Table 2.5: Summary of graph based approaches of extractive automatic text summarization.

Author, Year	**Features**	**Theme**
Salton [24], 1997	Bushy Path, Aggregate Similarity	These features used graph characteristics and generates intra-document links between passages and then identifies linked sentences as thematic sentences
Mihalcea et al [11], 2004	TextRank	Graph based page ranking approach is used to score sentences
Erkan et al [25], 2004	LexRank	Instead of simply using word overlap, idf modified cosine similarity is used to compute the link between the sentences

TextRank was proposed by Mihalcea et al. in 2004 [11] taking advantage of graph based page ranking approach. Sentences were considered as nodes and page ranking procedure is applied. Top ranked sentences were finally used for summary generation. TextRank was also termed as a voting method, where other sentences in actual votes a particular sentence to become a top ranked sentence. Experimental results were conducted on a standard dataset, and it was observed that having the advantage of links in between sentences, the performance of TextRank was better as compared to other previous state of the art methods. However, the absence of cohesive text degrades the performance of the method.

LexRank was proposed in 2004 by Erkan et al. [25]. It is also a graph based method. Instead of simply using word overlap, LexRank uses idf modified cosine similarity to compute the link in between the sentences. Page rank method is then used for further ranking of sentences. However, this method also performs in lower range when text units are not connected cohesively similar to TextRank.

Graph based approach is more useful in case there are multiple topics in the document as compared to very few topics. Sentences related to the same topic thus form a cluster that is represented by a disconnected subgraph. Each disconnected subgraph in the graph represents different topics. Thus distinct topics covered in the document can be extracted. To calculate the summary a representative sentence from each of the subgraphs can be chosen. These representative sentences are those which are connected to most of the sentences in the subgraph.

Although graph based approaches works well because it does not only rely on the local context of a text unit (vertex), rather it takes into account information recursively drawn from the entire text (graph) [11]. However, still there are various issues with these approaches. Graph based approaches also have same issues as with extractive text summarization such as ambiguous references. In addition to these issues, graph based approaches also have an issue of the requirement of multiple iterations as these approaches require multiple iterations to converge.

2.2.4 Machine Learning Approaches

With the increase in the number of features and heuristics for selection of important sentences, it was felt that there should be ways using which these indicators could be combined to utilize them adequately. Important machine learning methods are summarized in Table 2.6. To overcome this issue Kupic et al. in 1995 [12] proposed a machine learning based approach. For this purpose, they used cue word, paragraph position and term frequency features. They proposed two new features: sentence length cut off that take care of exclusion of sentences whose length is below a particular threshold and upper case feature that includes sentences containing proper names. They used Naive Bays classifier assuming that features are independent of each other. For training purpose, they used technical articles. As the training data should be labeled, different annotation techniques were used to mark the sentences as important or not using exact match, incomplete, partial match or combination of sentences with a variety of matching techniques. The results showed that features such as sentence location, cue phrase and sentence length in combined form performs better. Whereas, adding term frequency to this combination degrades the performance of the summarization system.

Table 2.6: Summary of important machine learning approaches of extractive automatic text summarization.

Author, Year	**Features Used**	**Learning Theme**
Kupic et al [12], 1995	Cue Word, Paragraph Position, Term Frequency, Sentence Length Cut Off, Upper Case	Naive Bayes classifier is used assuming that features are independent to each other
Aone et al [26], 1998	Sentence Length, tf-isf and Position of Sentences in Paragraph and Document	Naive Bayes classifier was used for learning
Conroy and O'Leary [27], 2001	Position of Sentence, Number of Words in the Sentence and Term Probability	It uses Hidden Markov Model (HMM)
NTT [28], 2002	Position, Length, Weight, Presence of Verb and Title Similarity	It uses support vector machines to classify the sentences as to be included or not in final summary
NetSum [29], 2007	Sentence Position, Keywords, Terms and Wikipedia Entities	It uses neural network based Query learning method, RankNet [30]
Schilder and Kondadadi [31], 2008	Title Similarity, Cue Word, Topic Description Word, Term Frequency, Headline Frequency, Sentence Length and Sentence Position	It is a support vector machine based approach
Wong et al [32], 2008	Content, Event and Restiveness	It combines supervised and semi-supervised learning algorithms

Aone et al. [26] used sentence length, tf-isf score and position of sentences in paragraph and document. Bayesian method was again used for learning. Cue word feature was not used by them. Later, Conroy and O'Leary in 2001 [27] used Hidden Markov Model (HMM) with features such as the position of the sentence, the number of words in the sentence and term probability. They suggest that probability of selecting a sentence for summary depends on the already selected sentences in the summary. NTT method [28] used support vector machines to classify the sentences as to be included or not in final summary using position, length, weight, the presence of verb and title similarity features. Svore et al. [29] in 2007 proposed NetSum for single document summarization using neural network based learning method RankNet [30]. Other than the features used such as sentence position and keywords they used query terms and Wikipedia entities as features.

A support vector machine based approach was proposed by Schilder and Kondadadi in 2008 [31]. Features such as title similarity, cue word, topic description word, term frequency, headline frequency, sentence length, sentence position (discrete and continuous) are used in for sentence scoring. Wong et al. in 2008 [32] proposed an approach combining supervised and semi-supervised learning algorithms. Content, event, and restiveness features were used for training. Although the experimental results are satisfactory, supervised algorithms require labeled data. Methods such as Restricted Boltzmann Machine, KNN, Decision Trees are also used for summarization process. Although machine learning algorithms allows to test performance of a high number of features in an easy way, however, all these approaches require labeled corpus of sentences. This labeling becomes a critical issue because it is difficult to mark summary sentences similar to abstracts manually. Another issue is that only domain specific training can give efficient results, as these specific feature can classify them in a higher dimensional space utilizing machine learning algorithms. Also, these algorithms are not portable in nature.

2.2.5 Evolutionary Approaches

Genetic Algorithms, a first evolutionary algorithm given by Holland et al. [33], is the most powerful optimization technique in a large solution space. In 1992, Vafaie et al. [34] proved GAs as a useful tool for solving difficult feature selection problems where both the size of the feature set and the performance of the underlying system are crucial for text summarization task. Silla in 2004 [35] investigated the effectiveness of genetic algorithm-based attribute selection to improve the performance of classification algorithms by solving automatic text summarization task. Afterward, Genetic Programming began with the evolutionary algorithms. It was first used by Nils Aall Barricelli. He applied it to evolutionary simulations. Genetic Programming (GP), [36] is an evolutionary paradigm for automatically finding solutions for a problem. The process flow is shown in Figure 2.3.

It is more sophisticated than GAs in terms of representation of problem space. The idea of using GP for text summarization tasks was first proposed by Xie et al. [37] and employed Gene Expression Programming as sentence ranking module. Price et al. in 2006 [38], replaced the classical crossover and mutation operators in GA by alternative operators and eventually, they came up with a suitable differential operator to handle the problem. They proposed a new algorithm based on this operator and called it Differential Evolution (DE). In 2009, Khosravi et al. [39] used all features in combination to train genetic programming (GP), vector approach, and fuzzy approach to constructing a text summarizer for each model. Cordon et al in 2004 [40], suggested Logic and Evolutionary Algorithms based method in which they implemented fuzzy logic for representation and inference of text with the extended Boolean query structure and applied multi-objective evolutionary algorithms to construct the fuzzy query system.

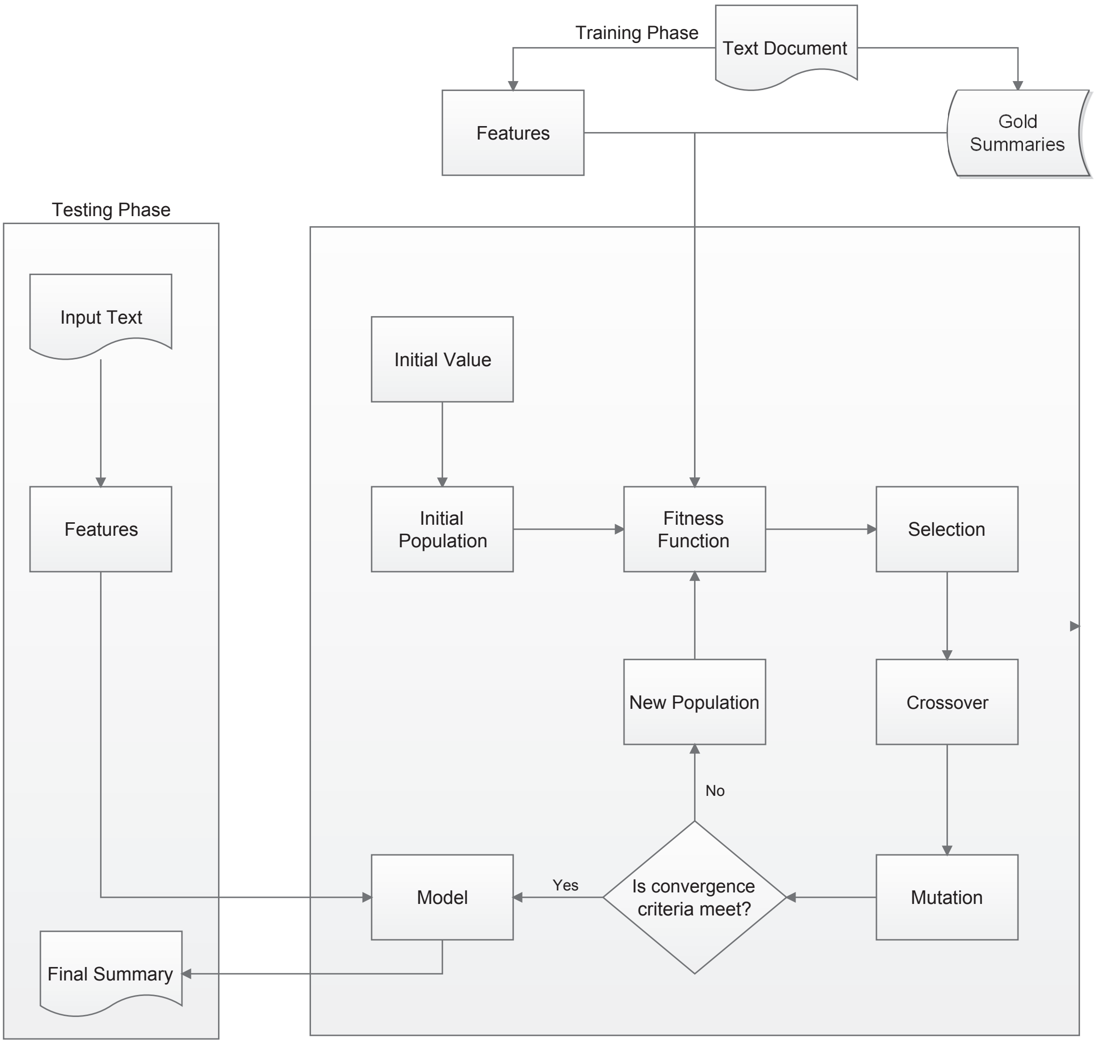


Figure 2.3: Process flow of evolutionary methods for extractive automatic text summarization.

In Hybrid fuzzy GA-GP methodology [41], GA has been used for string part (membership function) while GP has been used for the structural part in fuzzy logic. Fuzzy inference system has been used for selecting sentences based on their attributes and locations in the article. It has been used to remove any uncertainty and ambiguity in selecting values. The sentences were ranked in descending order, and top n sentences were selected as the final summary. Lamprier et al. in 2009 [42] proposed an algorithm called SenGen, which segments texts into homogeneous parts based on some thematic features; the process is based on two criteria: maximization of the internal cohesion of the formed segments and minimization of the similarity of the adjacent segments. Its objective is to segment texts into thematic homogeneous parts so that genetic algorithm can be applied after that with fitness function as the internal cohesion of sentences. Binwahlan et al. in 2009 [43], proposed Fuzzy logic with Particle Swarm Optimization. They incorporated fuzzy logic with swarm intelligence to avoid risk in choosing the vague values of feature weights (scores). Song et al. in 2011 [44] proposed a fuzzy evolutionary optimization model (FEOM) that simultaneously cluster the documents and generate summaries for the respective document. The method involves the concept of clustering the sentences of documents. According to fitness value, sentences are selected from each of the clusters to generate the final summary.

Aristoteles et al. in 2012 [45] presented an extra feature of sentence semantics, which can be determined using singular value decomposition technique (SVD). It can be calculated by forming a matrix of sentences and total terms available in the corpus i.e. it is same as finding the term co-occurrence matrix. Differential Evolution-Cluster-based Method [46] employed DE to optimize the data clustering process and to increase the quality of the generated text summaries. Mendoza et al. in 2014 [47] proposed a memetic algorithm approach known as MA-SingleDocSum to optimize the linear combination of the features. However, the issue of generalization similar to machine algorithms exists in the case of evolutionary methods as well.

2.2.6 Lexical Chain based Approaches

The assumption of lexical chain based approaches is that term frequency alone is not a very good measure to identify important parts of a text, as it does not show any connection between the words and simply works on the basis of their presence. However, if the relationship between the words is known, a better summarization system might be created using cohesion as the important property. Cohesion is a technique for sticking together different parts of the text. Cohesion is achieved through the use of semantically related terms, co-reference, ellipsis, and conjunctions. Among all these ways, finding semantically related terms is the easiest, and lexical cohesion is created by using semantically related words. Lexical cohesion can occur among a sequence of related words and not only between two related words. When there is cohesion between a sequence of words, it is called a lexical chain. Lexical chain based approaches find lexical chains in the source document.

Important lexical chain based methods are summarized in Table 2.7. Barzilay and Alhadad [13] in 1997 proposed an approach using lexical chains. They used the algorithms proposed by Hirst et. al. [48] in 1995 and Stairmand et. al. [49] in 1996 to compute lexical chains in the source text. Both of these algorithms use WordNet to find semantically related words and chains. A typical process flow for text summarization using lexical chain is shown in Figure 2.4.

Table 2.7: Summary of important lexical chain based approaches of extractive automatic text summarization.

Author, Year	**Theme**
Hirst et al [48], 1995	They proposed a method of computing lexical chain using WordNet to find semantically related words
Stairmand et al [49], 1996	They proposed another method of computing lexical chains using WordNet
Barzilay and Alhadad [13], 1997	Topic identification of the text is done by grouping words into lexical chains
Brunn et al [50], 2001	They described the summarizer of the University of Lethbridge at the DUC 2001
Carthy et al [51], 2002	A lexical chaining based topic tracking system, LexTrack, is presented
Moldovan et al [52], 2002	They proposed a method to find topically related words on an extended word-net
Galley et al [53], 2003	According to this, quality of chains can be increased by separating Word Sense Disambiguation(WSD) from the actual chaining of words
Doran et al [54], 2004	LexSum system based on a greedy lexical chaining approach is presented
Medelyan and Olena [55], 2007	A method of computing lexical chains using graph clustering is presented

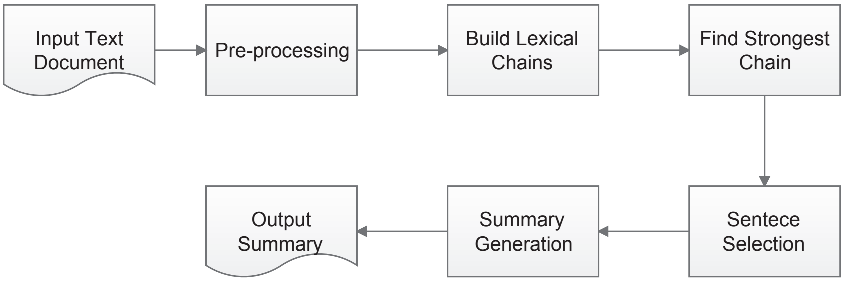


Figure 2.4: Lexical chain based generic text summarization process.

In lexical chain based approaches, after the pre-processing lexical chain is constructed. After constructing lexical chains, the next step is scoring the chains. For this purpose various metrics are used such as chain length, chain distribution in the text, text span covered by the chain, density, graph topology (diameter of the graph of the words) and number of repetitions [13]. The optimal value for all these metrics was calculated by manually evaluating some text documents. After the scoring of lexical chains, sentences were extracted on the basis of some heuristics such as for each chain in the summary representation, choose the sentence that contains the first appearance of a chain member in the text [13].

Later in 2004 Doran et al. [54] proposed a LexSum system that uses a greedy lexical chaining approach. First they did Part-of-Speech tagging (POS) of the source document, then they identify all the nouns, proper nouns, and noun phrases. The nouns and proper nouns are used as candidate words for lexical chaining. In their work, they produced two different chains one for each noun and proper nouns.

Each word pairs score is calculated as the sum of the frequencies of the two words, multiplied by the relationship score between them [54]. Then the algorithm ranks the sentences on the basis of the words present. The words score is a scaled score of its chains score. The distance between the word and its related words in the chain is used as the scaling value. The highest scored sentences are included in the summary.

In 2001, Brunn et al. [50] described the summarizer of the University of Lethbridge at the DUC 2001. Carthy et al. in 2002 [51] proposed a topic tracking system, LexTrack, which is based on lexical chaining. In 2002, Moldovan et al. [52] also presents a method to find topically related words on an extended wordnet. Galley et al. in 2003 [53] proposed a lexical chain based method. They suggest that quality of chains can be increased by separating Word Sense Disambiguation(WSD) from the actual chaining of words. In 2007, Medelyan and Olena [55] presented a new method for computing lexical chains using graph clustering. The main issue with lexical chain based methods is the requirement of knowledge source such as WordNet to find semantically related words and chains.

2.3 Abstractive Text Summarization

Abstractive summarization picks relevant information from the document and generates new sentences for the summary. Sentence compression, sentence fusion or natural language generation(NLG) are used for abstractive summarization. Broadly, abstraction summarization techniques can be classified in structure based [56–59] and semantic based [60, 61] techniques. Few of the important methods are summarized in Table 2.8. Tree based, rule based, template based techniques comes under structure based techniques and INIT based, Graph based techniques comes under semantic based techniques [62, 63]. In tree based approach, each sentence is converted into dependency tree rooted at the verb and these different trees are merged, if possible [64].

Semantic graph based method construct a semantic graph of the document and then reduce this graph. After that, the summary is generated from this reduced graph i.e. reduced semantic graph RSG [61]. In RSG, the verbs and nouns of the input document are represented as graph nodes along with edges corresponding to semantic and topological relations between them. A set of heuristic rules is applied to reduce the graph by replacing, deleting, or consolidating the graph nodes using the WordNet relations, then generate the abstractive summary from the reduced

Table 2.8: Summary of important methods for abstractive automatic text summarization.

Author, Year	**Theme**
Zhou et al [57], 2004	Headline length summary is generated using template based approach
Barzilay [65], 2005	Multidocument summaries is generated using sentence fusion, a text-to-text generation technique
Tanaka et al [59],2009	A new method for revision of lead sentences in a news broadcast is proposed
Ganesan et al [58], 2010	Opinosis, a graph-based summarization framework is proposed to generate abstractive summaries of highly redundant opinions
Genest et al [60], 2011	A framework is presented to generate abstract summaries using abstract representation of text documents based on concept of Information Items (INIT)
Genest et al [56], 2012	An approach of abstractive text summarization is proposed to generate abstract summaries based on Information Extraction and Natural Language Generation
Moawad et al [61], 2012	Abstractive summary of single document is generated using semantic graph reducing technique
Kikuchi et al [64], 2014	An approach is proposed to generate single document abstractive summary based on nested tree structure

rich semantic graph. Summary generated by tree and graph based approach are less abstractive because it contain sentences that are present in the text. This is so because these approach uses the concept of compression and fusion [65].

In lead and body phrase method, same phrases are searched in lead and body sentences [59]. Then these phrases are aligned using similarity metric. If the body phrase has rich information with the corresponding phrase, then we substitute the body phrase for the lead phrase. However, if body phrase has no counterpart, then insertion takes place.

INIT based technique extract information items (INIT) such as a single word or phrase from the document and populate these INITs [60] with subject-verb-object(S-V-O) triplets and also associate date and location if any. After that, these triplets are passed to the NLG, which generate sentences for the triplets. Sentence generated from this technique were properly formed.

Rule based approach uses the concept of abstraction scheme and generates short and well written abstractive summaries from clusters of news articles on same event [56]. The abstraction scheme uses a rule based information extraction module, content selection heuristics and one or more patterns for sentence generation. Each abstraction scheme deals with one theme or subcategory. For generating extraction rules for abstraction scheme, several verbs and nouns having similar meaning are determined, and their position is also identified. The information extraction (IE) module finds several candidate rules for each scheme of the category. Based on the output of the IE module, the content selection module selected the best candidate rule for each aspect and passed it to summary generation module. This module forms summary of text using generation patterns designed for each abstraction scheme. The strong point of this method is that it has a potential for creating summaries with greater information density than current state of art.

In template based approach, a template is created after analyzing training data [57]. After that relevant information is selected from the given text document to be summarized and filled into the empty slots. This approach produces highly coherent and condensed summaries, but much important content is neglected if it does not fit in the template slots. Nowadays researchers are focusing onto abstractive text summarization because summary produced by abstractive text summarization are more coherent, less redundant and rich in information. Though generating summary using abstractive summarization methods is a complex task since it requires more semantic and linguistic analysis.

2.4 Applications of Text Summarization

Various applications of text summarization are as following:

Legal Texts: Text summarization can be used to generate summaries of long legal documents as per requirement. [66]. Farzindar et al. in 2004 [67], proposed an approach to generate very short table style summary for a long legal document by exploring the documents architecture and thematic structures. In 2004, Hachey et al. [68] also proposed a classifier to determine the rhetorical status of sentences in texts from a corpus of judgments of the UK House of Lords.

Emails: Text summarization can be used to summarize emails [66]. Corston-Oliver et al. in 2004 [69], proposed a prototype system, SmartMail, which automatically identifies action items in email messages. The SmartMail system generates a task-focused summary of a message containing a list of action items extracted from the message [69]. Shrestha et al. in 2004 [70], proposed an approach to detect question-answer pairs in an email conversation using various features based on the structure of email threads. In 2004, Wan et al. [71] also proposed an approach of generating a summary of ongoing email discussions using the structure of the email threads. This approach uses word vector techniques for determining the sentences that should be extracted.

Web Pages: Web Pages can also be summarized using text summarization [66]. Diao et al. in 2006 [72], proposed an algorithm to summarize multiple web pages. In this algorithm, graph based ranking algorithm is used in addition to Maximum Marginal Relevance (MMR) to eliminate the redundancy from the summary sentences [72].

Web documents using mobile devices: Summary of web documents can also be generated from mobile devices [66]. Otterbacher et al. in 2006 [73], proposed a method using which Web documents summaries can be viewed on mobile devices. The proposed method summarizes plain text format news articles sent to a Web mail account.

News: Text summarization also helps in summarizing news articles [66]. McKeown et al in 2003 [74], proposed a system, Columbias Newsblaster. The proposed system first clusters news into events and then categorizes these events into broad topics. After that, it summarizes multiple articles on each event. [74]. Nenkova et al. in 2005 [75] also proposed an approach to improve machine summaries using knowledge about the cognitive status of news article referents. In 2005, Evans et al. [76] proposed an approach to summarizing documents from two different sources, English and machine translated Arabic texts.

Geographical Information Retrieval: Summarization can be used in geographical information retrieval systems as an intermediate stage, to reduce the document length. Thus, improving the access time for information searching and relevant documents will also be retrieved [77]. Perea-Ortega et al. in 2013 [77], proposed an approach to generate two types of summaries: generic and geographical.

2.5 Summary

In this chapter automatic summarization systems are discussed in detail. Further, classification of summarization systems from different perspectives is discussed. Thereafter, different extractive and abstractive methods are reviewed. It was observed that combinations of statistical and graph based features are not used in an efficient way to generate quality summaries.

Chapter 3

Features Used for Sentence Scoring

A typical sentence scoring module of extractive automatic text summarization requires various features to score sentences. A number of statistical and linguistics features have been proposed by researchers for the aforesaid scoring process. These features are categorized into three group of categories according to their processing level. The categories are word level scoring, sentence level scoring and graph based scoring and these are described in the following sections.

3.1 Word level Scoring

The features used for word level scoring methods score each term present in the given text according to a particular criterion. Sentence score is computed as the sum of each term present in a given sentence. Important word level scoring features are discussed in the following subsection.

3.1.1 Term Frequency

Luhn [8] proposed the feature term frequency in 1958 to create auto abstracts for technical documents. This feature computes the frequency of the each term in the whole text document. Luhn stated that the most frequent words in a document (excluding stop words) were the most important words, and they convey maximum information. Therefore, term frequency is important for sentence scoring. To improve the efficiency of this feature, stop words are removed well in advance. The term frequency for a given term t_i can be calculated as given in Equation 3.1. Sentence score is computed as the sum of all terms present in the given sentence, excluding stop words as given in Equation 3.2. After that normalized score of sentence S_i, $NTFS(S_i)$ is calculated as given in Equation 3.3.

$$tf(t_i, D) = \text{No of Occurrences of } t_i \text{ in } D \tag{3.1}$$

$$TFS(S_i) = \sum_{j=1}^{d_t(S_i)} tf(t_j, D) \tag{3.2}$$

$$NTFS(S_i) = \frac{TFS(S_i)}{Max_{TFS}} \tag{3.3}$$

where:
D: is the given text document
$d_t(S_i)$: is the number of total distinct terms of S_i
Max_{TFS} : is maximum TFS among all sentences in D

For example, term frequency score of sentence S_2 given in Subsection 2.2.1 is calculated as:

$$\begin{aligned} TFS(S_2) &= tf(accord, D) + tf(eyewit, D) + tf(bomb, D) + tf(place, D) \\ &\quad + tf(abandon, D) + tf(shop, D) \\ &= 1 + 1 + 1 + 1 + 1 + 1 \\ &= 6 \end{aligned} \tag{3.4}$$

As sentence S_5 has maximum TF score, 8. Therefore, the normalized TF score of sentence S_2 is:

$$\begin{aligned} NTFS(S_2) &= \frac{6}{8} \\ &= 0.75 \end{aligned} \tag{3.5}$$

3.1.2 Term Frequency-Inverse Sentence Frequency

The feature term frequency-inverse sentence frequency is a special version of inverse document frequency (IDF) [18]. IDF suggests that the terms that are dense in the given document and rare in the document set are most important. However, in case of single document summarization, inverse sentence frequency (ISF) is used instead of IDF as given in Equation 3.6. Term frequency is multiplied by ISF value to get the $tf\text{-}isf$ score of each term and the same is given in Equation 3.7. Similar to term frequency stop words are removed in advance. Sentence score is calculated as sum of each term's $tf\text{-}isf$ score, present in the respective sentence as given in Equation 3.8. After that normalized score of sentence S_i, $NTF\text{-}ISF(S_i)$ is calculated as given in Equation 3.9.

$$isf(t_i) = log_e \frac{N}{N(t_i)} \tag{3.6}$$

$$tf\text{-}isf(t_i, S_i) = tf(t_i, S_i) * isf(t_i) \tag{3.7}$$

$$TF\text{-}ISF(S_i) = \sum_{j=1}^{d_t(S_i)} tf\text{-}isf(t_j, S_i) \tag{3.8}$$

$$NTF\text{-}ISF(S_i) = \frac{TF\text{-}ISF(S_i)}{Max_{TF\text{-}ISF}} \tag{3.9}$$

where:
N : is the total number of sentences in the given text document D
$N(t_i)$: is the number of sentences in D which contains the term t_i
$tf(t_i, S_i)$: is term frequency of term t_i in sentence S_i
$d_t(S_i)$: is the number of total distinct terms of S_i
$Max_{TF\text{-}ISF}$: is maximum TF-ISF among all sentences in D

For example, tf-isf of term morn given in Subsection 2.2.1 is calculated as:

$$\begin{aligned} isf(morn) &= log_e(\frac{8}{1}) \\ &= 2.0794 \end{aligned} \tag{3.10}$$

$$\begin{aligned} tf\text{-}isf(morn, S_2) &= tf(morn, S_2) * isf(morn) \\ &= 1 * 2.0794 = 2.0794 \end{aligned} \tag{3.11}$$

Now, tf-isf of sentence S_2 is calculated as:

$$\begin{aligned} TF\text{-}ISF(S_2) &= tf\text{-}isf(accord, S_2) + tf\text{-}isf(eyewit, S_2) + tf\text{-}isf(bomb, S_2) \\ &\quad + tf\text{-}isf(place, S_2) + tf\text{-}isf(abandon, S_2) + tf\text{-}isf(shop, S_2) \\ &= 2.0794 + 2.0794 + 2.0794 + 2.0794 + 2.0794 + 2.0794 \\ &= 12.4764 \end{aligned} \tag{3.12}$$

As sentence S_6 has maximum TF-ISF score, 14.5561. Therefore, the normalized TF-ISF score of sentence S_2 is:

$$\begin{aligned} NTF\text{-}ISF &= \frac{12.4764}{14.5561} \\ &= 0.8571 \end{aligned} \tag{3.13}$$

3.1.3 Cue Words

The cue method [10] is based on the hypothesis that the sentences which contain pragmatic words such as “significantly”, “impossible”, “hardly” and starting with “in summary”, “in conclusion”, “our investigation”, “in short”, “the paper describes”, etc. are more probable to be included in the summary. This method uses a pre stored dictionary of cue words, which are discovered from manual summaries. A sentence S_i containing these cue words/phrases gets a higher score as compared to other sentences in the text document, using the formula given in Equation 3.14. After that normalized score of sentence S_i, $NCWS(S_i)$ is calculated as given in Equation 3.15.

$$CWS(S_i) = NCW(S_i) \tag{3.14}$$

$$NCWS(S_i) = \frac{CWS(S_i)}{Max_{CWS}} \tag{3.15}$$

where:
CWS(S_i) : is the cue word score of the sentence S_i
NCW(S_i) : is the number of cue words/phrases present in the sentence S_i
Max_{CWS} : is maximum CWS among all sentences in D

For example, sentence S_2 as given in Subsection 2.2.1 contains a cue word. Therefore,

$$CWS(S_2) = 1 \tag{3.16}$$

As sentence S_5 and S_6 has maximum CWS, 2. Therefore, the normalized CWS of sentence S_2 is:

$$\begin{aligned} NCWS(S_2) &= \frac{1}{2} \\ &= 0.5 \end{aligned} \tag{3.17}$$

3.1.4 Gain

In a typical IDF method, words that occur very scarcely in the corpora get a very high score, however sometimes their importance is very less. The feature gain [20] overcomes this weakness of IDF by introducing a new measure, as given in Equation 3.18. Gain score of term t_i for sentence S_i is calculated as given in Equation 3.19. Sentence score is calculated as the sum of Gain of each term present in the given text document as given in Equation 3.20. After that normalized score of sentence S_i, $NGS(S_i)$ is calculated as given in Equation 3.21.

$$Gain(t_i) = \frac{N(t_i)}{N}\left[\frac{N(t_i)}{N} - 1 - log_e\frac{N(t_i)}{N}\right] \tag{3.18}$$

$$GS(t_i, S_i) = log_e((1{+}tf(t_i, S_i))/Gain(t_i)) * N \tag{3.19}$$

$$GS(S_i) = \sum_{j=1}^{d_t(S_i)} GS(t_i, S_i) \tag{3.20}$$

$$NGS(S_i) = \frac{GS(S_i)}{Max_{GS}} \tag{3.21}$$

where:
$Gain(t_i)$: is the gain of term t_i
N : is the total number of sentences in the given text document D
$N(t_i)$: is the number of sentences in D which contains the term t_i
$GS(t_i, S_i)$: Gain score of term t_i for sentence S_i
$tf(t_i, S_i)$: is term frequency of term t_i in sentence S_i
$d_t(S_i)$: is the number of total distinct terms of S_i
Max_{GS} : is maximum GS among all sentences in D

For example, GS of term morn for sentence S_2 given in Subsection 2.2.1 is calculated as:

$$\begin{aligned} Gain(morn) &= \frac{1}{8} * (\frac{1}{8} - 1 - log_e(\frac{1}{8})) \\ &= 0.125 * (0.125 - 1 - (-2.0794)) \\ &= 0.15056 \end{aligned} \tag{3.22}$$

$$\begin{aligned} GS(morn, S_2) &= log_e((1{+}1)/0.15056) * 8 \\ &= 2.5866 * 8 \\ &= 20.6926 \end{aligned} \tag{3.23}$$

Now, GS of sentence S_2 is calculated as:

$$\begin{aligned} GS(S_2) = {} & GS(accord, S_2){+}GS(eyewit, S_2){+}GS(bomb, S_2) \\ & +GS(place, S_2){+}GS(abandon, S_2){+}GS(shop, S_2) \\ & = 20.6926{+}20.6926{+}20.6926{+}20.6926{+}20.6926{+}20.6926 \\ & = 124.1556 \end{aligned} \tag{3.24}$$

As sentence S_6 has maximum GS, 144.8481. Therefore, the normalized GS of sentence S_2 is:

$$\begin{aligned} NGS(S_2) &= \frac{124.1556}{144.8481} \\ &= 0.8571 \end{aligned} \tag{3.25}$$

3.1.5 Named Entity

The presence of named entities [21] in a sentence, suggests potential candidate sentences to be included in the final summary. Sentence score is computed according to the presence of total named entities in it. Normalized score for a particular sentence S_i which is named as NE(S_i) and is calculated as given in Equation 3.26.

$$\text{NE}(S_i) = \frac{\text{NNE}(S_i)}{\text{NNE}(D)} \tag{3.26}$$

where:
NNE(S_i) : is the total named entities in the sentence S_i
NNE(D) : is the total named entities present in D

3.1.6 Word Co-occurrence

According to the feature word co-occurrence, thematic words (most frequent words excluding stop words), if they co-occur [22](WCooc) in the sentences, then higher weight should be given to the respective sentences. Word co-occurrence score of a given sentence S_k, which is named as WCS(S_k) and calculated as given in Equation 3.27. After that normalized score of sentence S_k, $NWCS(S_k)$ is calculated as given in Equation 3.29.

$$WCS(S_k) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} p(t_i, t_j, S_k) \tag{3.27}$$

$$p(t_i, t_j, S_k) = \begin{cases} 1 & \text{if } t_i \in S_k \text{ and } t_j \in S_k \text{ , where } (t_i, t_j) \in T_{top} \\ 0 & \text{otherwise} \end{cases} \tag{3.28}$$

$$NWCS(S_k) = \frac{WCS(S_k)}{Max_{WCS}} \tag{3.29}$$

where:

T_{top} : is the set of n most frequent word in D

$p(t_i, t_j, S_k)$: represents the presence of t_i and t_j in the given sentence S_k

Max_{WCS} : is maximum WCS among all sentences in D

For example, consider top 10 most frequent terms as given in Table 3.1 for WCS calculation. Now, WCS of sentence S_2 is calculated as:

Table 3.1: Top ten most frequent terms

Most Frequent Terms	tf
blast	2
govern	2
8	1
morn	1
middl	1
market	1
accord	1
eyewit	1
bomb	1
place	1

$$\begin{aligned} WCS(S_2) &= \sum_{i=1}^{10} \sum_{j=i+1}^{10} p(t_i, t_j, S_2) \\ &= p(accord, eyewit, S_2) + p(accord, bomb, S_2) + p(accord, place, S_2) \\ &\quad p(eyewit, bomb, S_2) + p(eyewit, place, S_2) + p(bomb, place, S_2) \\ &= 1 + 1 + 1 + 1 + 1 + 1 \\ &= 6 \end{aligned} \tag{3.30}$$

3.2 Graph Level Scoring

The sentence score in case of graph level scoring method is calculated using the relationship of a sentence with other sentences. If two sentences are linked via some defined parameter, then an edge is established in between them. Their weights are used to generate sentence scores.

3.2.1 Bushy Path

In the feature bushy path [24] a sentence which has maximum number of sentences related to it with a particular similarity measure, is treated as the most important sentence in the given text document. The sentences are considered as nodes of the graph, and related sentences (the sentences that have common terms or some other similarity criterion) have edges between them. The sentence with a maximum number of edges is considered as the most informative sentence. Edge weight between two sentences S_i and S_j which is named as $EW(S_i, S_j)$ and calculated as given in Equation 3.31. The bushy path score BPS(S_i)of a sentence S_i is calculated as given in Equation 3.32. After that normalized score of sentence S_i, $NBPS(S_i)$ is calculated as given in Equation 3.34.

$$EW(S_i, S_j) = c_t(S_i, S_j) \tag{3.31}$$

$$BPS(S_i) = \sum_{j=1, j\neq i}^{N} SCount(S_i, S_j) \tag{3.32}$$

$$SCount(S_i, S_j) = \begin{cases} 1 & \text{if } EW(S_i, S_j) > threshold \\ 0 & \text{otherwise} \end{cases} \tag{3.33}$$

$$NBPS(S_i) = \frac{BPS(S_i)}{Max_{BPS}} \tag{3.34}$$

where:
$c_t(S_i, S_j)$: is the number of common terms of S_i and S_j
N : is the total number of sentences in the given text document D
$threshold$: is the threshold value in the range 0 to 1
$EW(S_i, S_j)$: is the edge weight between S_i and S_j
Max_{BPS} : is maximum BPS among all sentences in D
$SCount(S_i, S_j)$: represents existence of a link in between S_i and S_j for a given

$threshold$

For example, sentence S_1 given in Subsection 2.2.1 has only one term i.e. blast common with sentence S_7. Therefore,

$$EW(S_1, S_7) = 1 \tag{3.35}$$

Also,

$$SCount(S_1, S_7) = 1 \tag{3.36}$$

Now, BPS of sentence S_1 is calculated as:

$$\begin{aligned} BPS(S_1) &= \sum_{j=1, j\neq 1}^{N} SCount(S_1, S_j) \\ &= 1 \end{aligned} \tag{3.37}$$

As sentence S_1,S_4,S_5 and S_7 has maximum BPS, 1. Therefore, the normalized BPS of sentence S_1 is:

$$\begin{aligned} NBPS(S_1) &= \frac{1}{1} \\ &= 1 \end{aligned} \tag{3.38}$$

3.2.2 Aggregate Similarity

The bushy path scoring method counts the number of nodes that are linked to a particular node. The aggregate similarity is defined as the sum of similarity values of a particular node to each of the other nodes related to it [24]. Edge weight can be computed as given in Equation 3.31. Aggregate similarity value can be calculated as the sum of these edge weights for a particular node (sentence S_i) as given in Equation 3.39. For normalization, Aggregate similarity value of sentence is divided by $d_t(D)$ as given in Equation 3.39.

$$AS(S_i) = \frac{\sum_{j=1, j\neq i}^{N} EW(S_i, S_j)}{d_t(D)} \tag{3.39}$$

where:
N : is the total number of sentences in the given text document D

$d_t(D)$: is the total number of distinct terms in document D

For example, AS of sentence S_1 given in Subsection 2.2.1 is calculated as:

$$\begin{aligned} AS(S_1) &= \frac{1}{44} \\ &= 0.0227 \end{aligned} \tag{3.40}$$

3.2.3 TextRank

TextRank is based on the most popular ranking algorithm [11] used for web page ranking. The graph weights are calculated using term overlap between sentences as given in Equation 3.41 for node S_i and S_j.

$$EW(S_i, S_j) = \frac{c_t(S_i, S_j)}{log_e(n_t(S_i)) + log_e(n_t(S_j))} \tag{3.41}$$

where:
$EW(S_i, S_j)$: is weight of edge between node S_i and S_j
$c_t(S_i, S_j)$: is the number of common terms of S_i and S_j
$n_t(S_i)$: is the total number of terms of sentence S_i
$n_t(S_j)$: is the total number of terms of sentence S_j

Let $G = (V, E)$ be a directed graph with the set of vertices V and the set of edges E. Page rank algorithm is applied on this graph, which in turn gives a sequence of most important sentences. The page rank score of the vertex V_i is computed as given in Equation 3.42. Sentence symbol S_i is applied to vertex V_i in graph based model. When considering sentences as nodes of graph this page rank is considered as text rank. To stable the text rank values of each node, the same procedure of text rank calculation is repeated for a fixed number of times. After that normalized text rank of vertex V_i, $NTR(V_i)$ is calculated as given in Equation 3.43.

$$TR(V_i) = (1-d) + d * \sum_{V_j \in In(V_i)} \frac{EW(V_j, V_i)}{\sum_{V_k \in Out(V_j)} EW(V_j, V_k)} * TR(V_j) \tag{3.42}$$

$$NTR(V_i) = \frac{TR(V_i)}{Max_{TR}} \tag{3.43}$$

where:
d : is the damping factor (range 0 to 1)
$In(V_i)$: is the set of vertices that are predecessors of given vertex V_i
$Out(V_i)$ be the set of vertices that are successorsof given vertex V_i
$EW(V_i, V_j)$: is the edge weight between vertex V_i and V_j
Max_{TR} : is maximum TR among all sentences in D

For example, sentence S_1 given in Subsection 2.2.1 has only one term i.e. blast common with sentence S_7. Also, sentence S_1 has 5 terms and sentence S_7 has 4 terms. Therefore,

$$\begin{aligned} EW(S_1, S_7) &= \frac{1}{log_e 5 + log_e 4} \\ &= 0.3338 \end{aligned} \tag{3.44}$$

Now, TR of sentence S_1 for 1^{st} iteration is calculated as:

$$\begin{aligned} TR(V_1) &= (1-d) + d * \frac{EW(S_1, S_7) * TR(V_7)}{EW(S_1, S_7)} \\ &= (1 - 0.85) + 0.85 * \frac{0.3338 * 0.5}{0.3338} \\ &= 0.575 \end{aligned} \tag{3.45}$$

3.2.4 LexRank

LexRank [25] is also a graph based ranking model. This approach models the document as a graph and uses a PageRank algorithm similar to textrank to find top-ranked sentences for the summary generation. The difference between LexRank and TextRank comes while computing edge weight in between vertices. The edge weight between two vertices V_i and V_j is defined by the ISF modified cosine similarity between two corresponding vectors as given in Equation 3.46. After computing the similarity between sentences, lexrank of sentences is calculated as given in Equation 3.47. To stable the lexrank values of each node, the same procedure of lexrank calculation is repeated for a fixed number of times. After that normalized lexrank of vertex V_i, $NLR(V_i)$ is calculated as given in Equation 3.48.

$$CS_{isf}(V_i, V_j) = \frac{\sum_{d=1}^{d_t(V_i,V_j)} tf(t_d, V_i) * tf(t_d, V_j) * isf(t_d)^2}{\sqrt{\sum_{p=1}^{d_t(V_i)} (tf(t_p, V_i) * isf(t_p))^2 * \sum_{q=1}^{d_t(V_j)} (tf(t_q, V_j) * isf(t_q))^2}} \tag{3.46}$$

$$LR(V_i)=\frac{d}{N}+(1-d)*\sum_{V_j\in adj[V_i]}\frac{CS_{isf}(V_i,V_j)}{\sum_{V_k\in adj[V_j]}CS_{isf}(V_j,V_k)}*LR(V_j) \tag{3.47}$$

$$NLR(V_i)=\frac{LR(V_i)}{Max_{LR}} \tag{3.48}$$

where:
$d_t(V_i,V_j)$: is the number of total distinct terms of V_i and V_j
$d_t(V_i)$: is the number of total distinct terms of V_i
$d_t(V_j)$: is the number of total distinct terms of V_j
tf(t_d,V_j) : is the term frequency of term t_d in sentence V_j
isf(t_i) : is the inverse sentence frequency of term t_i
$LR(V_i)$: is LexRank of vertex V_i
d : is the damping factor (range 0 to 1)
$CS_{isf}(V_i,V_j)$: is ISF modified cosine similarity between vectors V_i and V_j
N : is the total number of sentences in the given text document D
adj[V_i] : is the set of the vertices that are neighbors of V_i in the graph
Max_{LR} : is maximum LR among all sentences in D

For example, ISF modified cosine similarity between vectors V_1 and V_7 is calculated as:

$$\begin{aligned}CS_{isf}(V_1,V_7)&=\frac{1.39*1.39}{\sqrt{2.08^2+2.08^2+1.39^2+2.08^2+2.08^2}*\sqrt{1.39^2+2.08^2+2.08^2+2.08^2}}\\&=0.1136\end{aligned} \tag{3.49}$$

Now, LR of vertex V_1 for 1^{st} iteration is calculated as:

$$\begin{aligned}LR(V_1)&=\frac{0.85}{8}+(1-0.85)*\frac{0.1136}{0.1136}*0.5\\&=0.1813\end{aligned} \tag{3.50}$$

3.3 Sentence Level

The features used for sentence level scoring methods use sentence level statistics for computing score for a given sentence. These features used for sentence level sentence scoring are described in the following subsection.

3.3.1 Title Similarity

Title similarity [10] or sentence resemblance to the title of the given text document is the vocabulary/term overlap between the given sentence and the document title [78–82]. In this feature, sentences similar to the title i.e. sentences that include the words present in the title are considered important. For a given sentence S_i title similarity score $TS(S_i)$ is calculated as given in Equation 3.51. After that normalized score of sentence S_i, $NTS(S_i)$ is calculated as given in Equation 3.52.

$$TS(S_i) = c_t(S_i, T) \tag{3.51}$$

$$NTS(S_i) = \frac{TS(S_i)}{Max_{TS}} \tag{3.52}$$

where:
$c_t(S_i, T)$: is the number of common terms of S_i and document title T
Max_{TS} : is maximum TS score among all sentences in D

For example, consider title of the given text document as given in Table 3.2.

Table 3.2: Title before preprocessing.

Blast threatened city, 10 died 25 injured.

Now, preprocessing is applied to the title. Stemmed title is given in Table 3.3 and title after removal of stopword and special symbol is given in Table 3.4.

Table 3.3: Stemmed title.

blast threaten citi , 10 die 25 injur

Table 3.4: Title after stopwords and special symbol removal.

blast threaten citi 10 die 25 injur

After preprocessing, distinct title terms of title is given in Table 3.5.

Table 3.5: Distinct title terms with their TF in title.

Title Words	TF
blast	1
threaten	1
citi	1
10	1
die	1
25	1
injur	1

Since there is one title word, blast in sentence S_1 given in Subsection 2.2.1. Therefore,

$$TS(S_1) = 1 \tag{3.53}$$

As sentence S_3 has maximum TS, 4. Therefore, the normalized TS of sentence S_1 is:

$$\begin{aligned} \text{Normalized } TS(S_1) &= \frac{1}{4} \\ &= 0.25 \end{aligned} \tag{3.54}$$

3.3.2 Sentence Location

This feature assigns a score to each sentence as per its location [9]. The first sentence always gets the highest priority in this feature. For sentence S_i sentence location score $SL(S_i)$ is defined as given in Equation 3.55.

$$SL(S_i) = \begin{cases} \frac{1}{i} & \text{if } i \leq k \\ 0 & \text{otherwise} \end{cases} \tag{3.55}$$

where:
k : is the position priority span defined by user

For example, if we take sentence position span, k as 3, then

$$SL(S_1) = \frac{1}{1} = 1 \tag{3.56}$$

$$SL(S_2) = \frac{1}{2} = 0.5 \tag{3.57}$$

$$SL(S_3) = \frac{1}{3} = 0.3333 \tag{3.58}$$

Rest of the other sentences are scored as 0.

3.3.3 Numerical Data

The presence of numerical data [82] such as date, time, money transaction, percent or some other number also signifies that a sentence is important. The sentences that contain numerical data are given higher weights. For a particular sentence S_i numerical data score $NDS(S_i)$ is calculated as given in Equation 3.59.

$$\text{NDS}(S_i) = \begin{cases} 1 & \text{if NND}(S_i)\text{>0} \\ 0 & \text{otherwise} \end{cases} \tag{3.59}$$

where:
NND(S_i) : is the total number of numerical terms present in sentence S_i

For example, there is one numerical term i.e. 8 is present in sentence S_1 for the text given in Subsection 2.2.1. Therefore,

$$NDS(S_1) = 1 \tag{3.60}$$

3.3.4 Cosine Similarity With Title

Cosine similarity [83] is a measure to find similarity between two vectors. To calculate a sentence score using this feature, first sentence and title T of the document are represented in the form of a vector and then cosine similarity between the sentence and title is calculated. Score of i^{th} sentence S_i is calculated as given in

Equation 3.61. After that normalized score of sentence S_i, $NCS(S_i, T)$ is calculated as given in Equation 3.62.

$$CS(S_i, T) = \frac{\sum_{d=1}^{d_t(S_i,S_j)} tf(t_d, S_i) * tf(t_d, T)}{\sqrt{\sum_{p=1}^{d_t(S_i)} tf(t_p, S_i)^2 * \sum_{q=1}^{d_t(S_j)} tf(t_q, T)^2}} \tag{3.61}$$

$$NCS(S_i, T) = \frac{CS(S_i, T)}{Max_{CS}} \tag{3.62}$$

where:
$d_t(S_i, S_j)$: is the number of total distinct terms of sentence S_i and title T
$d_t(S_i)$: is the number of total distinct terms of S_i
$d_t(T)$: is the number of total distinct terms of title T
tf(t_d, S_i) : is the term frequency of term t_d in sentence S_i
tf(t_d, T) : is the term frequency of term t_d in title T
Max_{CS} : is maximum CS among all sentences in D

For example, distinct terms of sentence S_1 given in Subsection 2.2.1 and title with their corresponding TF is given in Table 3.6.

Table 3.6: Distinct terms of S_1 and title with their corresponding TF.

Distinct Terms	$tf(t_i, S_1)$	**TF in Title**
8	1	0
morn	1	0
blast	1	1
middl	1	0
market	1	0
threaten	0	1
citi	0	1
10	0	1
die	0	1
25	0	1
injur	0	1

Now, CS Score of sentence S_1 is calculated as:

$$\begin{aligned} CS(S_1,T) &= \frac{1*1}{\sqrt{1^2+1^2+1^2+1^2+1^2}*\sqrt{1^2+1^2+1^2+1^2+1^2+1^2+1^2}} \\ &= \frac{1}{\sqrt{5}*\sqrt{7}} \\ &= 0.1690 \end{aligned} \tag{3.63}$$

As sentence S_3 has maximum CS, 0.6761. Therefore, the normalized CS of sentence S_1 is:

$$\begin{aligned} NCS(S_1,T) &= \frac{0.1690}{0.6761} \\ &= 0.25 \end{aligned} \tag{3.64}$$

3.3.5 Interaction With Sentences

In the feature interaction of the sentences [83], interaction of a term in all the sentences is considered. To calculate a sentence score, presence of each term is checked in all the other sentences except that sentence. Score of i^{th} sentence S_i is calculated as given in Equation 3.65. After that normalized score of sentence S_i, $NIWS(S_i)$ is calculated as given in Equation 3.67.

$$IWS(S_i) = \sum_{j=1}^{n_t(D)} \sum_{k=1,k\neq i}^{N} p(t_j,S_k) \tag{3.65}$$

$$p(t_j,S_k) = \begin{cases} 1 & \text{if } t_j \in S_k \\ 0 & \text{otherwise} \end{cases} \tag{3.66}$$

$$NIWS(S_i) = \frac{IWS(S_i)}{Max_{IWS}} \tag{3.67}$$

where:

N : is total number of sentences in document D

$n_t(D)$: is total number of terms in given document D

$p(t_j,S_k)$: represents the presence of t_j in the given sentence S_k

Max_{IWS} : is maximum IWS score among all sentences in D

For example, IWS Score of sentence S_1 given in Subsection 2.2.1 is equal to the number of terms present in all the other sentences S_2 to S_8. i.e.

$$\begin{aligned} IWS(S_1) &= 6+5+5+7+7+4+5 \\ &= 39 \end{aligned} \tag{3.68}$$

As sentence S_7 has maximum IWS, 40. Therefore, the normalized IWS of sentence S_1 is:

$$\begin{aligned} NIWS(S_1) &= \frac{39}{40} \\ &= 0.975 \end{aligned} \tag{3.69}$$

3.3.6 Sentence Entropy

In the feature sentence entropy [83], sentences having more entropy as compared to others sentences are considered as more important. Sentence score is calculated by computing the entropy of that sentence. Probability of occurrence of a term t_j in sentence S_i is calculated as given in Equation 3.70. Score of i^{th} sentence S_i is calculated as given in Equation 3.71. After that normalized score of sentence S_i, $NSE(S_i)$ is calculated as given in Equation 3.72.

$$P(t_j, S_i) = \frac{tf(t_j, S_i)}{\sum_{k=1}^{N} tf(t_j, S_k)} \tag{3.70}$$

$$SE(S_i) = -\sum_{j=1}^{d_t(S_i)} P(t_j, S_i) * log_2 P(t_j, S_i) \tag{3.71}$$

$$NSE(S_i) = \frac{SE(S_i) - Min_{SE}}{Max_{SE} - Min_{SE}} \tag{3.72}$$

where:
P(t_j, S_i) is probability of occurrence of term t_j in sentence S_i
$d_t(S_i)$: is the number of total distinct terms of S_i
tf(t_j, S_i) : is the term frequency of term t_j in sentence S_i
Min_{SE} : is minimum SE among all sentences in D
Max_{SE} : is maximum SE among all sentences in D

For example, SE Score of sentence S_1 given in Subsection 2.2.1 is calculated as given in Table 3.7. Here, $tf(t_j, S_1)$+1 is used in place of $tf(t_j, S_1)$ for simplification purpose.

Table 3.7: SE score calculation for sentence S_1.

Term(t_j)	$tf(t_j, S_1)$	$tf(t_j, S_1)$+1	$\sum_{k=1}^{N} tf(t_j, S_k)$	$P(t_j, S_1)$	$log_2(P(t_j, S_1))$	-($P(t_j, S_1)$*$log_2(P(t_j, S_1))$)
8	1	2	1	2	1	-2
morn	1	2	1	2	1	-2
blast	1	2	2	1	0	0
middl	1	2	1	2	1	-2
market	1	2	1	2	1	-2
accord	0	1	1	1	0	0
eyewit	0	1	1	1	0	0
bomb	0	1	1	1	0	0
place	0	1	1	1	0	0
abandon	0	1	1	1	0	0
shop	0	1	1	1	0	0
10	0	1	1	1	0	0
peopl	0	1	1	1	0	0
die	0	1	1	1	0	0
25	0	1	1	1	0	0
injur	0	1	1	1	0	0
govern	0	1	2	0.5	-1	0.5
form	0	1	1	1	0	0
sit	0	1	1	1	0	0
investig	0	1	1	1	0	0
matter	0	1	1	1	0	0
declar	0	1	1	1	0	0
compens	0	1	1	1	0	0
1	0	1	1	1	0	0
lakh	0	1	1	1	0	0
kin	0	1	1	1	0	0
dead	0	1	1	1	0	0
district	0	1	1	1	0	0
magistr	0	1	1	1	0	0
ramesh	0	1	1	1	0	0
kumar	0	1	1	1	0	0
condemn	0	1	1	1	0	0
incid	0	1	1	1	0	0
punish	0	1	1	1	0	0
terrorist	0	1	1	1	0	0
organ	0	1	1	1	0	0
respons	0	1	1	1	0	0
howev	0	1	1	1	0	0
specul	0	1	1	1	0	0
indian	0	1	1	1	0	0
muzahiddin	0	1	1	1	0	0
attack	0	1	1	1	0	0
					SE(S_1) =	-7.5

As sentence S_7 has maximum SE, -5.5 and sentence S_6 and S_8 has minimum SE, -13.0. Therefore, the normalized SE of sentence S_1 is:

$$\begin{aligned} NSE(S_1) &= \frac{-7.5-(-13.0)}{-5.5-(-13.0)} \\ &= 0.7333 \end{aligned} \tag{3.73}$$

3.3.7 Frequential Sum of Probability

In the feature frequential sum of probability [83], sentence score is calculated based on the probability of occurrence of terms in the whole document. To calculate sentence score, first the probability of occurrence of terms in the entire document is computed. The likelihood of occurrence of j^{th} term in the whole document is calculated as given in Equation 3.74. Sentence score of a given sentence S_i is calculated as given in Equation 3.75. After that normalized score of sentence S_i, $NFSPS(S_i)$ is calculated as given in Equation 3.76.

$$P(t_j) = \frac{1}{n_t(D)} \sum_{i=1}^{N} tf(t_j, S_i) \tag{3.74}$$

$$FSPS(S_i) = \sum_{j=1}^{d_t(S_i)} P(t_j, D) * tf(t_j, S_i) \tag{3.75}$$

$$NFSPS(S_i) = \frac{FSPS(S_i)}{Max_{FSPS}} \tag{3.76}$$

where:
tf(t_j, S_i) : is the term frequency of term t_j in sentence S_i
P(t_j, D) is probability of occurrence of term t_j in given document D
N : is the total number of sentences in the given text document D
$n_t(D)$: is the total terms in the whole document calculated as $\sum_{i=1}^{N} \sum_{j=1}^{d_t(D)} tf(t_j, S_i)$
$d_t(D)$: is the total number of distinct terms in document D
$d_t(S_i)$: is the number of total distinct terms of S_i
Max_{FSPS} : is maximum FSPS among all sentences in D

For example, FSPS of sentence S_1 is calculated as given in Table 3.8.

Table 3.8: FSPS calculation for sentence S_1.

Term (t_j)	$tf(t_j, S_1)$	$\sum_{i=1}^{N} tf(t_j, S_1)$	$P(t_j, D)$	$P(t_j, D) * tf(t_j, S_i)$
8	1	1	0.0227	0.0227
morn	1	1	0.0227	0.0227
blast	1	2	0.0455	0.0455
middl	1	1	0.0227	0.0227
market	1	1	0.0227	0.0227
accord	0	1	0.0227	0
eyewit	0	1	0.0227	0
bomb	0	1	0.0227	0
place	0	1	0.0227	0
abandon	0	1	0.0227	0
shop	0	1	0.0227	0
10	0	1	0.0227	0
peopl	0	1	0.0227	0
die	0	1	0.0227	0
25	0	1	0.0227	0
injur	0	1	0.0227	0
govern	0	2	0.0455	0
form	0	1	0.0227	0
sit	0	1	0.0227	0
investig	0	1	0.0227	0
matter	0	1	0.0227	0
declar	0	1	0.0227	0
compens	0	1	0.0227	0
1	0	1	0.0227	0
lakh	0	1	0.0227	0
kin	0	1	0.0227	0
dead	0	1	0.0227	0
district	0	1	0.0227	0
magistr	0	1	0.0227	0
ramesh	0	1	0.0227	0
kumar	0	1	0.0227	0
condemn	0	1	0.0227	0
incid	0	1	0.0227	0
punish	0	1	0.0227	0
terrorist	0	1	0.0227	0
organ	0	1	0.0227	0
respons	0	1	0.0227	0
howev	0	1	0.0227	0
specul	0	1	0.0227	0
indian	0	1	0.0227	0
muzahiddin	0	1	0.0227	0
attack	0	1	0.0227	0
threaten	0	0	0	0
citi	0	0	0	0
				FSPS(S_1) = 0.1364

As sentence S_5 given in Subsection 2.2.1 has maximum FSPS, 0.1818. Therefore, the normalized FSPS of sentence S_1 is:

$$\begin{aligned} NFSPS(S_1) &= \frac{0.1364}{0.1818} \\ &= 0.75 \end{aligned} \tag{3.77}$$

3.3.8 Hamming Distance

Hamming distance [83] measures the distance between pair of words. Hamming distance between two terms t_i and t_j is calculated as given in Equation 3.78. After calculating hamming distance between pair of words, sentence score is calculated as given in Equation 3.79. After that normalized score of sentence S_i, $NHDS(S_i)$ is calculated as given in Equation 3.81.

$$HD(t_i, t_j) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k=1}^{N} np(t_i, t_j, S_k) \tag{3.78}$$

$$HDS(S_i) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} HD(t_i, t_j) \quad \text{if } t_i \& t_j \in S_i. \tag{3.79}$$

$$np(t_i, t_j, S_k) = \begin{cases} 1 & \text{if } p(t_i, S_k) \neq p(t_i, S_k) \\ 0 & \text{otherwise} \end{cases} \tag{3.80}$$

$$NHDS(S_i) = \frac{HDS(S_i)}{Max_{HDS}} \tag{3.81}$$

where:
N : is total number of sentences in given document D
$np(t_i, t_j, S_k)$: represents the inequality presence of t_i and t_j in S_k
Max_{HDS} : is maximum HDS among all sentences in D
For example, HDS of sentence S_4 given in Subsection 2.2.1 is calculated as:

$$\begin{aligned} HDS(S_4) &= \text{HD(govern,form)} + \text{HD(govern,sit)} + \text{HD(govern,investig)} \\ &\quad + \text{HD(govern,matter)} + \text{HD(form,sit)} + \text{HD(form,investig)} \\ &\quad + \text{HD(form,matter)} + \text{HD(sit,investig)} + \text{HD(sit,matter)} \\ &\quad + \text{HD(investig,matter)} \\ &= 1 + 1 + 1 + 1 + 0 + 0 + 0 + 0 + 0 + 0 \\ &= 4 \end{aligned} \tag{3.82}$$

As sentence S_5 has maximum HDS, 6. Therefore, the normalized HDS of sentence S_4 is:

$$\begin{aligned} NHDS(S_4) &= \frac{4}{6} \\ &= 0.6667 \end{aligned} \tag{3.83}$$

3.3.9 Hamming Weight

Hamming weight [83] of a sentence is defined by the lexicon of that sentence. Each sentence is assigned a score equal to its lexicon. Sentences having higher Hamming weight are considered as more important. Hamming weight of i^{th} sentence is defined as given in Equation 3.84. After that normalized score of sentence S_i, $NHWS(S_i)$ is calculated as given in Equation 3.86.

$$HWS(S_i) = \sum_{j=1}^{d_t(S_i)} p(t_j, S_i) \tag{3.84}$$

$$p(t_j, S_i) = \begin{cases} 1 & \text{if } t_j \in S_i \\ 0 & \text{otherwise} \end{cases} \tag{3.85}$$

$$NHWS(S_i) = \frac{HWS(S_i)}{Max_{HWS}} \tag{3.86}$$

where:
$d_t(S_i)$: is the total number of distinct terms of sentence S_i
$p(t_j, S_i)$: represents the presence of t_i in the given sentence S_i
Max_{HWS} : is maximum HWS among all sentences in D

For example, sentence S_2 given in Subsection 2.2.1 contains 6 distinct terms. Therefore,

$$HWS(S_2) = 6 \tag{3.87}$$

As sentence S_5 and S_6 has maximum HWS, 7. Therefore, the normalized HWS of sentence S_2 is:

$$\begin{aligned} HWS(S_2) &= \frac{6}{7} \\ &= 0.8571 \end{aligned} \tag{3.88}$$

3.3.10 Sum of Hamming Weight of terms by Frequency

In the feature sum of hamming weights of terms by frequency [83], sentence score is calculated by summing Hamming weight of terms present in the given sentence, multiplied by their term frequency. Score of i^{th} sentence S_i is calculated as given in Equation 3.89. After that normalized score of sentence S_i, $NSHW(S_i)$ is calculated as given in Equation 3.90.

$$SHW(S_i) = \sum_{j=1}^{d_t(S_i)} HWS(S_i) * tf(t_j, S_i) \tag{3.89}$$

$$NSHW(S_i) = \frac{SHW(S_i)}{Max_{SHW}} \tag{3.90}$$

where:
$d_t(S_i)$: is the total number of distinct terms of sentence S_i
tf(t_j, S_i) : is the term frequency of term t_j in sentence S_i
Max_{SHW} : is maximum SHW among all sentences in D

For example, SHW Score of sentence S_2 given in Subsection 2.2.1 is calculated as:

$$\begin{aligned} SHW(S_2) = {} & \text{HWS}(S_2) * tf(\text{accord},S_2) + \text{HWS}(S_2) * tf(\text{eyewit},S_2) \\ & + \text{HWS}(S_2) * tf(\text{bomb},S_2) + \text{HWS}(S_2) * tf(\text{place},S_2) \\ & + \text{HWS}(S_2) * tf(\text{abandon},S_2) + \text{HWS}(S_2) * tf(\text{shop},S_2) \\ = {} & 6*1 + 6*1 + 6*1 + 6*1 + 6*1 + 6*1 \\ = {} & 36 \end{aligned} \tag{3.91}$$

As sentence S_5 and S_6 has maximum SHW, 49. Therefore, the normalized SHW of sentence S_2 is:

$$\begin{aligned} NSHW(S_2) &= \frac{36}{49} \\ &= 0.7347 \end{aligned} \tag{3.92}$$

3.3.11 Total Terms in Sentence

In the feature total terms in the sentence, a sentence score is calculated based on its length i.e. number of words present in that sentence. Longer sentences are considered as more important than the smaller ones. Score of i^{th} sentence(i.e.

$TTS(S_i))$ is calculated as given in Equation 3.93. After that normalized score of sentence S_i, $NTTS(S_i)$ is calculated as given in Equation 3.94.

$$TTS(S_i) = \sum_{j=1}^{d_t(S_i)} tf(t_j, S_i) \tag{3.93}$$

$$NTTS(S_i) = \frac{TTS(S_i)}{Max_{TTS}} \tag{3.94}$$

where:

$d_t(S_i)$: is the total number of distinct terms of sentence S_i

tf(t_j, S_i) : is the term frequency of term t_j in sentence S_i

Max_{TTS} : is maximum TTS among all sentences in D

For example, sentence S_2 given in Subsection 2.2.1 contains total 6 terms. Therefore,

$$TTS(S_2) = 6 \tag{3.95}$$

As sentence S_5 and S_6 has maximum TTS, 7. Therefore, the normalized TTS of sentence S_2 is:

$$\begin{aligned} NTTS(S_2) &= \frac{6}{7} \\ &= 0.8571 \end{aligned} \tag{3.96}$$

3.3.12 Sentence to Sentence Cohesion

In the feature sentence to sentence cohesion proposed by Neto et al [84], sentences with a higher degree of cohesion with other sentences are considered as more relevant. To calculate the sentence score, the similarity between the given sentence and each other sentence of the given text document is calculated. Score of i^{th} sentence S_i is calculated as given in Equation 3.97. After that normalized score of sentence S_i, $NSSCS(S_i)$ is calculated as given in Equation 3.98.

$$SSCS(S_i) = \sum_{j=1, j \neq i}^{N} c_t(S_i, S_j) \tag{3.97}$$

$$NSSCS(S_i) = \frac{SSCS(S_i)}{Max_{SSCS}} \tag{3.98}$$

where:
$c_t(S_i, S_j)$: is the number of common terms of S_i and S_j
Max_{SSCS} : is maximum SSCS among all sentences in D

For example, sentences S_1 and S_7 given in Subsection 2.2.1 has 1 common term, blast. Therefore,

$$c_t(S_1, S_7) = 1 \tag{3.99}$$

SSCS score of sentence S_1 is calculated as:

$$SSCS(S_1) = 5 + 1 = 6 \tag{3.100}$$

As sentence S_2 and S_4 has maximum SSCS, 11. Therefore, the normalized SSCS of sentence S_1 is:

$$\begin{aligned} NSSCS(S_1) &= \frac{6}{11} \\ &= 0.5455 \end{aligned} \tag{3.101}$$

3.3.13 Sentence to Centroid Cohesion

The feature sentence to centroid cohesion considers sentences with higher degree of centroid cohesion as more relevant as proposed by Neto et al [84]. In this feature similarity between sentence and centroid of the document is computed. Centroid of the document is defined as a vector $Centroid = \{v_1, v_2, ..., v_n\}$. Here, v_j value for j^{th} term is calculated using Equation 3.102. After computing centroid of the document, score of i^{th} sentence S_i is calculated as given in Equation 3.103. After that normalized score of sentence S_i, $NSCCS(S_i)$ is calculated as given in Equation 3.104.

$$v_j = \frac{\text{Number of sentences in which term } t_j \text{ occurs}}{N} \tag{3.102}$$

$$SCCS(S_i) = \sum_{j=1}^{d_t(S_i)} min(tf(t_j, S_i), v_j) \tag{3.103}$$

$$NSCCS(S_i) = \frac{SCCS(S_i)}{Max_{SCCS}} \tag{3.104}$$

where,
$d_t(S_i)$: is the total number of distinct terms of sentence S_i
tf(t_j, S_i) : is the term frequency of term t_j in sentence S_i
Max_{SCCS} : is maximum SCCS among all sentences in D

For example, value for term morn is calculated as:

$$\begin{aligned} v_{morn} &= \frac{1}{8} \\ &= 0.125 \end{aligned} \tag{3.105}$$

Now, SCCS score of sentence S_2 given in Subsection 2.2.1 is calculated as:

$$\begin{aligned} SCCS(S_2) &= min(tf(\text{accord,}S_2), v_{accord}) + min(tf(\text{eyewit,}S_2), v_{eyewit}) \\ &\quad + min(tf(\text{bomb,}S_2), v_{bomb}) + min(tf(\text{place,}S_2), v_{place}) \\ &\quad + min(tf(\text{abandon,}S_2), v_{abandon}) + min(tf(\text{shop,}S_2), v_{shop}) \\ &= 0.125 + 0.125 + 0.125 + 0.125 + 0.125 + 0.125 \\ &= 0.75 \end{aligned} \tag{3.106}$$

As sentence S_5 has maximum SCCS, 1. Therefore, the normalized SCCS of sentence S_2 is:

$$\begin{aligned} NSCCS(S_2) &= \frac{0.75}{1} \\ &= 0.75 \end{aligned} \tag{3.107}$$

3.3.14 Depth of Sentence in the Tree

In the feature depth of sentence in the tree proposed by Neto et al [84], sentences of a document are represented in the form of a binary tree and then sentence score is calculated based on depth of the given sentence in the tree. To construct the binary tree of the sentences, agglomerative clustering is used. In agglomerative clustering, similar sentences are grouped together, in a bottom-up fashion producing a hierarchical binary tree called as the dendrogram. To identify similar sentences to group together, the similarity between two sentences S_i and S_j is defined as term overlap between them. Score DSTS(S_i) of i^{th} sentence S_i is calculated as given in Equation 3.108. After that normalized score of sentence S_i,

$NDSTS(S_i)$ is calculated as given in Equation 3.109.

$$DSTS(S_i) = \text{depth of sentence } S_i \text{ in dendogram} \tag{3.108}$$

$$NDSTS(S_i) = \frac{DSTS(S_i)}{Max_{DSTS}} \tag{3.109}$$

where:
Max_{DSTS} : is maximum DSTS among all sentences in D

For example, sentence S_2 and S_4 given in Subsection 2.2.1 has highest similarity score, therefore these sentences are merged first and has highest depth in the tree i.e. 6. Thus,

$$DSTS(S_2) = DSTS(S_4) = 6 \tag{3.110}$$

As sentence S_2 and S_4 has maximum DSTS, 6. Therefore, the normalized DSTS of sentence S_2 and sentence S_4 is:

$$\begin{aligned} NDSTS(S_2) &= NDSTS(S_4) && (3.111)\\ &= \frac{6}{6} \\ &= 1 && (3.112)\end{aligned}$$

3.4 Summary

In this chapter different statistical and graph features are discussed with examples. Features are categorized in three groups that are word level features, graph level features and sentence level features. Similar is the categorization of sentence scoring. These features are used in Chapter 4 for proposed analysis of feature combinations.

Chapter 4

Impact Analysis of Feature Combinations on Sentence Scoring

A typical extractive automatic text summarization process completes in three steps namely preprocessing, sentence scoring and summary generation. The sentence scoring step computes the score of each sentence present in a source text document. In order to score these sentences a number of statistical and lexical features are proposed by a number of researchers. These features used for sentence scoring are already described in Chapter 3. In this chapter performance of these features is analyzed individually. After that, features that perform well termed as best features using ROUGE evaluation measures are used for creating feature combinations. After that, best performing feature combinations are identified. A deep performance evaluation of best performing feature combinations on short, medium and large size documents is also conducted using same ROUGE performance measures.

4.1 Progress In Statistical Analysis of Features and Motivation

From early 1960s, researchers are continuously trying to analyze the feature from different perspectives in order to generate efficient text summaries. Term frequency [8] was the first feature proposed for sentence scoring. Later, sentence location, title similarity and cue word features were proposed by Edmundson [10] and boosted researchers to combine the features using different parameters. It was concluded that all sentence location, cue word and title similarity, using their linear combination performs better as compared to term frequency. Thereafter, researchers added features such as named entity, tf-isf, gain, textrank, lexrank, word co-occurrence etc. for sentence scoring. In 2013, Rafael et al [85] assessed a broader set of features that includes term frequency, tf-isf, upper case, proper noun, word co-occurrence, lexical similarity, cue word, numerical data, sentence length, sentence position, title similarity, aggregate similarity , bushy path and textrank. They tested each and every algorithm for three different datasets. It was concluded that term frequency, tf-isf, sentence length, lexical similarity and text rank are better features for sentence scoring as compared to the rest of the features listed. Later in 2014, Rafael et al [86] analyzed the performance of previously proposed algorithms using different combinations. The proposed methodology was termed as context based summarization and three groups of features were created namely word level, sentence level and graph level. All combinations of intra-group and inter-group word level, sentence level and graph level features are tried to test their impact on sentence scoring. This approach concluded that word frequency, title similarity and sentence location are the best features specifically for News domain. However, in all assessments listed above impact analysis of features such as sentence entropy, sentence to sentence cohesion etc. has not been carried out and a single evaluation measure ROUGE-1 for extracting best feature combinations is used. Moreover, some semantic features are also analyzed with statistical features. However, lack of abundant knowledge sources makes it difficult to use semantic features. This motivated to use exhaustive set of features for impact analysis other than semantic features (due to lack of knowledge sources) and evaluation measures(ROUGE-1, ROUGE-2, ROUGE-L and ROUGE-W).

4.2 Process Flow

As discussed earlier, extractive text summarization in general completes in three steps namely: pre-processing, sentence scoring and summary generation. Process flow to analyze the impact of feature combinations is shown in Figure 4.1. Pre-processing is applied on the text document using the process described in Subsection 2.2.1. Each sentence is then scored using all the features one by one. After sentence scoring, summary is generated using each feature. ROUGE-1 evaluation measure as described in Section 4.5 is then applied on the summary and

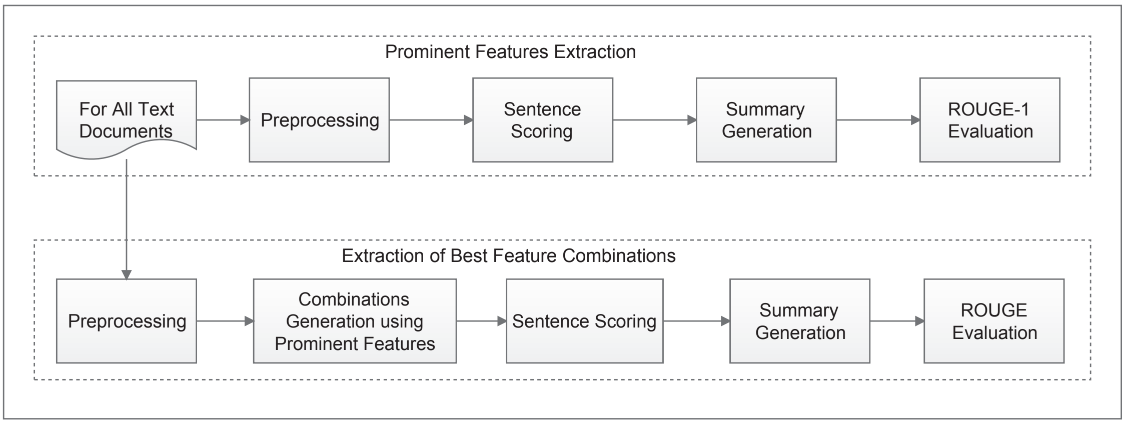


Figure 4.1: Summarization process flow for selection of best combinations.

relevant features are further used for generation of feature combinations. These relevant features are termed as prominent features. For each feature combination, a summary is generated. ROUGE evaluation measures (ROUGE-1, ROUGE-2, ROUGE-L, ROUGE-W) are further applied to get the better performing feature combinations which are termed as best feature combinations.

4.3 Feature Algorithms

The features used for proposed impact analysis of feature combinations are detailed in Chapter 3. These features are considered as feature algorithms for sentence scoring. Their level of processing and abbreviated feature algorithm names are shown in Table 4.1. For the feature algorithm sentence location two versions are used for sentence scoring, where the first version gives higher scores to sentence from the one end i.e. start of the document, in second version sentences are scored from both ends. Features performing well using ROUGE-1 are selected for feature combination generation.

4.4 Data Set Description

Document Understating Conference (DUC) 2002 dataset is used for experimental evaluation of proposed impact analysis. This dataset contains News data along with their gold summaries(manual summaries provided by humans). Sixty reference documents sets, each of them consisting of approximately ten documents for the evaluation task are provided in DUC 2002. For each document, DUC 2002

Table 4.1: All algorithms used for analysis.

S. No.	Short Name	Feature Algorithm Name	Level
1	FA1	Bushy Path	Graph Level
2	FA2	Cosine Similarity With Title	Sentence Level
3	FA3	Cue Words	Word Level
4	FA4	Depth of Sentence in the Tree	Sentence Level
5	FA5	Gain	Word Level
6	FA6	Hamming Distance	Sentence Level
7	FA7	Hamming Weight	Sentence Level
8	FA8	Interaction Between the Sentences	Sentence Level
9	FA9	LexRank	Graph Level
10	FA10	Named Entity	Word Level
11	FA11	Numerical Data	Sentence Level
12	FA12	Aggregate Similarity	Sentence Level
13	FA13	Sentence Entropy	Sentence Level
14	FA14	Sentence Location1	Sentence Level
15	FA15	Sentence Location2	Sentence Level
16	FA16	Sentence to Centroid Cohesion	Sentence Level
17	FA17	Sentence to Sentence Cohesion	Sentence Level
18	FA18	Sum of Probability	Sentence Level
19	FA19	Sum of Hamming Weights of Words by Frequency	Sentence Level
20	FA20	Term Frequency	Word Level
21	FA21	TextRank	Graph Level
22	FA22	TFIDF	Word Level
23	FA23	Title Similarity	Sentence Level
24	FA24	Total Terms in Sentence	Sentence Level
25	FA25	Word Co-occurrence	Sentence Level

provides two abstractive summaries as gold summaries with about hundred words each.

4.5 Evaluation Measures

Automatic evaluation of text summaries is a challenging and difficult task. the gold summaries provided by human experts and summary generated by automatic text summarization systems usually do not match exactly. Lin [14] proposed Recall-Oriented Understudy for Gisting Evaluation (ROUGE) measures for evaluation of text summarization systems. ROUGE measures are the only available commonly used measures for the purpose of automatic evaluation. ROUGE-1 and ROUGE-2 are versions of ROUGE-N, where N is 1 and 2, respectively. ROUGE-L and ROUGE-W are computed using longest common sub-sequences. The measures are described in detail in the following subsections.

4.5.1 ROUGE-N

Efficiency of information retrieval system is measured by precision, recall and f-measure. ROUGE-N recall, precision and f-measure are calculated as given in equation 4.1, 4.2 and 4.3 respectively. $Count_{match}(gram_n)$ computes the matching n-grams in between gold and system generated summaries.

$$Recall_n = \frac{\sum_{S \in GoldSummary} \sum_{gram_n \in S} count_{match}(gram_n)}{\sum_{S \in GoldSummary} \sum_{gram_n \in S} count(gram_n)} \tag{4.1}$$

$$Precision_n = \frac{\sum_{S \in SystemSummary} \sum_{gram_n \in S} count_{match}(gram_n)}{\sum_{S \in SystemSummary} \sum_{gram_n \in S} count(gram_n)} \tag{4.2}$$

$$F - measure_n = \frac{2 * Recall_n * Precision_n}{Recall_n + Precision_n} \tag{4.3}$$

ROUGE precision compares gold summary and system generated summary focusing on system generated summary. Recall on the other hand compares gold summary and system generated summary focusing on gold summary. F- Measure is harmonic mean of precision and recall. ROUGE-N checks N-gram co-occurrences of terms/phrases in between system generated summary and gold summary. ROUGE-L extracts longest common sub-sequence in sequence of n-grams. In ROUGE-W, weighted LCS is used that favors consecutive longest common sub-sequences.

4.5.2 ROUGE-L: Longest Common Sub-sequence

Longest common sub-sequence (LCS) of two sequences r and s is a common sub-sequence with maximum length. LCS is used as a string matching algorithm. To evaluate a summary using LCS summary sentences are viewed as a sequence of words. Sentences with longer LCS are considered as more similar to each other.

4.5.2.1 Sentence-Level LCS

LCS based measures for two summary sentences, reference summary sentence r of length m and system summary sentence s of length n is computed as given in Equation 4.5, 4.4 and 4.6.

$$Recall_{SLCS} = \frac{LCS(r, s)}{m} \tag{4.4}$$

$$Precision_{SLCS} = \frac{LCS(r,s)}{n} \tag{4.5}$$

$$F-measure_{SLCS} = \frac{2 * Recall_{LCS} * Precision_{LCS}}{Recall_{LCS} + Precision_{LCS}} \tag{4.6}$$

Where LCS(r,s) is the length of a LCS of r and s.

4.5.2.2 Summary-Level LCS

To compute summary-level LCS-based measures for two summaries, reference summary R and system summary S, union LCS matches between a reference summary sentence, r_i , and every system summary sentence, s_j is considered. The summary-level LCS-based measures for a reference summary, R of u sentences containing a total of m words and a system summary, S of v sentences containing a total of n words is computed as given in Equation 4.7, 4.8 and 4.9.

$$Recall_{LCS} = \frac{\sum_{i=1}^{u} LCS_{\cup}(r_i, S)}{m} \tag{4.7}$$

$$Precision_{LCS} = \frac{\sum_{i=1}^{v} LCS_{\cup}(r_i, R)}{n} \tag{4.8}$$

$$F-measure_{LCS} = \frac{2 * Recall_{LCS} * Precision_{LCS}}{Recall_{LCS} + Precision_{LCS}} \tag{4.9}$$

4.5.3 ROUGE-W: Weighted Longest Common Subsequence

There is a problem with LCS-based method that it does not differentiate LCSes of different spatial relations within their embedding sequences. To improve LCS-based method weighted LCS (WLCS) method is used. In WLCS, length of consecutive matches encountered so far is stored in a two dimensional dynamic program table computing LCS.

4.5.4 An Example: ROUGE Evaluation Measures

For the example text given in Subsection 2.2.1, let the gold summary, and peer or system genetratd summary are given in Table 4.2 and Table 4.4, respectively. Their stemmed version are also given in Table 4.3 and Table 4.5, respectively.

Table 4.2: Original gold summary.

S_1	At around 8 in the morning, around 10 people have lost their lives and 25 are injured in an incident of the bomb blast, happened in the middle of the market.
S_2	The government has ordered to pay Rs. 1 lakh in restitution to the next of kin of the dead.
S_3	The district magistrate ramesh kumar condemned the whole incident and announced to punish the terrorists behind the attack.

Table 4.3: Stemmed gold summary.

S_1	at around 8 in the morn , around 10 peopl have lost their live and 25 are injur in an incid of the bomb blast , happen in the middl of the market .
S_2	the govern has order to pay rs . 1 lakh in restitut to the next of kin of the dead .
S_3	the district magistr ramesh kumar condemn the whole incid and announc to punish the terrorist behind the attack .

Table 4.4: Original peer summary.

S_1	The government has also declared a compensation of 1 lakh to the next of kin of the dead.
S_2	The district magistrate Ramesh Kumar condemned the whole incident and said that those who have done this will be punished.
S_3	At around 8 in the morning, there was a blast in the middle of the market.
S_4	According to the eyewitnesses, the bomb was placed in an abandoned shop.
S_5	The government has formed an SIT to investigate the matter.

Table 4.5: Stemmed peer summary.

S_1	the govern has also declar a compens of 1 lakh to the next of kin of the dead.
S_2	the district magistr ramesh kumar condemn the whole incid and said that those who have done this will be punish.
S_3	at around 8 in the morn , there was a blast in the middl of the market.
S_4	accord to the eyewit , the bomb was place in an abandon shop.
S_5	the govern has form an sit to investig the matter.

To compute ROUGE-2 scores, 2-grams needs to be genereted from the gold and peer summaries. The genereted 2-grams for peer summary and gold summary are presented in Table 4.6 and Table 4.7, respectily. The common 2-grams of peer and gold summaries are shown in Table 4.8.

Table 4.6: Peer summary 2-grams.

Peer 2-Grams	Freq.	Peer 2-Grams	Freq.	Peer 2-Grams	Freq.	Peer 2-Grams	Freq.
middl of	1	the next	1	of the	2	said that	1
will be	1	punish at	1	be punish	1	at around	1
declar a	1	to investig	1	around 8	1	the dead	1
a compens	1	incid and	1	the eyewit	1	has also	1
those who	1	blast in	1	sit to	1	the govern	2
kumar condemn	1	the bomb	1	the morn	1	whole incid	1
of kin	1	eyewit the	1	and said	1	has form	1
the matter	1	condemn the	1	abandon shop	1	morn there	1
place in	1	district magistr	1	an sit	1	govern has	2
8 in	1	accord to	1	was a	1	in an	1
shop the	1	1 lakh	1	to the	2	lakh to	1
a blast	1	next of	1	there was	1	the market	1
was place	1	that those	1	compens of	1	the district	1
bomb was	1	have done	1	dead the	1	also declar	1
form an	1	this will	1	ramesh kumar	1	investig the	1
magistr ramesh	1	kin of	1	an abandon	1	the middl	1
of 1	1	in the	2	done this	1		
who have	1	market accord	1	the whole	1		
Total Number of Peer 2-Grams (NP2G) = 75							

Table 4.7: ROUGE-2 gold 2-grams.

Gold 2-Grams	Freq.	Gold 2-Grams	Freq.	Gold 2-Grams	Freq.	Gold 2-Grams	Freq.
middl of	1	their live	1	punish the	1	at around	1
bomb blast	1	rs 1	1	25 are	1	the dead	1
the attack	1	incid and	1	in the	2	terrorist behind	1
injur in	1	the bomb	1	of the	3	are injur	1
market the	1	blast happen	1	around 8	1	the govern	1
peopl have	1	the terrorist	1	behind the	1	whole incid	1
lost their	1	announc to	1	lakh in	1	10 peopl	1
kumar condemn	1	in restitut	1	live and	1	and 25	1
of kin	1	condemn the	1	the morn	1	incid of	1
an incid	1	has order	1	have lost	1	in an	1
8 in	1	district magistr	1	to the	1	govern has	1
pay rs	1	1 lakh	1	dead the	1	morn around	1
magistr ramesh	1	next of	1	happen in	1	the market	1
order to	1	to punish	1	ramesh kumar	1	the district	1
to pay	1	kin of	1	restitut to	1	and announc	1
the next	1	around 10	1	the whole	1	the middl	1
Total Number of Gold 2-Grams (NG2G) = 67							

Table 4.8: ROUGE-2 common 2-grams.

Common 2-Grams	Freq.	Common 2-Grams	Freq.	Common 2-Grams	Freq.	Common 2-Grams	Freq.
middl of	1	condemn the	1	the morn	1	whole incid	1
kumar condemn	1	district magistr	1	to the	1	in an	1
of kin	1	1 lakh	1	dead the	1	govern has	1
8 in	1	next of	1	ramesh kumar	1	the market	1
magistr ramesh	1	kin of	1	the whole	1	the district	1
the next	1	in the	2	at around	1	the middl	1
incid and	1	of the	2	the dead	1		
the bomb	1	around 8	1	the govern	1		
Total Number of Common 2-Grams (NC2G) = 32							

Now, ROUGE-2 recall, precision and f-measure values are calculated as follows:

$$\begin{aligned}\text{ROUGE-2 Recall, R} &= \frac{NC2G}{NG2G}\\ &= \frac{32}{67} = 0.4776 \end{aligned} \tag{4.10}$$

$$\begin{aligned}\text{ROUGE-2 Precision, P} &= \frac{NC2G}{NP2G}\\ &= \frac{32}{75} = 0.4267 \end{aligned} \tag{4.11}$$

$$\begin{aligned}\text{ROUGE-2 F-Measure} &= \frac{2 * P * R}{P + R}\\ &= \frac{2 * 0.4267 * 0.4776}{0.4267 + 0.4776}\\ &= 0.4507 \end{aligned} \tag{4.12}$$

Now for computing ROUGE-L score, LCS of gold summary sentences S_1, S_2, and S_3 with all peer summary sentences is shown in Table 4.9, Table 4.10, and Table 4.11, respectively.

Table 4.9: LCS of gold summary sentence, S_1 with all peer summary sentences.

S_1	the of the of the
S_2	the have
S_3	at around 8 in the morn blast in the middl of the market
S_4	the the bomb in
S_5	the an the
Union: the of have at around 8 in morn blast middl market bomb an	

Table 4.10: LCS of gold summary sentence, S_2 with all peer summary sentences.

S_1	the govern has 1 lakh to the next of kin of the dead
S_2	the the
S_3	in the the of the
S_4	to the the
S_5	the govern has to the
	Union: the govern has 1 lakh to next of kin dead in

Table 4.11: LCS of gold summary sentence, S_3 with all peer summary sentences.

S_1	the to the the
S_2	the district magistr ramesh kumar condemn the whole incid and punish
S_3	the the the
S_4	to the the
S_5	the to the
Union:	the to district magistr ramesh kumar condemn whole incid and punish

LCS length is calculated by adding length of each union i.e.

$$\text{LCS Length} = 13 + 11 + 11 = 35 \tag{4.13}$$

Now, ROUGE-L recall, precision and f-measure values are calculated as follows:

$$\begin{aligned}\text{ROUGE-L Recall, R} &= \frac{\text{LCS Length}}{\text{Size of Gold Summary}}\\ &= \frac{35}{68} = 0.5147 \end{aligned} \tag{4.14}$$

$$\begin{aligned}\text{ROUGE-L Precision, P} &= \frac{\text{LCS Length}}{\text{Size of Peer Summary}}\\ &= \frac{35}{76} = 0.4605 \end{aligned} \tag{4.15}$$

$$\begin{aligned}\text{ROUGE-L F-Measure} &= \frac{2 * P * R}{P + R} \\ &= \frac{2 * 0.4605 * 0.5147}{0.4605 + 0.5147} \\ &= 0.4861 \end{aligned} \tag{4.16}$$

4.6 Experimental Setup

For experiments the DUC 2002 data set is divided into three groups. Documents of size upto thirty sentences are grouped into short size group. This group contains three hundred forty four number of text documents. Documents which contain sentences in the range thirty one to sixty are clustered in medium size group. There are one hundred ninety nine documents available in this group. Documents having sentences more than sixty are grouped in third cluster i.e. large size group. Total seventeen documents are there in large size group. Documents of DUC 2002 are first converted into txt format, since they are available in xml format.

4.7 Results

After summary generation using a specific sentence scoring ROUGE-1, ROUGE-2, ROUGE-L and ROUGE-W scores are computed for each document. Initially, for each feature algorithm precision, recall and f-Measure value is calculated for each document. Thereafter, features that performs well are identified and termed as prominent features. These feature algorithms are then used for generating feature combinations. The feature algorithm combinations which performs well are termed as best feature combinations.

4.7.1 Selection of Best Features

Results of each feature algorithm using ROUGE-1 and ROUGE-2 on whole document set is presented in Table 4.12. ROUGE-L and ROUGE-W scores are shown in Table 4.13. In comparison with Rafeal et al [86] work, half of the prominent feature algorithms are new. These prominent feature algorithms are further used for feature combination generation. Sentence location1 [9] performs better with highest F-measure value of ROUGE-1 i.e. 0.47157 as compared to all other feature algorithms.

Table 4.12: All feature algorithms ROUGE-1 and ROUGE-2 scores for whole document set.

S. No.	Method	ROUGE-1			ROUGE-2		
		Precision	Recall	F-Measure	Precision	Recall	F-Measure
1	FA1	0.44975	0.47630	0.46078	0.18980	0.20144	0.19464
2	FA2	0.41988	0.44352	0.42965	0.16486	0.17461	0.16882
3	FA3	0.42419	0.45208	0.43609	0.16547	0.17696	0.17036
4	FA4	0.45437	0.47772	0.46396	0.19843	0.20891	0.20272
5	FA5	0.42805	0.44613	0.43498	0.17383	0.18179	0.17688
6	FA6	0.44850	0.47192	0.45801	0.19165	0.20239	0.19602
7	FA7	0.43375	0.45286	0.44115	0.18021	0.18878	0.18354
8	FA8	0.37643	0.40510	0.38891	0.12121	0.13118	0.12551
9	FA9	0.44580	0.47185	0.45670	0.18421	0.19552	0.18896
10	FA10	0.42704	0.45043	0.43655	0.16670	0.17650	0.17065
11	FA11	0.44081	0.47110	0.45390	0.18867	0.20240	0.19459
12	FA12	0.44692	0.47294	0.45765	0.18970	0.20133	0.19448
13	FA13	0.39950	0.43170	0.41350	0.13823	0.15039	0.14344
14	FA14	0.46362	0.48374	0.47157	0.20976	0.21976	0.21372
15	FA15	0.45478	0.47825	0.46455	0.20114	0.21205	0.20567
16	FA16	0.44865	0.47313	0.45875	0.19213	0.20320	0.19670
17	FA17	0.45433	0.47867	0.46436	0.19719	0.20867	0.20193
18	FA18	0.44223	0.46745	0.45263	0.18754	0.19866	0.19211
19	FA19	0.43327	0.45254	0.44073	0.17921	0.18796	0.18261
20	FA20	0.44808	0.47331	0.45851	0.19188	0.20330	0.19659
21	FA21	0.41771	0.44144	0.42751	0.15706	0.16689	0.16114
22	FA22	0.42145	0.43975	0.42854	0.16864	0.17662	0.17176
23	FA23	0.45761	0.47895	0.46613	0.20180	0.21192	0.20584
24	FA24	0.43495	0.45657	0.44344	0.18099	0.19067	0.18481
25	FA25	0.39223	0.41844	0.40334	0.13803	0.14785	0.14214

Table 4.13: All feature algorithms ROUGE-L and ROUGE-W scores for whole document set.

S. No.	Method	ROUGE-L			ROUGE-W		
		Precision	Recall	F-Measure	Precision	Recall	F-Measure
1	FA1	0.37494	0.39704	0.38567	0.37192	0.39379	0.38254
2	FA2	0.34796	0.36729	0.35737	0.34400	0.36305	0.35327
3	FA3	0.35065	0.37361	0.36176	0.34712	0.36973	0.35807
4	FA4	0.37890	0.39818	0.38830	0.37519	0.39420	0.38446
5	FA5	0.35725	0.37217	0.36456	0.35430	0.36901	0.36151
6	FA6	0.37581	0.39542	0.38536	0.37238	0.39174	0.38181
7	FA7	0.36225	0.37802	0.36997	0.35937	0.37488	0.36696
8	FA8	0.30418	0.32752	0.31542	0.30157	0.32467	0.31269
9	FA9	0.37021	0.39189	0.38074	0.36661	0.38804	0.37702
10	FA10	0.35156	0.37081	0.36093	0.34825	0.36720	0.35747
11	FA11	0.36706	0.39222	0.37922	0.36350	0.38837	0.37552
12	FA12	0.37404	0.39586	0.38464	0.37100	0.39251	0.38145
13	FA13	0.32333	0.34952	0.33592	0.32013	0.34593	0.33253
14	FA14	0.37990	0.39930	0.38936	0.37589	0.39509	0.38525
15	FA15	0.38860	0.40551	0.39687	0.38494	0.40163	0.39311
16	FA16	0.37637	0.39696	0.38639	0.37303	0.39337	0.38293
17	FA17	0.38072	0.40108	0.39064	0.37728	0.39737	0.38706
18	FA18	0.37076	0.39174	0.38096	0.36749	0.38822	0.37757
19	FA19	0.36144	0.37745	0.36927	0.35844	0.37421	0.36615
20	FA20	0.37579	0.39711	0.38616	0.37234	0.39343	0.38260
21	FA21	0.34599	0.36561	0.35553	0.34278	0.36210	0.35218
22	FA22	0.35114	0.36616	0.35849	0.34812	0.36289	0.35535
23	FA23	0.38272	0.40055	0.39143	0.37922	0.39682	0.38782
24	FA24	0.36252	0.38045	0.37127	0.35933	0.37701	0.36795
25	FA25	0.31938	0.34066	0.32968	0.31571	0.33663	0.32584

4.7.2 Best Feature Algorithm Combinations

Using prominent feature algorithms, different feature algorithm combinations are generated for sentence scoring. Initially, Best forty combinations for each evaluation measure ROUGE-1, ROUGE-2, ROUGE-L and ROUGE-W are selected. Combinations which are present in the lists of best performing combinations, for each of the four evaluation measures are selected as final best feature algorithm combinations. These combinations are listed in Table 4.14. The list comprised of features algorithms busy path, cosine similarity with title, depth of the sentence in the tree, lexrank, aggregate similarity, sentence location1, sentence to sentence cohesion and title similarity. Despite having better performance individually, sentence entropy did not produce higher ROUGE scores in combinations with other feature algorithms. All best combinations are renamed as COMB with number in the same order of their respective rank for ROUGE-2 F-measure, for the purpose of comparison in graphical form. For example: first best performing combinations is renamed as COMB1.

4.7.3 ROUGE scores for Short Size Document Group

Results of best feature algorithm combinations using ROUGE-1 and ROUGE-2 on short size document group are presented in Table 4.15. Short size document group consist of sentences, whose size is less then or equal to thirty. F-measure value is almost in the similar range for both ROUGE-1. COMB1 f-measure value for ROUGE-1 is 0.50099. ROUGE-2 F-measure score of COMB1 is 0.24038. ROUGE-L and ROUGE-W scores for the same size group of documents are shown in Table 4.16. F-measure value of COMB1 using ROUGE-L is reported as 0.42161. Its ROUGE-W score is reported as 0.41729.

4.7.4 ROUGE scores for Medium Size Document Group

Results of best feature algorithm combinations using ROUGE-1 and ROUGE-2 on short size document group are presented in Table 4.17. Medium size document group consist of text documents having number of sentences in the range thirty one to sixty. COMB1 f-measure value for ROUGE-1 is highest reported as 0.44161. ROUGE-2 F-measure score of COMB15 is highest reported as 0.18076. ROUGE-L and ROUGE-W scores for the same size group of documents are shown in Table 4.18. F-measure value of COMB3 is reported as 0.35709 using ROUGE-L which

Table 4.14: Best feature algorithm combinations details.

S. No.	Combination ID	Features in Combination
1	COMB1	Aggregate Similarity + LexRank + Sentence Location1
2	COMB2	LexRank + Sentence Location1
3	COMB3	Aggregate Similarity + LexRank + Sentence Location1 + Sentence to Sentence Cohesion
4	COMB4	Bushy Path + LexRank + Sentence Location1 + Sentence to Sentence Cohesion
5	COMB5	LexRank + Sentence Location1 + Sentence to Sentence Cohesion + Sentence to Centroid Cohesion
6	COMB6	LexRank + Sentence Location1 + Sentence to Sentence Cohesion
7	COMB7	Depth of the Sentence in the Tree + LexRank + Sentence Location1
8	COMB8	Aggregate Similarity + LexRank + Sentence Location1 + Sentence to Sentence Cohesion + Sentence to Centroid Cohesion
9	COMB9	Depth of the Sentence in the Tree + LexRank + Sentence Location1 + Sentence to Centroid Cohesion
10	COMB10	LexRank + Sentence Location1 + Sentence to Sentence Cohesion + Term Frequency
11	COMB11	Aggregate Similarity + Depth of the Sentence in the Tree + LexRank + Sentence Location1 + Sentence to Sentence Cohesion
12	COMB12	Depth of the Sentence in the Tree + LexRank + Sentence Location1 + Term Frequency
13	COMB13	Aggregate Similarity + Depth of the Sentence in the Tree + LexRank + Sentence Location1
14	COMB14	Aggregate Similarity + LexRank + Sentence Location1 + Sentence to Sentence Cohesion + Term Frequency
15	COMB15	Sentence Location1 + Sentence to Sentence Cohesion
16	COMB16	Aggregate Similarity + Depth of the Sentence in the Tree + LexRank + Sentence Location1 + Sentence to Centroid Cohesion
17	COMB17	Aggregate Similarity + Frequential Sum of Probability + LexRank + Sentence Location1 + Sentence to Sentence Cohesion
18	COMB18	Aggregate Similarity + Sentence Location1 + Sentence to Sentence Cohesion
19	COMB19	Aggregate Similarity + Depth of the Sentence in the Tree + LexRank + Sentence Location1 + Term Frequency
20	COMB20	Depth of the Sentence in the Tree + Sentence Location1 +Term Frequency
21	COMB21	Aggregate Similarity + Sentence Location1
22	COMB22	Aggregate Similarity + Depth of the Sentence in the Tree + Hamming distance + LexRank + Sentence Location1

is highest among all others. Highest ROUGE-W f-measure score is reported as 0.35353 for COMB3.

4.7.5 ROUGE scores for Large Size Document Group

Results of best feature algorithm combinations using ROUGE-1 and ROUGE-2 on large size document group are presented in Table 4.19.

ROUGE-1 and ROUGE-2 score of COMB2 is reported as highest as compared to all other combinations for medium size group of documents. Results of best

Table 4.15: ROUGE-1 and ROUGE-2 scores of best combinations for short size group.

S. No.	Method	ROUGE-1			ROUGE-2		
		Precision	Recall	F-Measure	Precision	Recall	F-Measure
1	COMB1	0.48887	0.51845	0.50099	0.23445	0.24904	0.24038
2	COMB2	0.48687	0.51668	0.49921	0.23275	0.2479	0.23898
3	COMB3	0.48694	0.51381	0.49787	0.23209	0.24597	0.23772
4	COMB4	0.48705	0.51408	0.49796	0.23082	0.24438	0.23627
5	COMB5	0.48632	0.51578	0.49832	0.23205	0.24643	0.23784
6	COMB6	0.4867	0.51183	0.49682	0.23206	0.24493	0.23723
7	COMB7	0.48654	0.51592	0.49871	0.23361	0.248	0.23952
8	COMB8	0.48657	0.51612	0.49866	0.23245	0.24707	0.23837
9	COMB9	0.48586	0.51676	0.49869	0.23308	0.24837	0.23939
10	COMB10	0.48597	0.51588	0.49823	0.23148	0.24621	0.23745
11	COMB11	0.48472	0.51394	0.4967	0.23106	0.24604	0.2372
12	COMB12	0.48566	0.51607	0.49831	0.23262	0.24737	0.2387
13	COMB13	0.4865	0.51502	0.49806	0.23293	0.24702	0.2386
14	COMB14	0.48666	0.51576	0.49853	0.2322	0.24656	0.238
15	COMB15	0.48466	0.5125	0.49607	0.23156	0.24612	0.23755
16	COMB16	0.48601	0.51672	0.4986	0.23331	0.24848	0.23948
17	COMB17	0.48518	0.51633	0.49795	0.23172	0.24696	0.2379
18	COMB18	0.48438	0.51272	0.49606	0.23093	0.24559	0.23698
19	COMB19	0.48521	0.51661	0.4983	0.23265	0.24789	0.23894
20	COMB20	0.48411	0.51528	0.497	0.23138	0.24674	0.23769
21	COMB21	0.48555	0.51438	0.49731	0.23375	0.24861	0.23978
22	COMB22	0.4851	0.51506	0.49748	0.23124	0.24565	0.23716

Table 4.16: ROUGE-L and ROUGE-W scores of best combinations for short size group.

S. No.	Method	ROUGE-L			ROUGE-W		
		Precision	Recall	F-Measure	Precision	Recall	F-Measure
1	COMB1	0.41136	0.43641	0.42161	0.40718	0.43189	0.41729
2	COMB2	0.40916	0.43444	0.4196	0.40507	0.43	0.41537
3	COMB3	0.40946	0.43233	0.41877	0.40613	0.42872	0.41532
4	COMB4	0.40905	0.43213	0.41838	0.40587	0.42858	0.41504
5	COMB5	0.40967	0.43447	0.41977	0.40609	0.43054	0.41604
6	COMB6	0.40904	0.43025	0.41758	0.40542	0.42631	0.41383
7	COMB7	0.40874	0.43318	0.41885	0.40529	0.42951	0.41531
8	COMB8	0.41059	0.43561	0.42081	0.40678	0.43143	0.41685
9	COMB9	0.41038	0.43649	0.42121	0.40664	0.43246	0.41735
10	COMB10	0.40964	0.4349	0.41998	0.4061	0.431	0.41629
11	COMB11	0.40707	0.43163	0.41714	0.40346	0.42774	0.41341
12	COMB12	0.41026	0.43599	0.42095	0.40651	0.43197	0.4171
13	COMB13	0.40907	0.43294	0.41874	0.40573	0.42931	0.41528
14	COMB14	0.41055	0.43518	0.42058	0.40684	0.43109	0.41672
15	COMB15	0.40682	0.4305	0.41655	0.40328	0.42671	0.41291
16	COMB16	0.41073	0.4367	0.42137	0.40704	0.43272	0.41756
17	COMB17	0.40975	0.43608	0.42054	0.40586	0.43175	0.41646
18	COMB18	0.40682	0.43077	0.41671	0.40305	0.42674	0.41284
19	COMB19	0.4103	0.43689	0.42138	0.40666	0.43296	0.41762
20	COMB20	0.40807	0.43445	0.41897	0.40422	0.43036	0.41503
21	COMB21	0.40897	0.43351	0.41899	0.40506	0.42927	0.41495
22	COMB22	0.40853	0.43381	0.41898	0.40454	0.42948	0.41485

Table 4.17: ROUGE-1 and ROUGE-2 scores of best combinations for medium size group.

S. No.	**Method**	**ROUGE-1**			**ROUGE-2**		
		Precision	Recall	F-Measure	Precision	Recall	F-Measure
1	COMB1	0.43063	0.45595	0.44161	0.17596	0.18636	0.18048
2	COMB2	0.43001	0.44804	0.43774	0.1749	0.18207	0.17798
3	COMB3	0.43223	0.45149	0.44034	0.17722	0.18512	0.18056
4	COMB4	0.4316	0.45055	0.43967	0.1723	0.18022	0.17569
5	COMB5	0.42834	0.45125	0.4382	0.17337	0.18247	0.17732
6	COMB6	0.43166	0.45334	0.44094	0.17605	0.18526	0.18002
7	COMB7	0.43071	0.44575	0.43679	0.17544	0.1815	0.17788
8	COMB8	0.42635	0.45087	0.43691	0.17144	0.1812	0.17566
9	COMB9	0.42857	0.44635	0.43597	0.17317	0.18006	0.17603
10	COMB10	0.42724	0.4498	0.43693	0.17208	0.18113	0.176
11	COMB11	0.43099	0.44997	0.43907	0.17644	0.18425	0.17977
12	COMB12	0.42819	0.44578	0.43551	0.1734	0.18044	0.17634
13	COMB13	0.42895	0.44588	0.43604	0.17316	0.18006	0.17605
14	COMB14	0.42564	0.44878	0.43553	0.17108	0.1806	0.17519
15	COMB15	0.43158	0.45089	0.4398	0.17712	0.18555	0.18076
16	COMB16	0.42677	0.44381	0.43379	0.17237	0.17938	0.17527
17	COMB17	0.42697	0.44782	0.43563	0.17272	0.18134	0.17634
18	COMB18	0.42972	0.45048	0.43865	0.17563	0.18462	0.17954
19	COMB19	0.42658	0.44323	0.43348	0.17209	0.17888	0.17492
20	COMB20	0.42798	0.44556	0.43527	0.17266	0.1798	0.17563
21	COMB21	0.42972	0.44731	0.43695	0.17666	0.18397	0.17971
22	COMB22	0.42806	0.44455	0.43477	0.17282	0.17863	0.17514

Table 4.18: ROUGE-L and ROUGE-W scores of best combinations for medium size group.

S. No.	**Method**	**ROUGE-L**			**ROUGE-W**		
		Precision	Recall	F-Measure	Precision	Recall	F-Measure
1	COMB1	0.3599	0.3807	0.36892	0.35672	0.37711	0.36556
2	COMB2	0.35976	0.37455	0.36611	0.35654	0.37103	0.36276
3	COMB3	0.3637	0.37941	0.3703	0.36003	0.37548	0.36652
4	COMB4	0.36026	0.37591	0.36693	0.35662	0.37198	0.36316
5	COMB5	0.35885	0.37766	0.36695	0.35561	0.37417	0.3636
6	COMB6	0.36285	0.38065	0.37046	0.35882	0.37635	0.36632
7	COMB7	0.36086	0.37325	0.36586	0.35724	0.36937	0.36213
8	COMB8	0.35689	0.3771	0.3656	0.35378	0.37371	0.36236
9	COMB9	0.35927	0.37356	0.36518	0.3559	0.3699	0.36168
10	COMB10	0.35742	0.37601	0.36541	0.35461	0.37299	0.36252
11	COMB11	0.36183	0.37753	0.36851	0.35857	0.37406	0.36516
12	COMB12	0.3585	0.37292	0.36449	0.35522	0.36933	0.36107
13	COMB13	0.35912	0.37292	0.36489	0.35634	0.36985	0.36199
14	COMB14	0.3565	0.37573	0.36474	0.35332	0.37229	0.36145
15	COMB15	0.36375	0.37993	0.37065	0.35989	0.37581	0.36668
16	COMB16	0.3579	0.37176	0.36359	0.35447	0.36809	0.36005
17	COMB17	0.35797	0.37518	0.36512	0.35486	0.37177	0.36189
18	COMB18	0.36223	0.37969	0.36975	0.35855	0.37577	0.36597
19	COMB19	0.35787	0.37145	0.36348	0.35454	0.36782	0.36002
20	COMB20	0.35968	0.37398	0.36559	0.35619	0.37022	0.36198
21	COMB21	0.35988	0.37445	0.36587	0.35609	0.37041	0.36198
22	COMB22	0.35851	0.37175	0.36386	0.35535	0.36825	0.36055

Table 4.19: ROUGE-1 and ROUGE-2 scores of best combinations for large size group.

S. No.	Method	ROUGE-1			ROUGE-2		
		Precision	Recall	F-Measure	Precision	Recall	F-Measure
1	COMB1	0.41583	0.44027	0.4266	0.13374	0.14141	0.13709
2	COMB2	0.41304	0.44628	0.42787	0.13479	0.14675	0.14016
3	COMB3	0.41315	0.42512	0.41767	0.13243	0.1358	0.13363
4	COMB4	0.41657	0.41475	0.4149	0.12636	0.12566	0.12575
5	COMB5	0.40819	0.4243	0.41505	0.1263	0.1327	0.12908
6	COMB6	0.41315	0.42512	0.41767	0.13243	0.1358	0.13363
7	COMB7	0.41039	0.42864	0.4185	0.13355	0.13967	0.13628
8	COMB8	0.40819	0.4243	0.41505	0.1263	0.1327	0.12908
9	COMB9	0.41926	0.43121	0.42386	0.13416	0.13885	0.13601
10	COMB10	0.4086	0.42499	0.41558	0.12653	0.13304	0.12936
11	COMB11	0.41236	0.43859	0.42337	0.13621	0.14595	0.14033
12	COMB12	0.41863	0.43184	0.42407	0.13412	0.13813	0.13577
13	COMB13	0.41747	0.42967	0.42264	0.13755	0.14138	0.13917
14	COMB14	0.4086	0.42499	0.41558	0.12653	0.13304	0.12936
15	COMB15	0.4162	0.42647	0.42	0.13448	0.13748	0.13554
16	COMB16	0.42559	0.42733	0.42571	0.13698	0.13778	0.13713
17	COMB17	0.40389	0.4229	0.41195	0.12244	0.12794	0.12476
18	COMB18	0.41315	0.42512	0.41767	0.13181	0.13511	0.13298
19	COMB19	0.42434	0.43149	0.42685	0.13578	0.13813	0.13662
20	COMB20	0.42443	0.43356	0.42794	0.13962	0.14292	0.1409
21	COMB21	0.3886	0.40076	0.39357	0.11031	0.11455	0.11207
22	COMB22	0.40546	0.4255	0.41394	0.12508	0.1313	0.12772

feature algorithm combinations using ROUGE-L and ROUGE-W on the same set are reported in Table 4.20. It is observed that COMB16 performs better as compared to all other combination for both ROUGE-L and ROUGE-W.

4.7.6 ROUGE scores for All Documents

Results of best feature algorithm combinations using ROUGE-1 and ROUGE-2 on all documents are presented in Table 4.21. This group contains five hundred sixty three text documents. Results of best feature algorithm combinations using ROUGE-1 and ROUGE-2 on same document group are reported in Table 4.22. ROUGE-1, ROUGE-2 , ROUGE-L and ROUGE-W f-measure scores of COMB1 are higher as compared to all other feature algorithm combinations and reported as 0.47973, 0.21823, 0.38285 and 0.37929.

Table 4.20: ROUGE-L and ROUGE-W scores of best combinations for large size group.

S. No.	Method	ROUGE-L			ROUGE-W		
		Precision	Recall	F-Measure	Precision	Recall	F-Measure
1	COMB1	0.33522	0.3542	0.34353	0.3372	0.35633	0.34559
2	COMB2	0.33269	0.36022	0.34498	0.33367	0.36164	0.34617
3	COMB3	0.33656	0.34585	0.34001	0.33427	0.34344	0.33768
4	COMB4	0.33683	0.33483	0.33521	0.33649	0.33448	0.33486
5	COMB5	0.33235	0.34617	0.33826	0.33274	0.34654	0.33864
6	COMB6	0.33665	0.34586	0.34007	0.33407	0.3431	0.33741
7	COMB7	0.33522	0.35059	0.34206	0.33665	0.35198	0.34347
8	COMB8	0.33211	0.34585	0.33798	0.33283	0.34657	0.3387
9	COMB9	0.34237	0.35198	0.34604	0.34382	0.35335	0.34745
10	COMB10	0.33254	0.34648	0.3385	0.33205	0.34581	0.33792
11	COMB11	0.33721	0.35902	0.34639	0.33454	0.35594	0.34353
12	COMB12	0.34277	0.35369	0.34726	0.34389	0.35471	0.34833
13	COMB13	0.34197	0.35189	0.34618	0.34435	0.35429	0.34856
14	COMB14	0.33166	0.34549	0.33757	0.33178	0.3455	0.33764
15	COMB15	0.34292	0.35108	0.34591	0.34193	0.35006	0.34491
16	COMB16	0.34875	0.34989	0.3487	0.35004	0.35091	0.34986
17	COMB17	0.32543	0.34058	0.33185	0.32296	0.33783	0.32924
18	COMB18	0.34101	0.35075	0.34467	0.33899	0.34869	0.34264
19	COMB19	0.34646	0.3523	0.34849	0.34663	0.35228	0.34857
20	COMB20	0.35017	0.35781	0.35311	0.34981	0.35711	0.35259
21	COMB21	0.31414	0.32366	0.31799	0.31625	0.32576	0.3201
22	COMB22	0.33218	0.34801	0.33886	0.33099	0.34696	0.33774

Table 4.21: ROUGE-1 and ROUGE-2 scores of best combinations for all documents.

S. No.	Method	ROUGE-1			ROUGE-2		
		Precision	Recall	F-Measure	Precision	Recall	F-Measure
1	COMB1	0.46802	0.49608	0.47973	0.21283	0.22588	0.21823
2	COMB2	0.46645	0.4925	0.47735	0.21141	0.22387	0.21658
3	COMB3	0.46722	0.49124	0.47707	0.21168	0.22334	0.21645
4	COMB4	0.46718	0.49085	0.47684	0.2091	0.22046	0.21373
5	COMB5	0.46543	0.49241	0.4766	0.21024	0.2227	0.21536
6	COMB6	0.46688	0.49056	0.47659	0.21128	0.22271	0.21595
7	COMB7	0.4664	0.49082	0.47648	0.21211	0.22357	0.2168
8	COMB8	0.46494	0.49251	0.4764	0.20987	0.2227	0.21516
9	COMB9	0.46548	0.49162	0.47633	0.21104	0.22332	0.21611
10	COMB10	0.46485	0.49202	0.47614	0.20946	0.22213	0.21468
11	COMB11	0.46536	0.49116	0.47604	0.21085	0.22336	0.21603
12	COMB12	0.46521	0.49101	0.47594	0.21081	0.22278	0.21576
13	COMB13	0.46597	0.4903	0.47592	0.21102	0.22251	0.21568
14	COMB14	0.46478	0.49162	0.47588	0.2096	0.22218	0.21477
15	COMB15	0.46561	0.49023	0.47579	0.21135	0.22362	0.21645
16	COMB16	0.46515	0.49067	0.47561	0.211	0.22315	0.21595
17	COMB17	0.46414	0.49161	0.47544	0.20972	0.22256	0.21497
18	COMB18	0.46475	0.4902	0.47535	0.21039	0.22292	0.21562
19	COMB19	0.46453	0.49051	0.47534	0.21045	0.22261	0.21548
20	COMB20	0.46428	0.49046	0.47511	0.20991	0.22228	0.215
21	COMB21	0.46489	0.48963	0.475	0.21202	0.22415	0.21697
22	COMB22	0.46448	0.48979	0.47491	0.20951	0.2209	0.21417

Table 4.22: ROUGE-L and ROUGE-W scores of best combinations for all documents.

S. No.	Method	ROUGE-L			ROUGE-W		
		Precision	Recall	F-Measure	Precision	Recall	F-Measure
1	COMB1	0.39265	0.41616	0.40245	0.38894	0.41211	0.3986
2	COMB2	0.39111	0.41302	0.40026	0.38743	0.40903	0.39645
3	COMB3	0.39269	0.41288	0.40097	0.38928	0.4092	0.39744
4	COMB4	0.39131	0.41133	0.39949	0.38805	0.40774	0.39609
5	COMB5	0.39114	0.41371	0.40048	0.38777	0.41003	0.39698
6	COMB6	0.39215	0.41194	0.40025	0.38842	0.40791	0.3964
7	COMB7	0.39126	0.41154	0.39963	0.38788	0.40793	0.39614
8	COMB8	0.39109	0.41425	0.40071	0.38762	0.41046	0.3971
9	COMB9	0.39198	0.41382	0.40103	0.38849	0.41005	0.39742
10	COMB10	0.39065	0.41346	0.40012	0.38743	0.40992	0.39677
11	COMB11	0.39055	0.41214	0.39949	0.38707	0.40841	0.39591
12	COMB12	0.39166	0.41333	0.40067	0.38819	0.40958	0.39708
13	COMB13	0.39108	0.4113	0.39933	0.38806	0.40801	0.3962
14	COMB14	0.39092	0.41352	0.40027	0.38748	0.40975	0.39669
15	COMB15	0.39115	0.412	0.39979	0.38757	0.40817	0.39611
16	COMB16	0.39192	0.41332	0.40068	0.38844	0.40956	0.39708
17	COMB17	0.39073	0.4138	0.40022	0.38713	0.40981	0.39646
18	COMB18	0.39061	0.41208	0.39957	0.38692	0.40814	0.39577
19	COMB19	0.39158	0.4134	0.40064	0.38813	0.40966	0.39707
20	COMB20	0.39081	0.41278	0.39989	0.38717	0.40888	0.39614
21	COMB21	0.39057	0.41148	0.39912	0.38685	0.40746	0.39528
22	COMB22	0.39029	0.4114	0.39898	0.38664	0.40741	0.39518

4.8 Analysis

Using the results obtained in the previous section, observations made on pre-processing, feature algorithm combinations and different size document groups are discussed in the following section.

4.8.1 Effect of Stemming

Pre-processing is an important step of any text processing system. Final results are very much sensitive to this pre-processing. Comparison of results obtained using ROUGE-1 evaluation measure on all documents for single feature algorithms, with stemming and without stemming is shown in Figure 4.2.

It is observed that results using stemming are better as compared to results without stemming. The fact that stemming use the root forms of the words, enhances the performance. Comparing terms using their stemmed version gives more number of matches, that ultimately improves the results. Comparison of results obtained using ROUGE-2 evaluation measure on all documents for single feature algorithms,

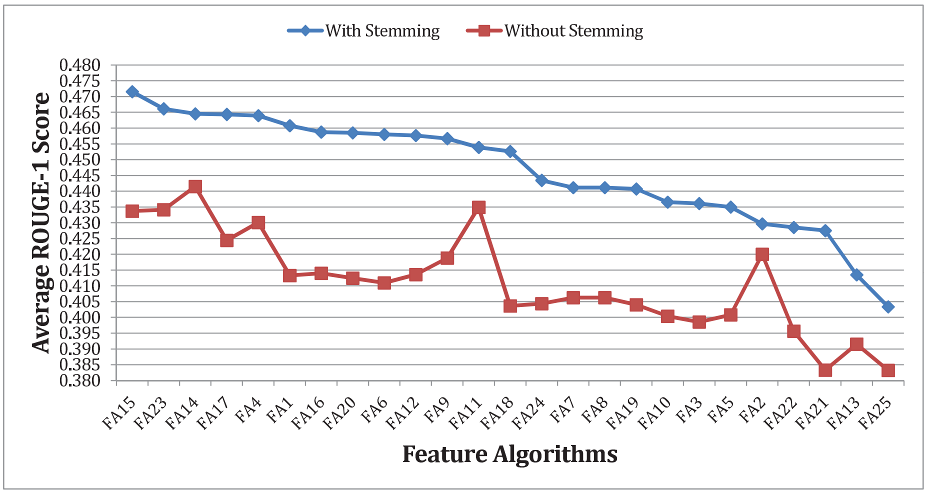


Figure 4.2: Comparisons of results obtained for single feature algorithms with and without stemming using ROUGE-1.

with stemming and without stemming is shown in Figure 4.3. Here also, the performance of stemmed version is better as compared without non-stemmed version.

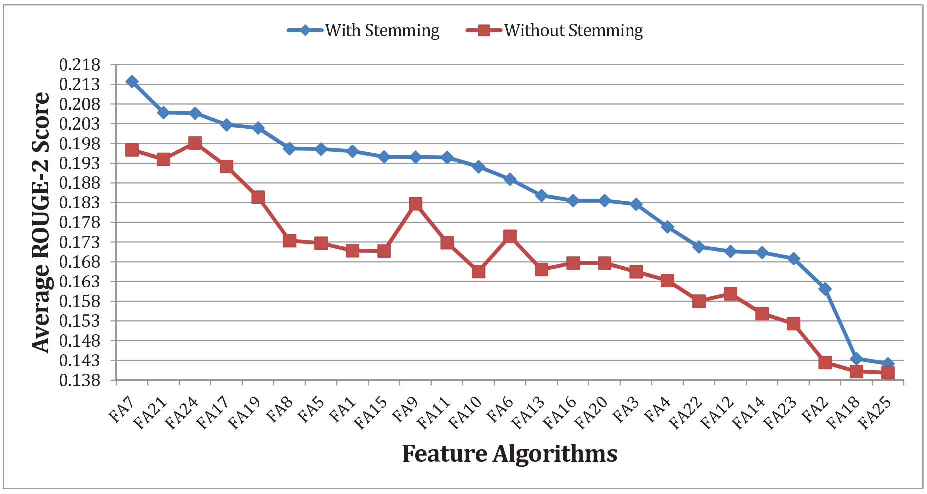


Figure 4.3: Comparisons of results obtained for single feature algorithms with and without stemming using ROUGE-2.

4.8.2 Feature algorithm combinations Effect on All Documents

Precision, Recall and F-Measure comparisons of best feature algorithm combinations for all document set using ROUGE-1 and ROUGE-2 are presented in Figure 4.4 and 4.5, respectively. Results are sorted on the basis of f-measure. It is observed that their are notable variations in ROUGE scores of precision and recall for some of the combinations. This is primarily due to the size of summary to be produced i.e. compression rate. In proposed experiments, the compression rate is

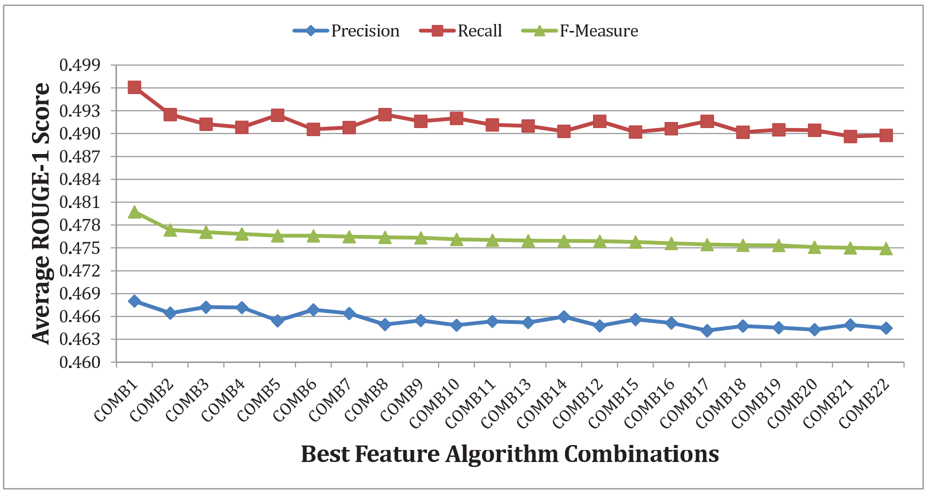


Figure 4.4: Variation in precision, recall and f-measure for whole document set using ROUGE-1.

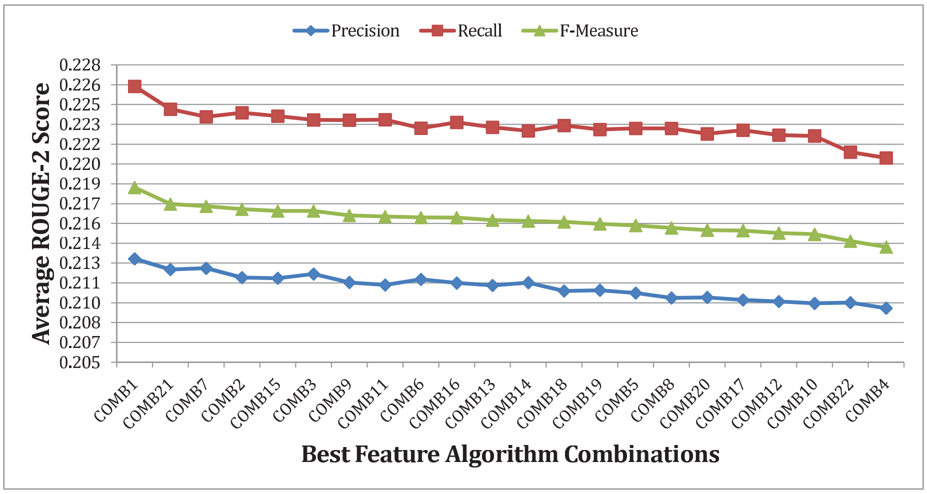


Figure 4.5: Variation in precision, recall and f-measure for whole document set using ROUGE-2.

hundred words. At the time, total words in the summary exceeds the limit beyond hundred words, the variation in between precision and recall is visible. More the limit exceeds, more the difference is reported in precision and recall. The smooth curve shows that COMB1 performs better as compared to all other feature algorithm combinations. ROUGE-2, ROUGE-L and ROUGE-W comparisons for the similar group are shown in Figure4.5, 4.6 and 4.7, respectively.

Similar to ROUGE-1 and ROUGE-2, it is observed that COMB1 performs better in case of ROUGE-L and ROUGE-W evaluation as well. COMB1 comprised of

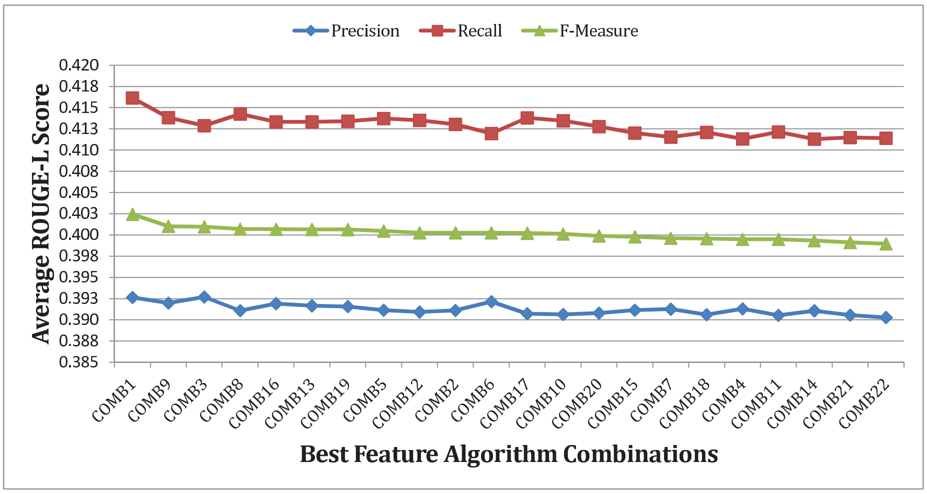


Figure 4.6: Variation in precision, recall and f-measure for whole document set using ROUGE-L.

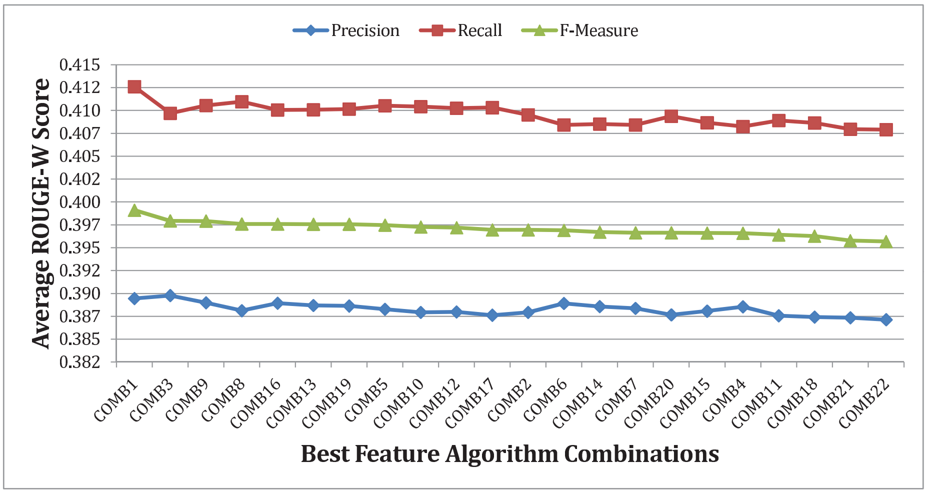


Figure 4.7: Variation in precision, recall and f-measure for whole document set using ROUGE-W.

feature algorithms aggregate similarity [24], lexrank [25] and sentence location1. This is primarily due to using graph based feature algorithms such as aggregate similarity [24] and lexrank [25] which focuses on the sentences which have more number of interlinks. In rest of the other combinations, feature algorithms such as depth of the sentence in the Tree [84], frequential sum of probability [83], hamming distance [83], sentence to sentence cohesion [84], term frequency [8] and title similarity [10] are present other then the three features present in COMB1. All the best twenty two feature algorithm combinations differs slightly in ROUGE scores. Comparing with individual feature algorithms, these feature algorithm combinations performs better as shown in Figure 4.8. It is clearly visible that all best feature algorithm combinations performs better as compared to all individual algorithms. Similar to individual feature algorithms, it is not a good idea to use a single feature algorithm combination for summary generation. As different feature algorithm combinations have different inherent properties. In comparison with work carried out by Rafael et al [85,86] reported best combination is COMB1 contains two graph level and one sentence level features. Moreover, no dependent features such as cue words [10] , named entity [21] etc. are present in the reported best combinations.

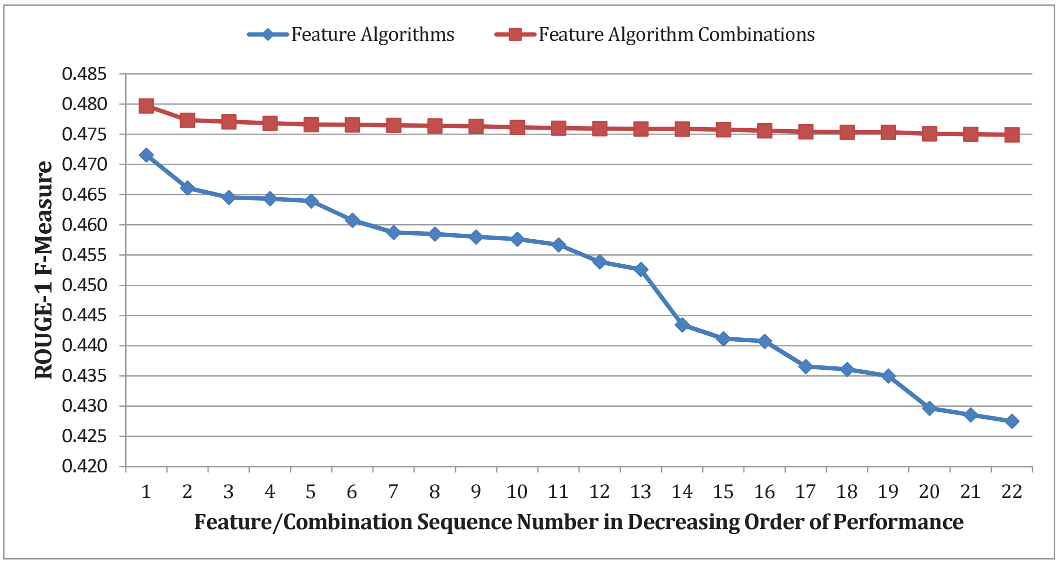


Figure 4.8: Variation in ROUGE-1 f-measure for whole document set for single features algorithms and their combinations.

4.8.3 Impact on Size of Text Document

Comparisons in between ROUGE-1 scores for short size, medium size, large size and all set of documents are shown in Figure 4.9. It is observed from the figure that performance of summarization system decreases as the size of document increases. This degradation is due to increase in the number of sentences because as the number of sentences increases, probability of selection of informative sentences decreases. ROUGE-2 scores comparisons for short size, medium size, large size and all set of documents are shown in Figure 4.10. Here also performance decreases with the increase in the size of input text document.

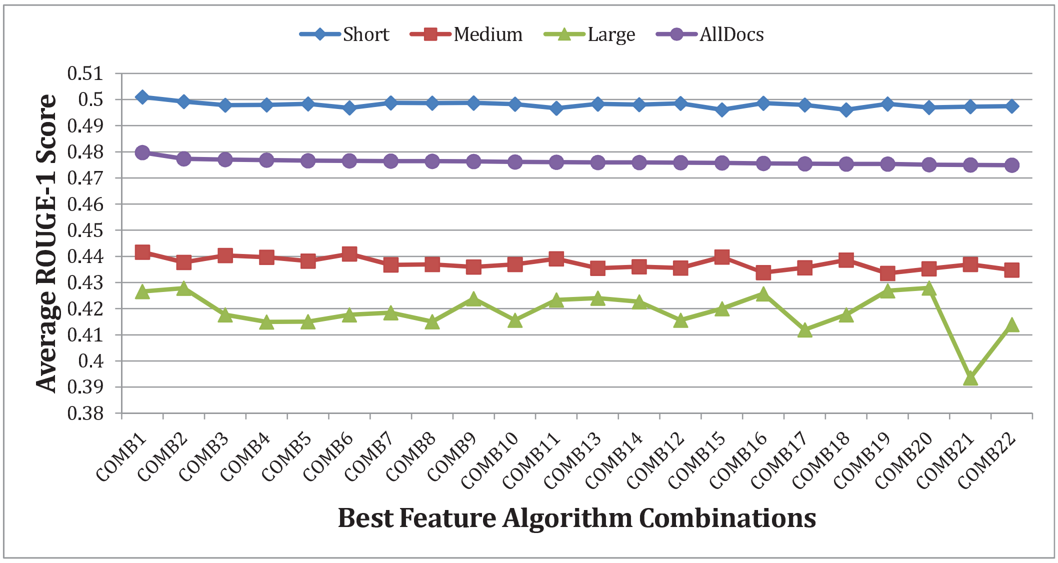


Figure 4.9: Variation in ROUGE-1 f-measure for different size document for best feature algorithm combinations.

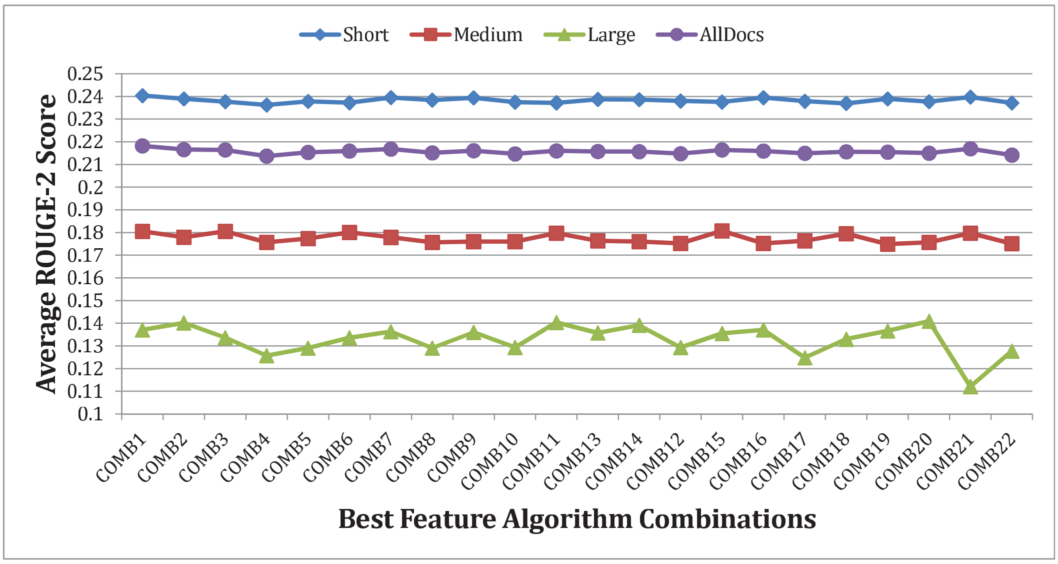


Figure 4.10: Variation in ROUGE-2 f-measure for different size document for best feature algorithm combinations.

4.9 Summary

In this chapter performance of various individual features algorithms such as bushy path, cosine similarity, cue words etc. is tested using ROUGE evaluation measures. It is observed that single feature algorithms only use specific feature strengths that are not capable of generating efficient summaries. Stemming impact is also analyzed while pre-processing of input text document, it is observed that with stemming performance improves significantly. Thereafter, feature algorithm combinations are generated using best performing single feature algorithms using ROUGE evaluation measures. Feature algorithm combinations which performs well for all ROUGE evaluation measures are reported as best feature algorithm combinations. Their performance is investigated on short, medium and large size documents. It is observed that performance degrades as the size of document increases. The generated feature algorithm combinations will be used for further improvement in quality of text summaries in Chapter 5 and Chapter 6.

Chapter 5

Sentence Filtering Using Feature Algorithm Combinations

Features algorithms extract specific themes or content from the input source text document. Their combinations also extract sentences in the similar way focusing on specific criteria. The issue of feature weighting (identification of weights for each feature) for sentence scoring, as discussed earlier in Chapter 2, can be resolved if only the linear combination of feature algorithms is considered for sentence scoring. Further, these feature algorithm combinations if applied, in a specific sequence to filter out the sentences and this might improve the performance of summarization process. This process that basically does sentence removal instead of sentence selection is discussed in detail in the following sections. Results of sentence filtering algorithm indicate that feature algorithm combinations used in a specific permutation enhances the performance of summarization process.

5.1 Related Sentence Removal Techniques

Mostly text summarization systems use techniques of sentence selection for the summary generation. The sentence removal/rejection was first proposed by Rush et al. in 1971 [87]. They used a set of specific rules to reject sentences; that was later extended by Pollock et al. for chemical abstracts [88]. Rush et al. developed nineteen rules to extract the sentences using Word Control List prepared by them. Considering these nineteen rules at that time it was much difficult to process computation task due to memory limitations. World Control List contains a set of words that are specifically historical data, examples, explanations, results of already reported words, etc. One of the rules is, sentences that contain question marks and equal signs must be deleted. This rule specifically removes equations and sentences containing questions. One of the other rule reported by them is that sentences containing figures and tables should also be removed from the set of potential candidates for the final summary. In 2000 Jing [89] proposed a new sentence phrase reduction method that removes sentence phrases on the basis of extraneous ones. These phrases are extracted similarly to the work of Rush et al. [87]. The phrase reduction systems used multiple sources of knowledge for phrase extraction. From a corpus, Jing extracted phrases using statistics, context information and syntactic knowledge. Another sentence removal method was proposed by Bonzanini et al. in 2013 [90]. They used a topic to find the similarity of it to all other sentences. Initially, all sentences are grouped in candidate summary set. Later on one sentence is rejected in each iteration until the desired size of the summary is not produced. The sentence that has less similarity score with the given topic is removed in each iteration. The method focuses only on a single topic. However, in general, more than one topics are present in the text documents. All the above discussed methods only utilize specific topic or set of words for sentence rejection. This motivated to use multiple levels of rejection so that quality of text summary might improved further.

5.2 Sentence Filtering Process Flow

The process flow for sentence filtering which involves mainly pre-processing, filtering permutation generation using best feature algorithm combinations, sentence scoring, sentence filtering, summary generation and ROUGE evaluation is shown in figure 5.1. Pre-processing to the input text is applied as described earlier in Chapter 2, best feature algorithms extracted in Chapter 4 are used for generation of permutations. Permutation generator generates permutation sequences of different size for best feature combinations. These feature algorithm combination permutation sequences are used to score sentences using linear combinations of feature algorithms as discussed in Chapter 4. Using the scores obtained, lower scoring sentences are flagged out in sentence filtering process. Sentence scoring and filtering process is repeated for all the generated sequences using permutation generator. All other permutations generated by permutation generator of feature algorithm combinations are further used to find the best filtering permutations. Every feature algorithm combinations gets equal chance to filter weak sentences (which are not information rich sentences) since proposed sentence filtering process

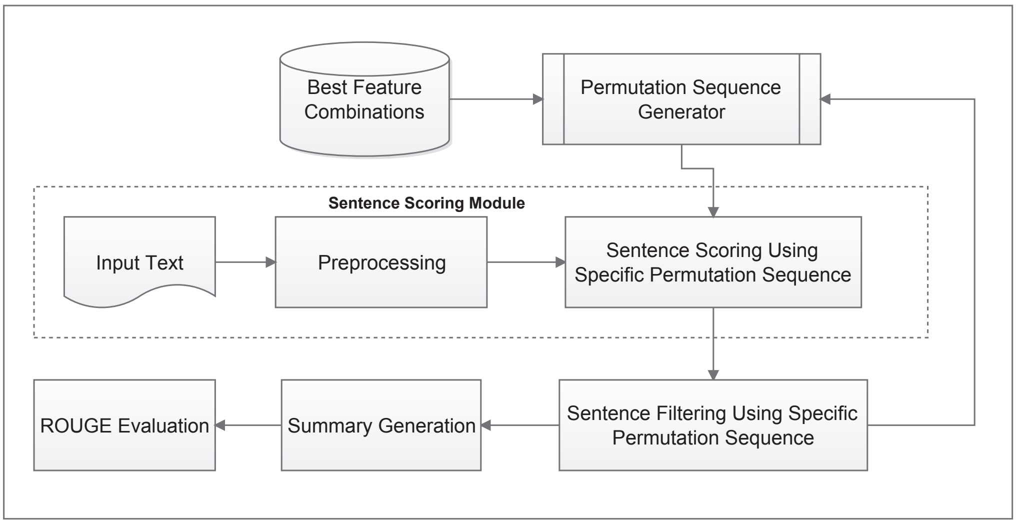

Best Feature Combinations
Permutation Sequence Generator
Sentence Scoring Module
Input Text
Preprocessing
Sentence Scoring Using Specific Permutation Sequence
ROUGE Evaluation
Summary Generation
Sentence Filtering Using Specific Permutation Sequence

Figure 5.1: Summarization process flow using sentence filtering.

use multiple levels for sentence filtering as compared to other proposed sentence removal [87] [89] [90] techniques. The whole sentence filtering process is presented in Algorithm 1.

Feature algorithm combinations wise score of all sentences is stored in a temporary two dimensional vector. A threshold value H to filter sentences is computed as the fraction of difference of total number of sentences and number of sentences required in final summary, divided by total number of feature algorithm combinations participating for summarization process. These total number of participating feature algorithm combinations are termed as filtering permutation sequences. Using the scores of these filtering permutations, the threshold value eliminates(rejects) H number of sentences in each level of filtering permutation sequence. A variable that stores the processing information of sentences, marks each sentence in filtering level so that its further processing will not take place. In other words, the sentence is no more going to be considered for summarization process. Sentences that are remaining after sentence filtering process are reported as the final summary sentences. However, in case number of remaining sentences after sentence filtering are more than required number of sentences in final summary, then the last filtering permutation sequence scores are used to select the sentences for the final summary. Sentences are sorted according to the last filtering permutation and top ranked sentences are selected for the final summary. The sentences are placed in the same order as they appear in the original text document in the final summary. These sentences selected using proposed sentence filtering approach are used to compare with a gold summary (reference summary provided for comparison with system generated summary) for evaluation purpose.

Algorithm 1 Proposed sentence filtering algorithm.

```
INITIALISATION:
$N$: Total number of sentences in text document.
$N_s$: Number of sentences required in summary.
$P$: Number of levels used for filtering process.
$D = \{S_1, S_2, \cdots, S_N\}$    // Text document containing sequence of $N$ sentences.
$L = \{L_1, L_2, \cdots, L_P\}$    // Level sequence.
$score(S_k, L_i)$: Score of sentence $S_k$ due to level $L_i$.

$H = (N\text{-}N_s)/P$          // Filtering Threshold
for Every statement $S_i$ in document D, where $i = 1$ to $N$ do
    $flag(S_i) \leftarrow 0$
end for
for Every filtering level $L_i$, where $i = 1$ to $P$ do
    for $j = 1$ to $H$ do
        $min \leftarrow 999$
        $index \leftarrow -1$
        for Every statement $S_k$ in document D, where $k = 1$ to $N$ do
            if $flag(S_k) = 0$ and $min > score(S_k, L_i)$ then
                $min \leftarrow score(S_k, L_i)$
                $index \leftarrow k$
            end if
        end for
        $flag(S_{index}) \leftarrow 1$
    end for
end for
$k \leftarrow 0$
$Summary = \phi$
for Every statement $S_i$ in document D, where $i = 1$ to $N$ do
    if $flag(S_i)$ ==0 then
        $Summary \leftarrow Summary + S_i$
        $k \leftarrow k + 1$
    end if
end for
```

For example, let the number of sentences in the given text document, numbers of sentences required in summary and total filtering levels for sentence filtering are sixteen, ten and three, respectively. The sentence scores due to each level are shown in Table 5.1.

Table 5.1: Ranks of sentence with different level sequences.

Sequence	Sentence ID															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
COMB1	0.12	0.04	0.11	0.14	0.02	0.62	0.3	0.63	0.76	1	0.05	0.62	0.24	0.93	0.54	0.67
COMB2	0.45	0.87	0.67	0.08	0.31	0.99	0.35	0.65	0.32	0.09	0.03	0.76	0.94	0.36	0.62	0.88
COMB3	0.49	0.01	0.26	0.95	0.6	0.54	0.36	0.8	0.88	0.43	0.2	0.08	0.03	0.67	0.67	0.62

Filtering of sentences using each level sequence is shown in Figure 5.2. There are sixteen nodes shown with their corresponding sentence IDs. Sentence are flagged depending on the current filtering level. Node 5 and 11 are filtered in level 1, node 4 and 10 are filtered to the second level. In third filtering level node 2 and 13 are filtered out.

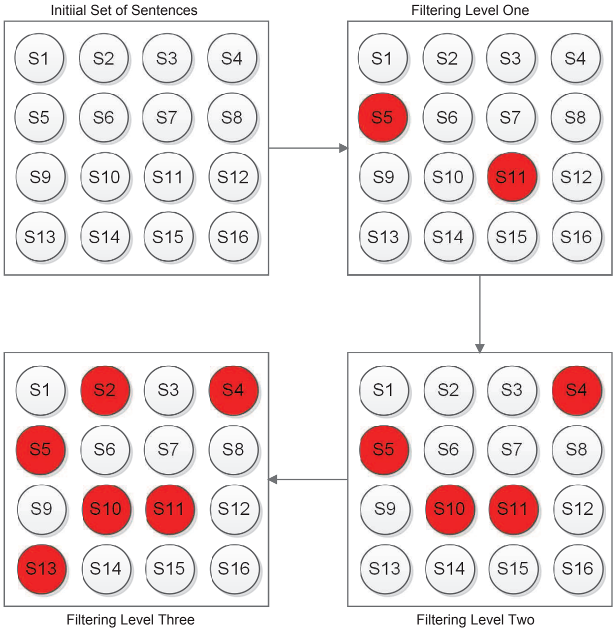


Figure 5.2: Level-wise filtering of sentences.

5.2.1 An Example : Sentence Filtering

For the example text document given in Subsection 2.2.1 of Chapter 2, firstly, the aggregate score of each sentence is calculated by adding the individual feature score of the corresponding sentence using two feature algorithms combination COMB1 and COMB2. COMB1 is combination of sentence location1(FA15), aggregate similarity(FA12) and lexrank(FA9) while COMB2 is combination of sentence Location1(FA15), sentence to sentence cohesion(FA17) and lexrank(FA9) as described below:

$$COMB1\_Score(S_1) = SL(S_1) + AS(S_1) + NLR(S_1)$$
$$COMB2\_Score(S_1) = SL(S_1) + NSSCS(S_1) + NLR(S_1)$$

COMB1 and COMB2 scores of each sentence is shown in Table 5.1.

Table 5.2: COMB1 and COMB2 scores of each sentence.

Sentence No.	**COMB1 Score**	**COMB2 Score**
S_1	2.0227	2.5454
S_2	1.3500	2.3500
S_3	1.1833	1.6378
S_4	1.2727	2.2500
S_5	1.2227	1.200
S_6	1.0166	1.4712
S_7	1.1655	1.2337
S_8	0.9750	1.6113

After calculating combination scores for each sentence, filtering is applied. In the first iteration, two sentences having lowest COMB1 score are flagged and then in the second iteration, two sentences from remaining sentences having lowest COMB2 score are flagged. Sentences S_8 and S_6 has the lowest COMB1 score, so these sentences are flagged in the first iteration. From remaining sentences, sentence S_5 and S_7 has the lowest COMB2 score, so these sentences are flagged in the second iteration. Thus, sentences S_5, S_6, S_7 and S_8 are flagged and does not consider to be included into the summary.

5.3 Experimental Setup and Evaluation Measures

In order to investigate the performance of feature algorithm combination based sentence filtering process top twenty two feature algorithm combinations reported in Chapter 4 are used. These feature algorithm combinations are considered as different levels. All these levels are used for making different permutations of different sizes. For example, if three feature combination clusters named as COMB1, COMB2 and COMB3 are used, then possible permutations of these are shown in table 5.3.

For experiments, similar to the Chapter 4, the DUC 2002 dataset is divided into three groups. In total, we have four group of documents namely short, medium, large and all documents. For all twenty two feature algorithm combinations, different permutations of different size are generated as described in Section 5.2. ROUGE evaluation measure as described in section 4.5 of Chapter 4 are used for

Table 5.3: Feature algorithm combination permutations.

Sequence ID	Features Algorithm Combination Cluster Sequence
123	COMB1 → COMB2 → COMB3
232	COMB1 → COMB3 → COMB2
213	COMB2 → COMB1 → COMB3
231	COMB2 → COMB3 → COMB1
312	COMB3 → COMB1 → COMB2
321	COMB3 → COMB2 → COMB1

performance testing. Each of the ROUGE-1, ROUGE-2, ROUGE-L and ROUGE-W scores are computed for performance evaluation for each of the text documents.

5.4 Results

Sentence filtering is done using different size groups of feature algorithm combinations. Level sequences of results reported in this section are of size two, three, four and five only. Sequences of size more than five performs poorly, therefore the results are not reported.

5.4.1 Best ten filtering sequences for sentence filtering.

Best ten sequences selected on the basis of ROUGE-1 f-measure are presented in Table 5.4 for all document group. Each sequence is renamed as BFP(best filtering permutation) with a sequence id as index number of respective feature algorithm combinations as shown in Table 5.4.

Table 5.4: Best ten feature algorithm combinations filtering sequences.

S. No.	Sequence ID	Features Sequence	Short Name
1	126181	COMB12 → COMB6 → COMB18 → COMB1	BFP1
2	228111	COMB22 → COMB8 → COMB11 → COMB1	BFP2
3	612181	COMB6 → COMB12 → COMB18→ COMB1	BFP3
4	4231	COMB4 → COMB2 → COMB3 → COMB1	BFP4
5	1741	COMB17 → COMB4 → COMB1	BFP5
6	941	COMB9 → COMB4 → COMB1	BFP6
7	42531	COMB4 → COMB2 → COMB5 → COMB3 → COMB1	BFP7
8	24531	COMB2 → COMB4 → COMB5 → COMB3 → COMB1	BFP8
9	4171	COMB4 → COMB17 → COMB1	BFP9
10	8211119	COMB8 → COMB21 → COMB11 → COMB1	BFP10

Each feature algorithm combination level sequence filters H number of sentences as describes in Section 5.2.

5.4.2 ROUGE scores for All Document

ROUGE-1 and ROUGE-2 scores of sentence filtering using feature algorithm combinations sequences, on all documents set are presented in Table 5.5. This group contains five hundred sixty three documents. Results of best sentence filtering sequences using ROUGE-L and ROUGE-W on same document group are reported in Table 5.6. ROUGE-1, ROUGE-2 , ROUGE-L and ROUGE-W scores of BFP1 are higher as compared to all other filtering permutations and reported as 0.48317, 0.22038, 0.40555 and 0.40184.

Table 5.5: ROUGE-1 and ROUGE-2 scores of best filtering permutations for all documents.

S. No.	**Method**	**ROUGE-1**			**ROUGE-2**		
		Precision	Recall	F-Measure	Precision	Recall	F-Measure
1	BFP1	0.46983	0.49729	0.48317	0.21415	0.22697	0.22038
2	BFP2	0.46933	0.49699	0.48276	0.21365	0.22657	0.21992
3	BFP3	0.46952	0.49664	0.48270	0.21387	0.22657	0.22004
4	BFP4	0.46912	0.49701	0.48266	0.21364	0.22657	0.21992
5	BFP5	0.46926	0.49678	0.48263	0.21326	0.22616	0.21952
6	BFP6	0.46932	0.49669	0.48262	0.21322	0.22613	0.21949
7	BFP7	0.46924	0.49673	0.48259	0.21358	0.22638	0.21979
8	BFP8	0.46935	0.49659	0.48258	0.21370	0.22639	0.21986
9	BFP9	0.46855	0.49747	0.48258	0.21297	0.22652	0.21954
10	BFP10	0.46926	0.49667	0.48257	0.21345	0.22634	0.21970

Table 5.6: ROUGE-L and ROUGE-W scores of best filtering permutations for all documents.

S. No.	**Method**	**ROUGE-L**			**ROUGE-W**		
		Precision	Recall	F-Measure	Precision	Recall	F-Measure
1	BFP1	0.39437	0.41737	0.40555	0.39082	0.41350	0.40184
2	BFP2	0.39399	0.41714	0.40523	0.39046	0.41329	0.40155
3	BFP3	0.39415	0.41685	0.40518	0.39057	0.41295	0.40145
4	BFP4	0.39358	0.41696	0.40493	0.39005	0.41309	0.40124
5	BFP5	0.39398	0.41703	0.40518	0.39049	0.41324	0.40155
6	BFP6	0.39403	0.41698	0.40518	0.39053	0.41318	0.40153
7	BFP7	0.39365	0.41671	0.40485	0.39018	0.41292	0.40122
8	BFP8	0.39391	0.41672	0.40500	0.39046	0.41298	0.40140
9	BFP9	0.39333	0.41761	0.40511	0.38973	0.41368	0.40135
10	BFP10	0.39386	0.41681	0.40501	0.39037	0.41301	0.40137

5.4.3 ROUGE scores for Short Size Group.

ROUGE-1 and ROUGE-2 scores of sentence filtering using feature algorithm combinations sequences, on all documents set are presented in Table 5.7. Results of best filtering permutations using ROUGE-L and ROUGE-W on same document group are reported in Table 5.8. BFP9 performs better than BFP1 for ROUGE-1, ROUGE-L and ROUGE-W f-measure. However, for ROUGE-2 f-measure value of BFP1 is higher as compared to BFP9.

Table 5.7: ROUGE-1 and ROUGE-2 scores of best filtering permutations for short size group.

S. No.	Method	ROUGE-1			ROUGE-2		
		Precision	Recall	F-Measure	Precision	Recall	F-Measure
1	BFP1	0.48964	0.51987	0.50431	0.23518	0.25007	0.24240
2	BFP2	0.48940	0.51983	0.50416	0.23471	0.24968	0.24196
3	BFP3	0.48921	0.51918	0.50375	0.23475	0.24958	0.24194
4	BFP4	0.48937	0.51975	0.50410	0.23495	0.24985	0.24217
5	BFP5	0.48892	0.51946	0.50373	0.23441	0.24946	0.24170
6	BFP6	0.48909	0.51937	0.50377	0.23442	0.24943	0.24170
7	BFP7	0.48902	0.51953	0.50381	0.23445	0.24943	0.24171
8	BFP8	0.48929	0.51941	0.50390	0.23466	0.24945	0.24183
9	BFP9	0.48938	0.52037	0.50440	0.23480	0.25015	0.24223
10	BFP10	0.48873	0.51917	0.50349	0.23407	0.24910	0.24135

Table 5.8: ROUGE-L and ROUGE-W scores of best filtering permutations for short size group.

S. No.	Method	ROUGE-L			ROUGE-W		
		Precision	Recall	F-Measure	Precision	Recall	F-Measure
1	BFP1	0.41200	0.43754	0.42439	0.40787	0.43305	0.42008
2	BFP2	0.41176	0.43745	0.42422	0.40757	0.43290	0.41986
3	BFP3	0.41171	0.43702	0.42399	0.40754	0.43250	0.41965
4	BFP4	0.41170	0.43744	0.42418	0.40768	0.43308	0.42000
5	BFP5	0.41178	0.43760	0.42430	0.40769	0.43319	0.42005
6	BFP6	0.41188	0.43753	0.42432	0.40780	0.43312	0.42008
7	BFP7	0.41142	0.43727	0.42395	0.40742	0.43292	0.41978
8	BFP8	0.41177	0.43725	0.42413	0.40780	0.43295	0.42000
9	BFP9	0.41197	0.43824	0.42470	0.40786	0.43377	0.42042
10	BFP10	0.41133	0.43706	0.42381	0.40724	0.43262	0.41954

5.4.4 ROUGE scores for Medium Size Group

ROUGE-1 and ROUGE-2 scores of sentence filtering using feature algorithm combinations sequences, on medium size document set are reported in Table 5.9. Results of best filtering permutations using ROUGE-L and ROUGE-W on same document group are presented in Table 5.10. ROUGE-1, ROUGE-2 , ROUGE-L and ROUGE-W scores of BFP1 are higher as compared to all other filtering permutations and reported as 0.44557, 0.18312, 0.37283 and 0.36994.

Table 5.9: ROUGE-1 and ROUGE-2 scores of best filtering permutations for medium size group.

S. No.	**Method**	**ROUGE-1**			**ROUGE-2**		
		Precision	Recall	F-Measure	Precision	Recall	F-Measure
1	BFP1	0.43531	0.45633	0.44557	0.17885	0.18759	0.18312
2	BFP2	0.43438	0.45555	0.44471	0.17830	0.18719	0.18263
3	BFP3	0.43523	0.45570	0.44523	0.17883	0.18734	0.18299
4	BFP4	0.43332	0.45610	0.44442	0.17759	0.18690	0.18213
5	BFP5	0.43447	0.45599	0.44497	0.17744	0.18641	0.18182
6	BFP6	0.43434	0.45589	0.44485	0.17731	0.18639	0.18174
7	BFP7	0.43465	0.45551	0.44484	0.17854	0.18710	0.18272
8	BFP8	0.43445	0.45529	0.44463	0.17847	0.18709	0.18268
9	BFP9	0.43140	0.45629	0.44349	0.17579	0.18616	0.18083
10	BFP10	0.43496	0.45568	0.44508	0.17861	0.18737	0.18289

Table 5.10: ROUGE-L and ROUGE-W scores of best filtering permutations for medium size group.

S. No.	**Method**	**ROUGE-L**			**ROUGE-W**		
		Precision	Recall	F-Measure	Precision	Recall	F-Measure
1	BFP1	0.36440	0.38166	0.37283	0.36164	0.37862	0.36994
2	BFP2	0.36381	0.38121	0.37231	0.36126	0.37837	0.36962
3	BFP3	0.36429	0.38110	0.37251	0.36151	0.37802	0.36958
4	BFP4	0.36225	0.38090	0.37134	0.35929	0.37757	0.36821
5	BFP5	0.36322	0.38085	0.37183	0.36055	0.37788	0.36901
6	BFP6	0.36318	0.38084	0.37180	0.36043	0.37782	0.36892
7	BFP7	0.36327	0.38034	0.37161	0.36051	0.37728	0.36871
8	BFP8	0.36335	0.38042	0.37169	0.36062	0.37741	0.36882
9	BFP9	0.36086	0.38138	0.37084	0.35786	0.37806	0.36768
10	BFP10	0.36386	0.38089	0.37218	0.36123	0.37795	0.36940

5.4.5 ROUGE scores for Large Size Group.

ROUGE-1 and ROUGE-2 scores of sentence filtering using feature algorithm combinations sequences, on all documents set are presented in Table 5.11. Results of best filtering permutations using ROUGE-L and ROUGE-W on same document group are reported in Table 5.12. ROUGE-1, ROUGE-2 , ROUGE-L and ROUGE-W scores of BFP10 are higher as compared to all other filtering permutations and reported as 0.42972, 0.13895, 0.34619 and 0.34750. However, all top filtering permutations of feature algorithm combinations perform in similar range with small variation.

Table 5.11: ROUGE-1 and ROUGE-2 scores of best filtering permutations for large size group.

S. No.	Method	ROUGE-1			ROUGE-2		
		Precision	Recall	F-Measure	Precision	Recall	F-Measure
1	BFP1	0.40690	0.44689	0.42596	0.12990	0.14234	0.13583
2	BFP2	0.40539	0.44619	0.42481	0.12911	0.14163	0.13508
3	BFP3	0.40690	0.44689	0.42596	0.12990	0.14234	0.13583
4	BFP4	0.41167	0.44162	0.42612	0.13190	0.14092	0.13626
5	BFP5	0.41386	0.44162	0.42729	0.13245	0.14092	0.13655
6	BFP6	0.41386	0.44162	0.42729	0.13245	0.14092	0.13655
7	BFP7	0.40797	0.44409	0.42527	0.13017	0.14163	0.13566
8	BFP8	0.40797	0.44409	0.42527	0.13017	0.14163	0.13566
9	BFP9	0.41386	0.44162	0.42729	0.13245	0.14092	0.13655
10	BFP10	0.41230	0.44867	0.42972	0.13352	0.14485	0.13895

Table 5.12: ROUGE-L and ROUGE-W scores of best filtering permutations for large size group.

S. No.	Method	ROUGE-L			ROUGE-W		
		Precision	Recall	F-Measure	Precision	Recall	F-Measure
1	BFP1	0.32889	0.36069	0.34406	0.33002	0.36212	0.34532
2	BFP2	0.32733	0.35964	0.34272	0.32819	0.36072	0.34369
3	BFP3	0.32889	0.36069	0.34406	0.33002	0.36212	0.34532
4	BFP4	0.33313	0.35681	0.34456	0.33466	0.35860	0.34622
5	BFP5	0.33424	0.35611	0.34483	0.33572	0.35790	0.34646
6	BFP6	0.33424	0.35611	0.34483	0.33572	0.35790	0.34646
7	BFP7	0.32979	0.35859	0.34359	0.33055	0.35967	0.34450
8	BFP8	0.32979	0.35859	0.34359	0.33055	0.35967	0.34450
9	BFP9	0.33424	0.35611	0.34483	0.33572	0.35790	0.34646
10	BFP10	0.33250	0.36105	0.34619	0.33370	0.36249	0.34750

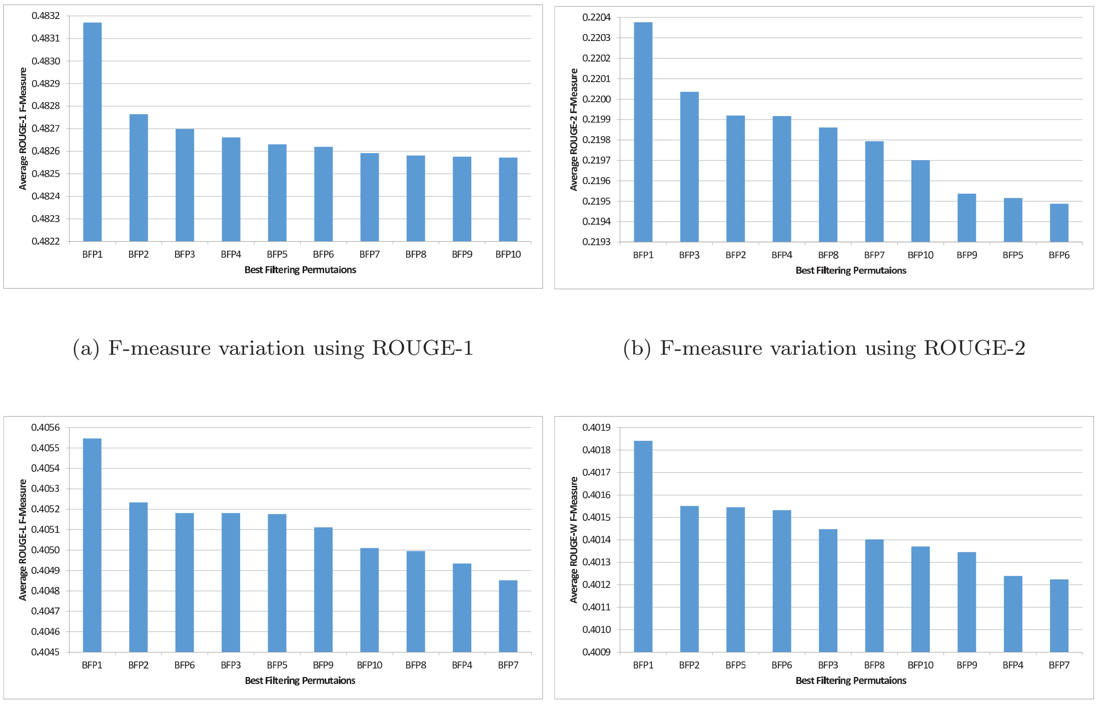
BFP1 performs better for short and medium size documents as compared to all other filtering permutations. Analysis of best filtering permutations obtained using the feature algorithm combination clusters is discussed in the following section.

5.5 Analysis

Analysis of results obtained in the previous section on the basis of size and with in the document group using ROUGE evaluation measure are discussed in following section.

5.5.1 Average Performance Analysis on All Document Set

F-measure comparisons of best filtering permutations for all document set using ROUGE evaluation measures is presented in Figure 5.3. Results are sorted on the basis of f-measure. It is observed that BFP1 performs better as compared to all other filtering permutations. Filtering sequence of BFP1 is COMB12 → COMB6 → COMB18 → COMB1. COMB12 scores each sentence using linear sum of feature



(a) F-measure variation using ROUGE-1

(b) F-measure variation using ROUGE-2

(c) F-measure variation using ROUGE-L

(d) F-measure variation using ROUGE-W

Figure 5.3: F-measure variation of best filtering permutations for all document set using ROUGE evaluation measures.

algorithms such as depth of the sentence in the tree [84], term frequency [8], lexrank [25] and sentence location1 [9]. Specific number of sentences are filtered out using this feature algorithm combination. Same process continued for rest of the other filtering combinations. In BFP2 the filtering permutation sequence is COMB22 → COMB8 → COMB11 → COMB1. This sequence filters sentence using different combinations as compared to BFP1. Specifically, performance depends on the strength of the specific filtering permutations.

5.5.2 Performance Analysis for Size of Input Documents

Comparison of average ROUGE-1 scores of different size input data groups is shown in Figure 5.4. For all the filtering permutations, it is observed that with the increase in size (number of sentences) of the text document, the performance using all the ROUGE evaluation measures decreases. This is mainly due to increase in the size of lexicon that ultimately, lowers the probability of selection of a particular sentence.

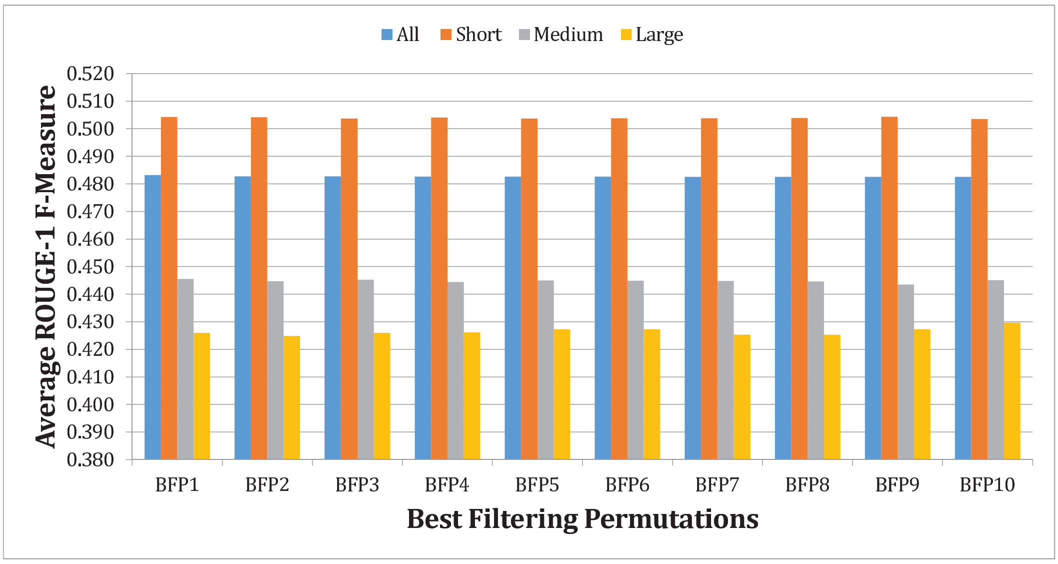


Figure 5.4: Variation in Precision, Recall and F-Measure for different size document using ROUGE-1.

Another reason of performance degradation due to increase in the number of sentences is, increase in redundant text. Same information is sometimes repeated in the given text. Depending on the selection methodology, even with low probability of selection, chances are still their that redundant sentence might be selected in the final summary. While evaluation using ROUGE matrices that is primarily based on N-grams and LCS, the common matching value between system generated summary and gold summary is decreased.

5.6 Comparison with Existing Approaches

Comparison in between best filtering permutation (BFP1), existing single feature algorithms such as bushy path(FA1) [24], depth of the sentence in the tree(FA4) [84], sentence to sentence cohesion(FA17) [84], title similarity(FA23) [10] and sentence location1(FA15) [9] using ROUGE-1 and ROUGE-2 is shown in Figure 5.5 and 5.6, respectively.

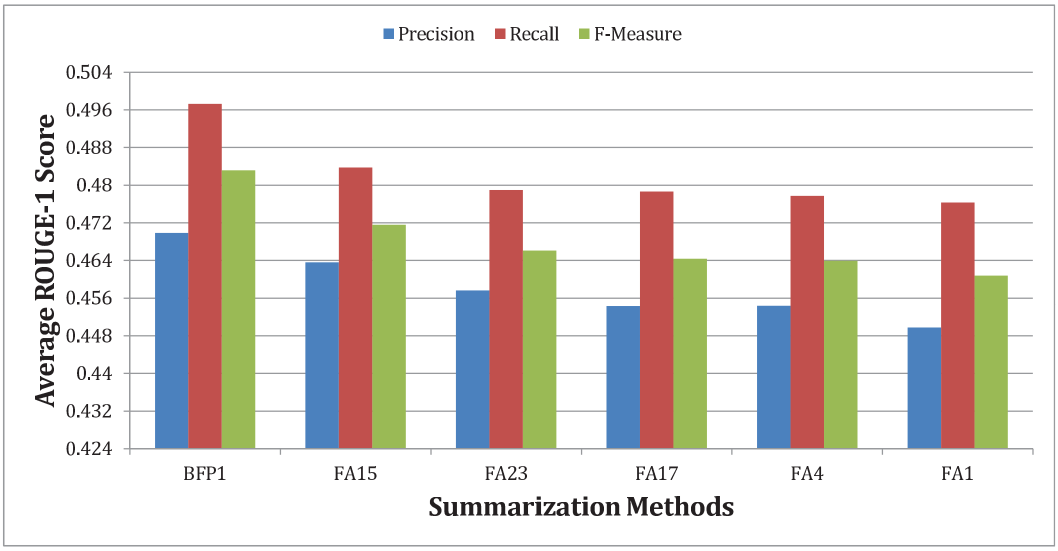


Figure 5.5: Comparison of existing best five feature algorithms with BFP1 using ROUGE-1.

Proposed filtering permutation BFP1 performs better as compared to all other individual feature algorithms. Comparison in between best filtering permutation (BFP1) and existing individual feature algorithms using ROUGE-L and ROUGE-W is shown in Figure 5.7 and 5.8, respectively.

Sentence Location1 performance is better as compared to other individual feature algorithms. Every feature algorithms generate summary as per its own strengths. However, feature algorithm combinations improves quality of summary using strengths of individual algorithms. Further, using combinations as filtering permutations improves the results.

Comparison in between best filtering permutation (BFP1), best combination(COMB1), professional tools such as MSWord and Copernic, lexrank, Cortex and LSA using

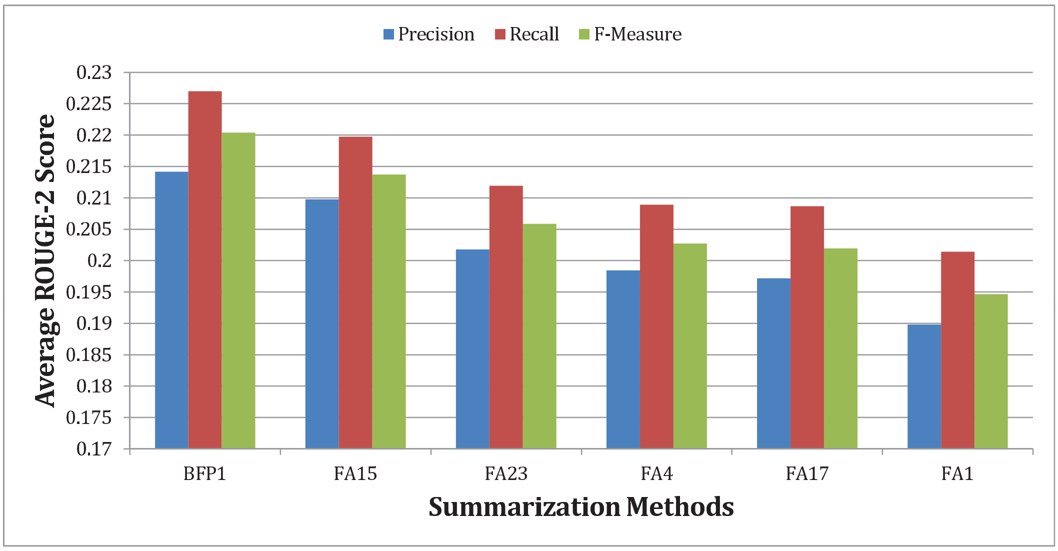


Figure 5.6: Comparison of existing best five feature algorithms with BFP1 using ROUGE-2.

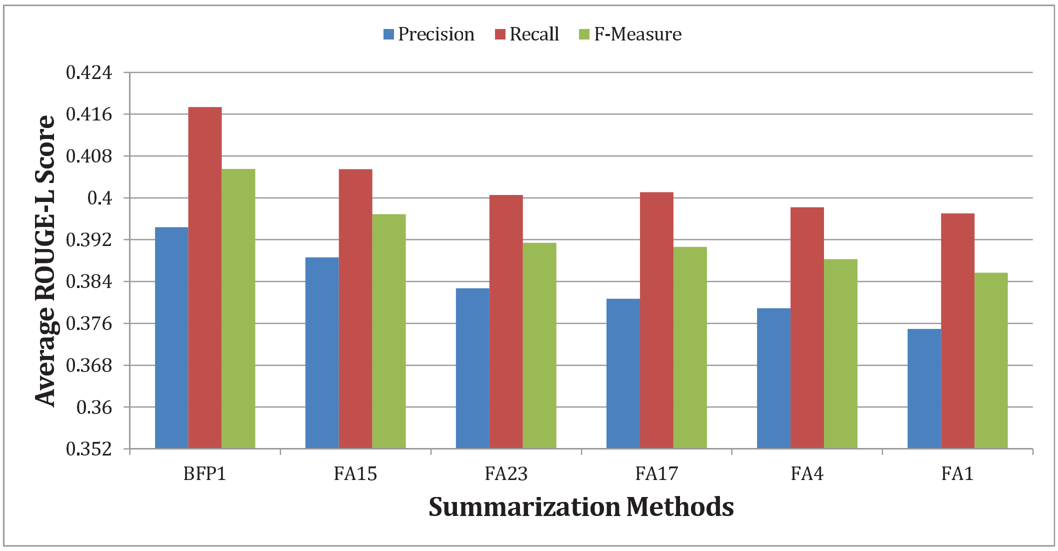


Figure 5.7: Comparison of existing best five feature algorithms with BFP1 using ROUGE-L.

ROUGE-1 and ROUGE-2 is shown in Figure 5.9 and 5.10, respectively.

Proposed filtering permutation BFP1 performs better as compared to all other feature based approaches. However, Copernic recall value is more than recall value of BFP1. This is due to higher value of compression in Copernic. The difference in between the values of precision and recall is very high that ultimately lowers the f-measure. Comparison in between best filtering permutation (BFP1), best combination(COMB1), MSWord, Copernic, lexrank, Cortex and LSA using ROUGE-L and ROUGE-W are shown in Figure 5.11 and 5.12, respectively.

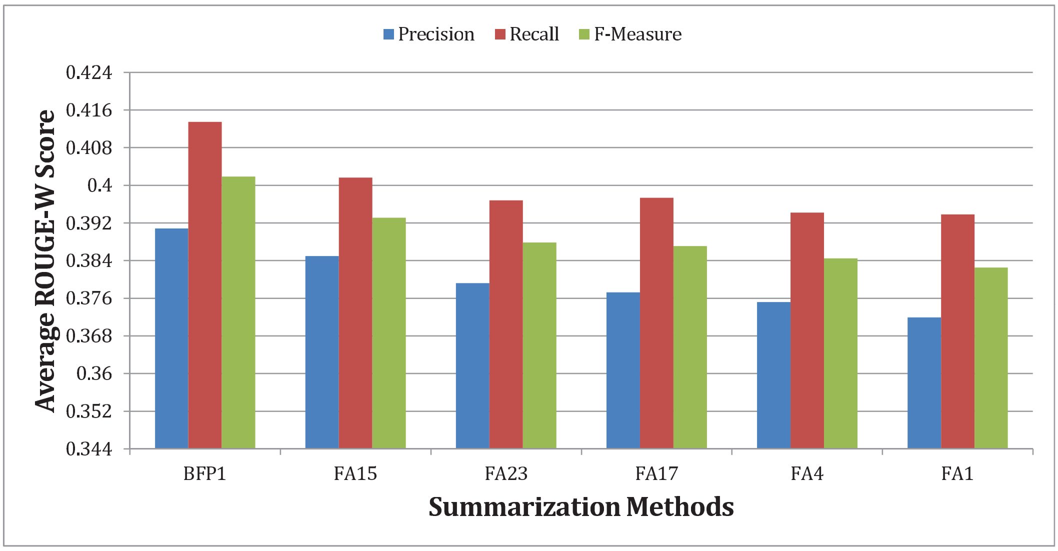


Figure 5.8: Comparison of existing best five feature algorithms with BFP1 using ROUGE-W.

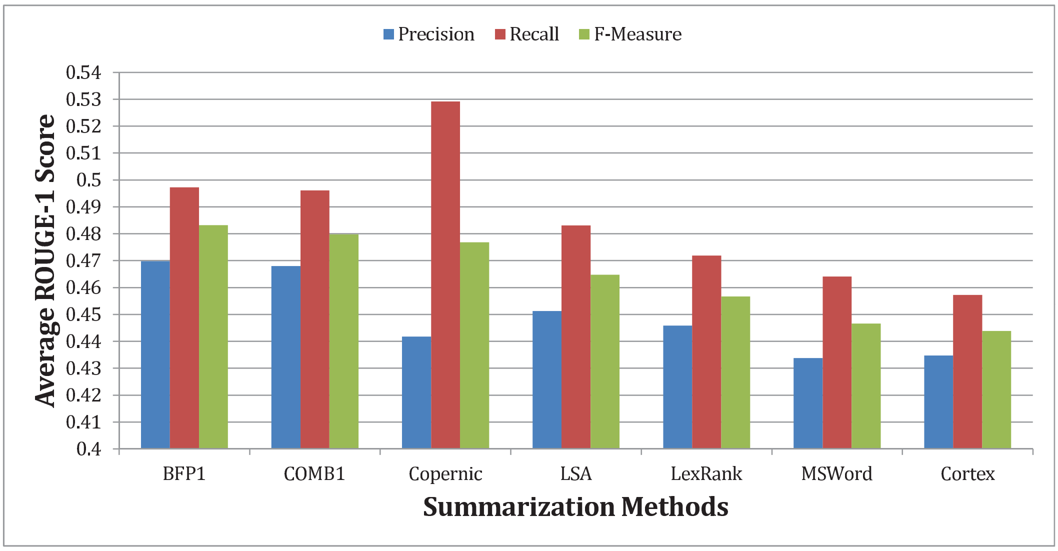


Figure 5.9: Comparison of existing approaches with BFP1 using ROUGE-1.

BFP1 performs better as compared to all other existing approaches for ROUGE-L f-measure. However, Copernic performs better as compared to other existing methods and performs in the similar range of BFP1. Copernic performs better as compared to BFP1 and all other approaches for ROUGE-W f-measure. However, precision value of BPF1 is more as compared Copernic. By comparing existing methods with best filtering permutation (BFP1), it has been observed that using the multiple level processing and strengths of best feature combinations, performance of BFP1 is better as compared to all other existing statistical and graph based methods. Professional tools also performs worse as compared to BFP1.

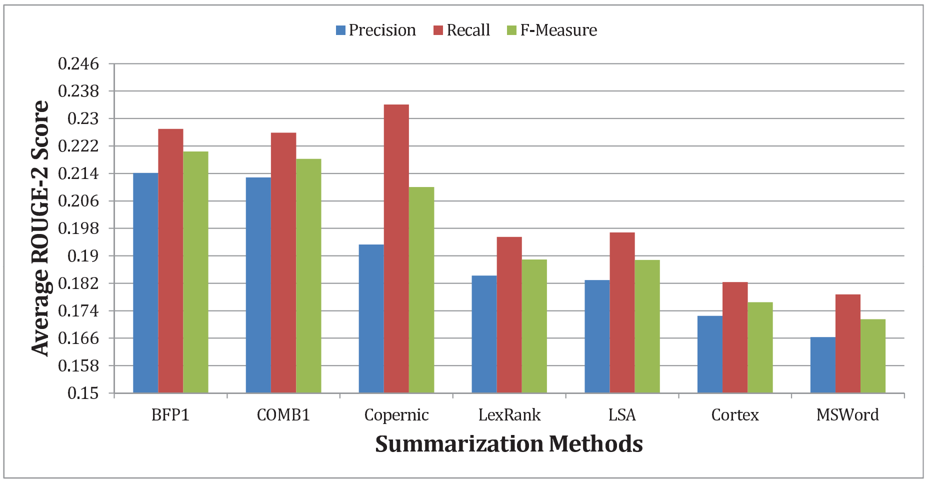


Figure 5.10: Comparison of existing approaches with BFP1 using ROUGE-2.

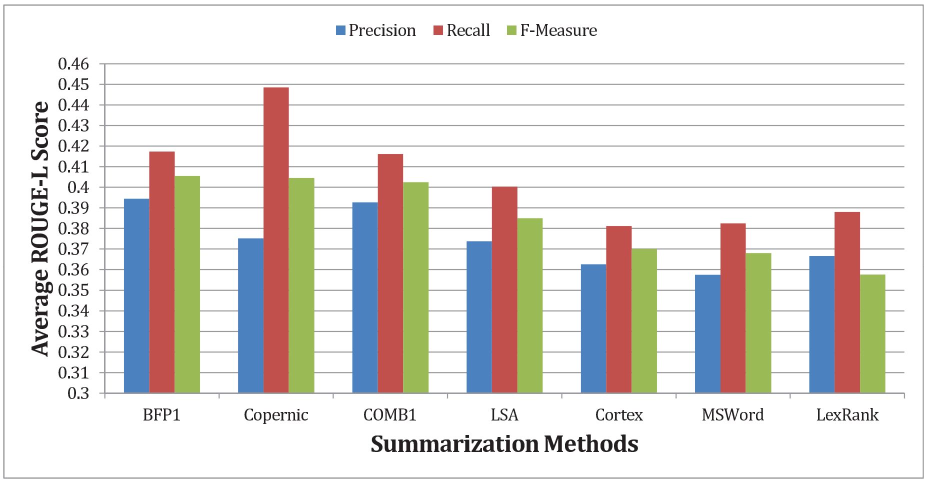


Figure 5.11: Comparison of existing approaches with BFP1 using ROUGE-L.

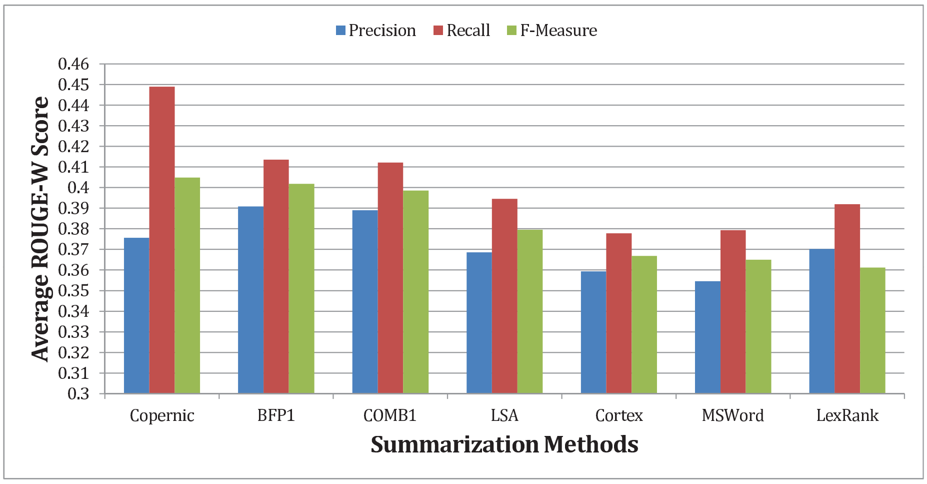


Figure 5.12: Comparison of existing approaches with BFP1 using ROUGE-W.

5.7 Summary

In this Chapter a sentence filtering method using feature algorithm combinations is proposed for summarization process. Best feature algorithm combination found in Chapter 4 are used to filter out the sentences using specific filtering permutation sequences. These filtering permutations improves the quality of text summaries using strengths of feature combinations and multiple levels of sentence filtering. Best filtering permutation(BFP1) performs better as compared to other state of the art methods for summarization process. The DUC 2002 dataset consists of news documents and proposed filtering process improves the performance on news domain effectively. Moreover, the same feature combinations and filtering process might generate efficient summaries in case of other domains too.

Chapter 6

Voting Based Sentence Filtering Approach

Similar to combining all feature algorithms, ensemble different approaches for text summarization do not improve the quality of generated text summaries. Sentence filtering improves the performance but to a certain extent. In order to further improve the performance, voting models which are basically proposed for expert search systems may be utilized in conjunction with sentence filtering. Voting models, their use in text summarization and integration with sentence filtering are discussed in the following sections.

6.1 Use of Voting Models for Text Summarization

Voting models are first proposed for expert search systems where user's requirements are to find people who have relevant expertise in a given topic of interest. Voting methods were proposed for expert search to aggregate the ranks of retrieved documents from the given set of documents. These methods initially rank retrieved documents using specific standard method termed as initial ranking process. Thereafter, a voting model is chosen with specific similarity measure to finally rank the retrieved documents. Voting based approach such as Votes [91], Reciprocal Rank (RR) [92], BordaFuse [93], CombANZ, CombMED, CombMIN, CombMAX, CombMNZ [94], expCombANZ, exCombSUM and expCombMNZ [95] for expert search systems were proposed by researchers. Effectiveness of these voting models [91] [96] was investigated in the Text Retrieval Conference (TREC) 2005 and 2006. For text summarization, specifically Kumar et al. [97] in 2013 proposed a cross document relationship based genetic approach using CombMAX voting model. Kumar et al. [98] in 2014 discussed about adaption of twelve voting models for text summarization which are originally proposed for expert search task. However, no experimental results were reported for proposed voting approach [98]. Voting methods are totally dependent on strength of initial ranking process and similarity measure used for finding set of voters for the given sentence. Various voting models proposed in literature are discussed in the following section.

6.2 Voting Models

Generally a voting process uses specific criterion depends on user's need to get the votes of candidate documents. In query based search systems once a set of candidate documents is retrieved their voters can be identified using any standard similarity measure such as Jaccard or Cosine (profile voting) with specific threshold value. In the similar way, for text summarization process, sentences are scored using a specific initial scoring process and then ranked according to their scores. For a given sentence S_i initial score is represented as $IntialScore(S_i)$ and intial rank is represented as $IntialRank(S_i)$. Thereafter, using the specific similarity measure, a set of voters S_{voters} is computed for each of the given sentence. Twelve voting models proposed for expert search task that are used for text summarization process are summarized in Table 6.1.

Table 6.1: Summary of the voting models.

Voting Model	**Renamed As**	**Score Calculation**
Votes	V_1	Size of S_{voters} (Number of voters)
Reciprocal Rank	V_2	Sum of inverse of initial ranks of sentences in S_{voters}
BordaFuse	V_3	Sum of (N-ranks of sentences in S_{voters})
CombMED	V_4	median of scores on documents in S_{voters}
CombMIN	V_5	minimum of scores on documents in S_{voters}
CombMAX	V_6	maximum of scores on documents in S_{voters}
CombSUM	V_7	sum of scores of documents in S_{voters}
CombANZ	V_8	CombSUM/Size of S_{voters} (Number of voters)
CombMNZ	V_9	Size of S_{voters} (Number of voters)xCombSUM
expCombSUM	V_{10}	sum of exponents of scores of documents in S_{voters}
expCombANZ	V_{11}	expCombSUM/Size of S_{voters} (Number of voters)
expCombMNZ	V_{12}	Size of S_{voters} (Number of voters)xexpCombSUM

These voters are then used for final ranking of sentences. Final score for a given sentence S_i for a specific voting model V_i is represented as $FinalScore(S_i, V_i)$. is represented as For the basic model Votes(V_1) [91], final score for a given sentence is updated as number of its voters as described in Algorithm 2. Firstly, final score is initialized to zero. If similarity Sim(S_i, S_j) in between two sentences S_i and S_j is more than the specified threshold then final score is updated. Here, for a sentence S_i flag label $flag(S_i)$ is used for heuristic purpose, if required such as sentence filtering described earlier in Chapter 5. Reciprocal Rank (RR)(V_2) [92] method sums the reciprocal (inverse) ranks of each voter of the candidate sentence as given in Algorithm 3. Similar to Votes, for a given sentence if similarity value exceeds threshold criteria then its inverse is added to final score.

Algorithm 2 Voting$_1$() : Votes

```
for Every statement $S_i$ in document D, where $i = 1$ to $N$ do
  $FinalScore(S_i, V_1) \leftarrow 0$
  for Every statement $S_j$ in document D, where $j = 1$ to $N$ do
    if $Sim(S_i, S_j) > Threshold$ and $i \neq j$ and $flag(S_j) == 0$ then
      $FinalScore(S_i, V_1) \leftarrow FinalScore(S_i, V_1) + 1$
    end if
  end for
end for
```

Algorithm 3 Voting$_2$() : Reciprocal Rank (RR)

```
for Every statement $S_i$ in document D, where $i = 1$ to $N$ do
  $FinalScore(S_i, V_2) \leftarrow 0$
  for Every statement $S_j$ in document D, where $j = 1$ to $N$ do
    if $Sim(S_i, S_j) > Threshold$ and $flag(S_j) == 0$ and $i \neq j$ then
      $FinalScore(S_i, V_2) \leftarrow FinalScore(S_i, V_2) + \frac{1}{InitialRank(S_j)}$
    end if
  end for
end for
```

BordaFuse(V_3) [93] voting model computes final score of a given sentence as the sum of total number of sentences minus rank of voter for all voters of given sentence as described in Algorithm 4. For each voter of given sentence S_i, difference of total number of sentences and initial rank of voter is summed to compute final score.

CombMED(V_4) computes final score of each candidate sentence as median of scores of voter sentences as described in Algorithm 5. After initializing final score to zero, a sorted list of initial scores is created, if meets the similarity criteria. Median of this list is reported as the CombMED score of the given sentence S_i. CombMIN(V_5)

Algorithm 4 Voting$_3$() : BordaFuse

```
for Every statement $S_i$ in document D, where $i = 1$ to $N$ do
    $FinalScore(S_i, V_3) \leftarrow 0$
    for Every statement $S_j$ in document D, where $j = 1$ to $N$ do
        if $Sim(S_i, S_j) > Threshold$ and $flag(S_j) == 0$ and $i \neq j$ then
            $FinalScore(S_i, V_3) \leftarrow FinalScore(S_i, V_3) + N - InitialRank(S_j)$
        end if
    end for
end for
```

Algorithm 5 Voting$_4$() : CombMED

```
for Every statement $S_i$ in document D, where $i = 1$ to $N$ do
    for Every statement $S_j$ in document D, where $j = 1$ to $N$ do
        if $Sim(S_i, S_j) > Threshold$ and $flag(S_j) == 0$ and $i \neq j$ then
            $List.add(InitialScore(S_j))$
        end if
    end for
    $FinalScore(S_i, V_4) \leftarrow$ Median of List
end for
```

[94] updates final score as minimum of scores of given sentence's voter sentences as presented in Algorithm 6. Firstly, the min value is initialized to a sufficiently large value i.e. 1000. Thereafter, the min value of score of all possible voters is calculated. This value is finally reported as CombMIN score of respective sentence.

Algorithm 6 Voting$_5$() : CombMIN

```
$min \leftarrow 1000$
for Every statement $S_i$ in document D, where $i = 1$ to $N$ do
    for Every statement $S_j$ in document D, where $j = 1$ to $N$ do
        if $Sim(S_i, S_j) > Threshold$ and $flag(S_j) == 0$ and $i \neq j$ then
            if $InitialScore(S_j) < min$ then
                $min = InitialScore(S_j)$
            end if
        end if
    end for
    if $min == 1000$ then
        $FinalScore(S_i, V_5) \leftarrow 0$
    else
        $FinalScore(S_i, V_5) \leftarrow min$
    end if
    $min \leftarrow 1000$
end for
```

CombMAX(V_6) [94] computes final score of each candidate sentence as maximum of scores of voter sentences as given in Algorithm 7. Opposite to combMIN, in CombMAX maximum value voters initial score is reported as score of the given candidate sentence. CombSUM(V_7) [94] update final score of candidate sentence as sum of scores of voters of given sentences as described in Algorithm 8. Procedure simply aggregates all the initial score of the voters of the given sentence.

Algorithm 7 Voting$_6$() : CombMAX

```
$max = -1$
**for** Every statement $S_i$ in document D, where $i = 1$ to $N$ **do**
  **for** Every statement $S_j$ in document D, where $j = 1$ to $N$ **do**
    **if** $Sim(S_i, S_j) > Threshold$ and $flag(S_j) == 0$ and $i \neq j$ **then**
      **if** $InitialScore(S_j) > max$ **then**
        $max = InitialScore(S_j)$
      **end if**
    **end if**
  **end for**
  **if** $max == -1$ **then**
    $FinalScore(S_i) \leftarrow 0$
  **else**
    $FinalScore(S_i) \leftarrow max$
  **end if**
  $max = -1$
**end for**
```

Algorithm 8 Voting$_7$() : CombSUM

```
**for** Every statement $S_i$ in document D, where $i = 1$ to $N$ **do**
  $FinalScore(S_i, V_7) \leftarrow 0$
  **for** Every statement $S_j$ in document D, where $j = 1$ to $N$ **do**
    **if** $Sim(S_i, S_j) > Threshold$ and $flag(S_j) == 0$ and $i \neq j$ **then**
      $FinalScore(S_i, V_7) \leftarrow FinalScore(S_i, V_7) + InitialScore(S_j)$
    **end if**
  **end for**
**end for**
```

CombANZ(V_8) [94] computes final score of each candidate sentence as fraction of CombSUM divided by number of voter sentences as described in Algorithm 9. Similar to previously described voting algorithms, after initialization and similarity check, CombSUM score of each sentence is computed. Thereafter, CombSUM is divided by total number of voters(Votes) to get the final CombANZ score of the given candidate sentence. CombMNZ (V_9) [94] scores each candidate sentence as sum of multiplication of CombSUM and number of voter sentences as given in Algorithm 10.

Algorithm 9 Voting$_8$() : CombANZ

```
for Every statement $S_i$ in document D, where $i = 1$ to $N$ do
  $FinalScore(S_i, V_8) \leftarrow FinalScore(S_i, V_7)/FinalScore(S_i, V_1)$
end for
```

Algorithm 10 Voting$_9$() : CombMNZ

```
for Every statement $S_i$ in document D, where $i = 1$ to $N$ do
  $FinalScore(S_i, V_9) \leftarrow FinalScore(S_i, V_7)*FinalScore(S_i, V_1)$
end for
```

Final score for expCombSUM(V_{10}) [95] for each candidate sentence is computed as sum of exponential scores of its voter sentences as given in Algorithm 11. Exponnential of initial scores for each voter sentence the given candidate sentence is computed to get the final score. Score using expCombANZ(V_{11}) [95] for each candidate sentence is computed as fraction of expCombSUM divided by number of its voter sentences as described in Algorithm 12. Final score value for each candidate set using expCombMNZ(V_{12}) [95] is computed as multiplication of expCombSUM divided by number of its voter sentences as described in Algorithm 13. All voting models described above score each of the sentences present in the document.

Algorithm 11 Voting$_{10}$() : expCombSUM

```
for Every statement $S_i$ in document D, where $i = 1$ to $N$ do
  $FinalScore(S_i, V_{10}) \leftarrow 0$
  for Every statement $S_j$ in document D, where $j = 1$ to $N$ do
    if $Sim(S_i, S_j) > Threshold$ and $flag(S_j) == 0$ and $i \neq j$ then
      $FinalScore(S_i, V_{10}) \leftarrow FinalScore(S_i, V_{10}) + exp(InitialScore(S_j))$
    end if
  end for
end for
```

Algorithm 12 Voting$_{11}$: expCombANZ

```
for Every statement $S_i$ in document D, where $i = 1$ to $N$ do
  $FinalScore(S_i, V_{11}) \leftarrow FinalScore(S_i, V_{10})/FinalScore(S_i, V_1)$
end for
```

Algorithm 13 Voting$_{12}$: expCombMNZ

```
for Every statement $S_i$ in document D, where $i = 1$ to $N$ do
  $FinalScore(S_i, V_{12}) \leftarrow FinalScore(S_i, V_{10})*FinalScore(S_i, V_1)$
end for
```

6.3 Proposed Summarization Process

For document retrieval task user query is used as topic for initial retrieval of documents. Thereafter, initial ranking is done using specific criteria. For each document, set of voters are identified using specified similarity measure. Finally, document ranks are calculated for final ranking of the given set of documents. For text summarization task, linear sum of a set of feature algorithms is computed for initial ranking process. Thereafter, Jaccard similarity measure is used to find the set of voters. Then for each of the voting model sentence score is calculated. The process flow of proposed voting based approach is shown in Figure 6.1.

As shown in the Figure 6.1, input document is pre-processed first. Thereafter, initial ranking process is applied using single features algorithms. Voting models are then applied to get the final scores of sentences. Along with final sentence scoring and ranking process voting and sentence filtering process using a specific feature algorithm combination as described earlier in Chapter 5 is applied. The procedure of computation of final score of a given sentence using various voting models is described in Algorithm 14. Firstly, all the constants and variables are initialized. Initial score of sentence is calculated by sum of inverse of all the feature algorithms scores for that sentence. After calculating initial scores for each sentence of the document, the initial rank of each sentence based on their initial scores is calculated as describes in Algorithm 15. This initial rank will be used in voting schemes to calculate the final scores for the sentences. Now to calculate the final score for each sentence, any one of the voting algorithm is applied. Final score for each sentence using corresponding voting algorithm, V_i is calculated first. After computing final scores, final rank is assigned to each sentence based on these final

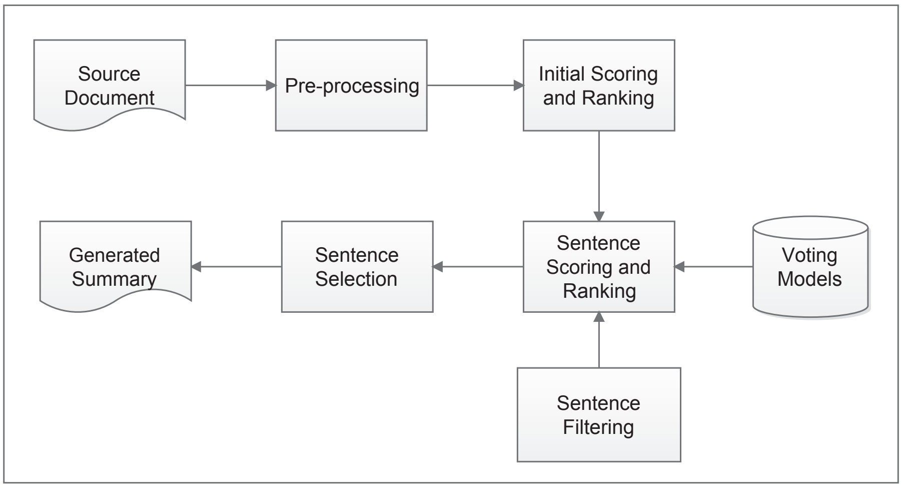


Figure 6.1: Voting based summarization process flow using sentence filtering.

scores using Algorithm 15. After that, all the sentences of document are sorted in descending order according to their final ranks. Now first N_s sentences from the sorted list are included into the summary. Thus generating the summary of N_s sentences using the voting scheme V_i.

Algorithm 14 Voting Algorithm

```
INITIALIZATION:
$N$: Total number of sentences in text document.
$N_{rem}$: Total sentence required after sentence filtering.
$N_s$: Total number of sentences required in the final summary.
$M$: Total number of feature algorithms considered for initial ranking.
$D = \{S_1, S_2, \cdots, S_N\}$    // Text document containing sequence of $N$ sentences.
$V = \{V_1, V_2, \cdots, V_{12}\}$    // Voting Algorithms.
$Sim(S_i, S_j)$: Similarity between sentence $S_i$ and sentence $S_j$.
Threshold: Threshold used for similarity check between two sentences.
$Score_m(S_i)$: Score of sentence $S_i$ using $m^{th}$ Feature Algorithm.
$CombScore(S_i)$: Feature Combination Score of sentence $S_i$.
$InitialScore(S_i)$: Initial Score of sentence $S_i$.
$FinalScore(S_i, V_j)$: Final Score of sentence $S_i$ for respective Voting $V_j$.
for Every statement $S_i$ in document D, where $i = 1$ to $N$ do
    $InitialScore(S_i) = \frac{1}{Score_1(S_i)} + \frac{1}{Score_2(S_i)} + \cdots + \frac{1}{Score_m(S_i)}$
end for
$InitialRank(D) \leftarrow Rank(InitialScore(D))$ // Rank of sentence $S_i$ according to initial scores
for Every statement $S_i$ in document D, where $i = 1$ to $N$ do
    $flag(S_i) \leftarrow 0$
end for
$r \leftarrow N$
while $r \geq N_{rem}$ do
    $min \leftarrow 999$
    $index \leftarrow -1$
    for Every statement $S_k$ in document D, where $k = 1$ to $N$ do
        if $flag(S_k) = 0$ and $min > CombScore(S_k)$ then
            $min \leftarrow CombScore(S_k)$
            $index \leftarrow k$
        end if
    end for
    $flag(S_{index}) \leftarrow 1$
    $r \leftarrow r - 1$
end while
for Every Voting $j$, where $j = 1$ to 12 do
    call Voting$_j$()
    FinalRank$(D, V_j) \leftarrow Rank(FinalScore(D, V_j))$
end for
```

Algorithm 15 Rank(InitialScore(D)) Sentence Ranking Algorithm

```
for Every statement $S_i$ in document D, where $i = 1$ to $N$ do
  $MaxScore \leftarrow -999$
  for Every statement $S_j$ in document D, where $j = 1$ to $N$ do
    if $InitialScore(S_j) > MaxScore$ then
      $MaxScore \leftarrow InitialScore(S_j)$
      $Temp \leftarrow j$
    end if
  end for
  $InitialRank(Temp) \leftarrow i$
  $InitialScore(Temp) \leftarrow -999$
end for
```

6.4 An Example: Voting

For the given text document, in Subsection 2.2.1 of Chapter 2, first the initial score of each sentence is calculated by adding reciprocal of score of corresponding sentence using the individual features: Word Co-occurrence, TextRank and Interaction between Sentences. After calculating initial scores, each sentence is assigned an initial rank based on its score. Initial score and initial rank of each sentence is shown in Table 1.1.

Table 6.2: Initial score and rank of each sentence.

Sentence No.	Initial Score	Initial Rank
S_1	3.3955	7
S_2	8.7192	4
S_3	15.6923	2
S_4	3.3589	8
S_5	3.4810	6
S_6	15.7477	1
S_7	3.5238	5
S_8	15.6923	3

Now, to calculate the voters for each sentence, similarity between each sentence pair is calculated based on term overlap. In the above example, there are only two sentence pairs that has term overlap: (S_1, S_7) and (S_4, S_5). So, the sentences in each pair has similarity with the another sentence in that pair and act as voter for that sentence. After calculating voters for each sentence, corresponding voting scheme is applied to calculate the final score of each sentence. Final score calculation for sentence S_1 using each voting is as follows:

$$\begin{aligned}
Score(S_1, V_1) &= \text{No. of voters for } S_1 \\
&= 1 && (6.1) \\
Score(S_1, V_2) &= \frac{1}{InitialRank(S_7)} \\
&= \frac{1}{5} = 0.2 && (6.2) \\
Score(S_1, V_3) &= \text{No. of Sentence} - InitialRank(S_7) \\
&= 8 - 5 = 3 && (6.3) \\
Score(S_1, V_4) &= \text{Median of initial scores of all voters of } S_1 \\
&= InitialScore(S_7) = 3.5238 && (6.4) \\
Score(S_1, V_5) &= \text{Minimum of initial scores of all voters of } S_1 \\
&= InitialScore(S_7) = 3.5238 && (6.5) \\
Score(S_1, V_6) &= \text{Maximum of initial scores of all voters of } S_1 \\
&= InitialScore(S_7) = 3.5238 && (6.6) \\
Score(S_1, V_7) &= \text{Sum of initial scores of all voters of } S_1 \\
&= InitialScore(S_7) = 3.5238 && (6.7) \\
Score(S_1, V_8) &= \frac{\text{Sum of initial scores of all voters of } S_1}{\text{No. of voters for } S_1} \\
&= \frac{InitialScore(S_7)}{1} = 3.5238 && (6.8) \\
Score(S_1, V_9) &= \text{Sum of initial scores of all voters of } S_1 \times \text{No. of voters for } S_1 \\
&= InitialScore(S_7) \times 1 = 3.5238 && (6.9) \\
Score(S_1, V_{10}) &= \text{Sum of exponential of initial scores of all voters of } S_1 \\
&= exp(InitialScore(S_7)) = exp(3.5238) = 33.9133 && (6.10) \\
Score(S_1, V_{11}) &= \frac{\text{Sum of exponential of initial scores of all voters of } S_1}{\text{No. of voters for } S_1} \\
&= \frac{exp(InitialScore(S_7))}{1} = 33.9133 && (6.11) \\
Score(S_1, V_{12}) &= \text{Sum of exponential of initial scores of all voters of } S_1 \times \text{No. of voters for } S_1 \\
&= exp(InitialScore(S_7)) \times 1 = 33.9133 && (6.12) \\
Score(S_1, V_{13}) &= \frac{1}{InitialRank(S_7)} + \frac{1}{\text{Location of } S_1} \\
&= \frac{1}{5} + \frac{1}{1} = 0.2 + 1 = 1.2 && (6.13)
\end{aligned}$$

6.5 Experimental Setup and Evaluation Measures

For experiments Kumar et al. [98] voting process for text summarization is implemented. Thereafter, proposed voting process with sentence filtering is implemented. Feature algorithm combinations are used for sentence filtering process whic are reported in Chapter 4. Here also similar to the Chapter 4, the DUC 2002 dataset is divided into three groups. In total we have four group of documents namely, short, medium, large and all documents. As the proposed voting process requires initial ranking also, different combinations are experimented for the same. ROUGE evaluation measure as described in Section 4.5 of Chapter 4 are used for performance testing. Each of the ROUGE-1, ROUGE-2, ROUGE-L and ROUGE-W scores are computed for performance evaluation for each of the text documents.

6.6 Results

Results of Kumar et al. [98] and proposed work are discussed in the following section. Best five voting models for proposed approach are selected for analysis. These voting models as proposed consist of a filtering combination, intial ranking combination and a specific voting model.

6.6.1 ROUGE scores of Voting Models proposed earlier

Results of voting models proposed by Kumar et al. [98] using ROUGE-1 and ROUGE-2 are presented in Table 6.3. It is observed that the f-measure values of all voting models is in the similar range as like feature algorithms. Similar observation is made for the results of Kumar et al. [98] work using ROUGE-L and ROUGE-W as shown in Table 6.3. This observation primarily set a belief that it is not adequate to use voting models only for text summarization process. Therefore, additional heuristics are required to enhance the performance of text summaries.

Best initial ranking combinations without sentence filtering are shown in Table 6.5. Here all the possible combinations of features are used to find the best out of them. ROUGE-1 and ROUGE-2 scores of these best five voting methods without sentence filtering named as WBVM are presented in Table 6.6. It is observed

Table 6.3: ROUGE-1 and ROUGE-2 scores of voting models proposed by Kumar et al. in 2014.

S. No.	**Method**	**ROUGE-1**			**ROUGE-2**		
		Precision	Recall	F-Measure	Precision	Recall	F-Measure
1	Voting1	0.45401	0.48460	0.46694	0.18830	0.20149	0.19388
2	Voting2	0.42894	0.45579	0.44007	0.15793	0.16862	0.16236
3	Voting3	0.44697	0.47599	0.45908	0.18163	0.19379	0.18671
4	Voting4	0.41904	0.44731	0.43093	0.15043	0.16155	0.15509
5	Voting5	0.39332	0.42182	0.40540	0.13321	0.14417	0.13783
6	Voting6	0.44278	0.47269	0.45549	0.17478	0.18710	0.17999
7	Voting7	0.44562	0.47344	0.45745	0.17863	0.19036	0.18363
8	Voting8	0.40101	0.42891	0.41288	0.13502	0.14538	0.13943
9	Voting9	0.45020	0.47798	0.46195	0.18399	0.19560	0.18890
10	Voting10	0.43052	0.46122	0.44343	0.16054	0.17284	0.16570
11	Voting11	0.40314	0.43026	0.41482	0.13594	0.14605	0.14031
12	Voting12	0.44007	0.46987	0.45275	0.17097	0.18314	0.17615

Table 6.4: ROUGE-L and ROUGE-W scores of voting models proposed by Kumar et al. in 2014.

S. No.	**Method**	**ROUGE-L**			**ROUGE-W**		
		Precision	Recall	F-Measure	Precision	Recall	F-Measure
1	Voting1	0.45401	0.48460	0.46694	0.18830	0.20149	0.19388
2	Voting2	0.42894	0.45579	0.44007	0.15793	0.16862	0.16236
3	Voting3	0.44697	0.47599	0.45908	0.18163	0.19379	0.18671
4	Voting4	0.41904	0.44731	0.43093	0.15043	0.16155	0.15509
5	Voting5	0.39332	0.42182	0.40540	0.13321	0.14417	0.13783
6	Voting6	0.44278	0.47269	0.45549	0.17478	0.18710	0.17999
7	Voting7	0.44562	0.47344	0.45745	0.17863	0.19036	0.18363
8	Voting8	0.40101	0.42891	0.41288	0.13502	0.14538	0.13943
9	Voting9	0.45020	0.47798	0.46195	0.18399	0.19560	0.18890
10	Voting10	0.43052	0.46122	0.44343	0.16054	0.17284	0.16570
11	Voting11	0.40314	0.43026	0.41482	0.13594	0.14605	0.14031
12	Voting12	0.44007	0.46987	0.45275	0.17097	0.18314	0.17615

that ROUGE scores are better as compared to the method proposed by Kumar et al [98]. Same observation is made for results of ROUGE-L and ROUGE-W on the same set of all documents and is reported in Table 6.7. Sentence filtering as described in the previous Chapter 5 is used a heuristic to further improve the quality of text summaries.

Table 6.5: Details of best five voting methods without sentence filtering.

S. No.	Short Name	Initial Ranking Features	Voting Model
1	WBVM1	FA25, FA10, FA21, FA 12, FA2, FA8, FA19	Voting6
2	WBVM2	FA25, FA10, FA21, FA 12, FA2, FA8, FA19	Voting8
3	WBVM3	FA25, FA10, FA21, FA 12, FA2, FA8, FA19	Voting10
4	WBVM4	FA25, FA10, FA21, FA 12, FA2, FA8, FA19	Voting12
5	WBVM5	FA25, FA10, FA21, FA 12, FA2, FA8, FA19	Voting11

Table 6.6: ROUGE-1 and ROUGE-2 scores of best voting methods without sentence filtering for all documents.

S. No.	Method	ROUGE-1			ROUGE-2		
		Precision	Recall	F-Measure	Precision	Recall	F-Measure
1	WBVM1	0.47026	0.49514	0.48046	0.20861	0.22055	0.21350
2	WBVM2	0.47012	0.49501	0.48032	0.20849	0.22044	0.21339
3	WBVM3	0.47003	0.49506	0.48029	0.20840	0.22043	0.21334
4	WBVM4	0.47003	0.49506	0.48029	0.20840	0.22043	0.21334
5	WBVM5	0.47004	0.49501	0.48028	0.20837	0.22035	0.21328

Table 6.7: ROUGE-L and ROUGE-W scores of best voting methods without sentence filtering for all documents.

S. No.	Method	ROUGE-L			ROUGE-W		
		Precision	Recall	F-Measure	Precision	Recall	F-Measure
1	WBVM1	0.39139	0.41215	0.38287	0.39515	0.41616	0.38653
2	WBVM2	0.39123	0.41201	0.38271	0.39500	0.41602	0.38638
3	WBVM3	0.39119	0.41209	0.38261	0.39494	0.41608	0.38626
4	WBVM4	0.39119	0.41209	0.38261	0.39494	0.41608	0.38626
5	WBVM5	0.39114	0.41198	0.38259	0.39489	0.41598	0.38624

6.6.2 Best five voting methods

From the different sets of initial ranking feature set, sentence filtering combinations and voting models for a specific voting method, best five methods are selected. These best methods are used for further analysis and comparisons. These best Five methods are shown in Table 6.8. Names of best voting methods are shortened as BVM.

Average ROUGE-1 and ROUGE-2 scores of best five voting methods on whole documents set are presented in Table 6.9. Results of top voting methods using ROUGE-L and ROUGE-W on same document group are reported in Table 6.10. ROUGE-1, ROUGE-2 , ROUGE-L and ROUGE-W scores of BVM1 are higher as compared to all other experimented voting methods and f-measure scores are reported as 0.48317, 0.22038, 0.40555 and 0.40184, respectively.

Table 6.8: Details of best five voting methods with sentence filtering.

S. No.	Short Name	Combinations	Initial Ranking Features	Voting Model
1	BVM1	COMB12	FA25, FA10, FA21, FA 12, FA2, FA8, FA19	Voting10
2	BVM2	COMB22	FA25, FA10, FA21, FA 12, FA2, FA8, FA19	Voting12
3	BVM3	COMB9	FA25, FA10, FA21, FA 12, FA2, FA8, FA19	Voting11
4	BVM4	COMB9	FA25, FA10, FA21, FA 12, FA2, FA8, FA19	Voting6
5	BVM5	COMB12	FA25, FA10, FA21, FA 12, FA2, FA8, FA19	Voting6

Table 6.9: ROUGE-1 and ROUGE-2 scores of best voting methods for all documents.

S. No.	Method	ROUGE-1			ROUGE-2		
		Precision	Recall	F-Measure	Precision	Recall	F-Measure
1	BVM1	0.47818	0.50781	0.49052	0.21275	0.22660	0.21849
2	BVM2	0.47864	0.50701	0.49039	0.21290	0.22628	0.21843
3	BVM3	0.47847	0.50681	0.49019	0.21246	0.22579	0.21796
4	BVM4	0.46873	0.49795	0.48094	0.19960	0.21281	0.20512
5	BVM5	0.46424	0.49521	0.47728	0.19666	0.20976	0.20214

Table 6.10: ROUGE-L and ROUGE-W scores of best voting methods for all documents.

S. No.	Method	ROUGE-L			ROUGE-W		
		Precision	Recall	F-Measure	Precision	Recall	F-Measure
1	BVM1	0.40379	0.42872	0.41417	0.39823	0.42273	0.40843
2	BVM2	0.40411	0.42798	0.41399	0.39835	0.42176	0.40804
3	BVM3	0.40391	0.42773	0.41376	0.39834	0.42175	0.40801
4	BVM4	0.39471	0.41935	0.40499	0.38943	0.41365	0.39953
5	BVM5	0.39089	0.41671	0.40175	0.38517	0.41056	0.39585

6.7 Analysis

F-Measure comparisons of best voting methods with and without sentence filtering for all document set using ROUGE-1 and ROUGE-2 are presented in Figure 6.2 and 6.3, respectively. Results are sorted in descending order of f-measure. It is observed that BVM1 performs better as compared to all other Voting methods. COMB12 is used in BVM1 for sentence filtering. COMB12 scores each sentence using linear sum of feature algorithms such as depth of the sentence in the tree, term frequency, lexrank and sentence location1. F-Measure comparisons of best voting methods for all document set using ROUGE-L is shown in Figure 6.4. Here also, BVM1 performs better as compared to all other top voting methods since sentence filtering as heuristic is used in BVM1.

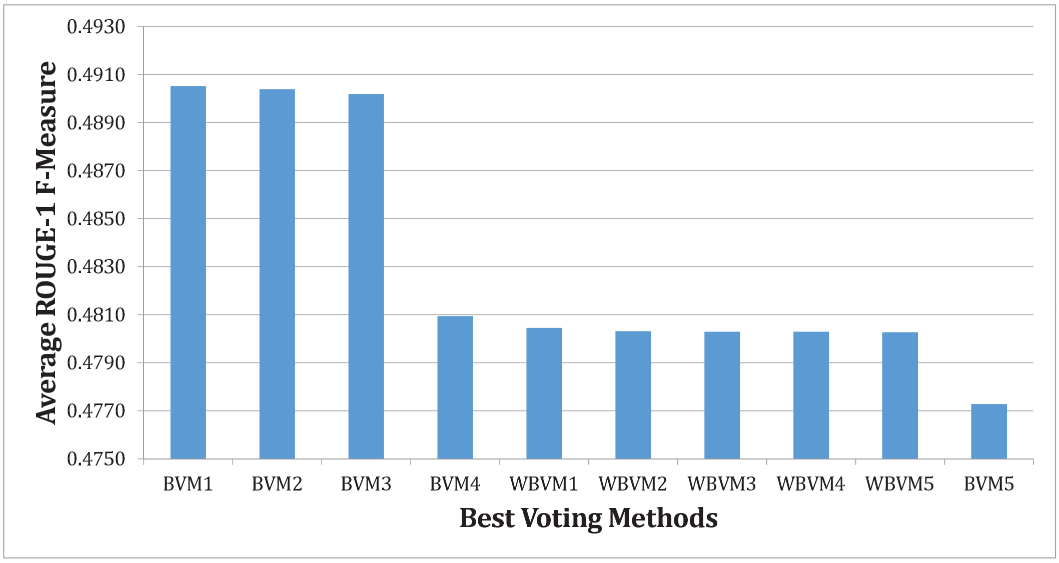


Figure 6.2: Effect of sentence filtering on voting methods using ROUGE-1 f-measure.

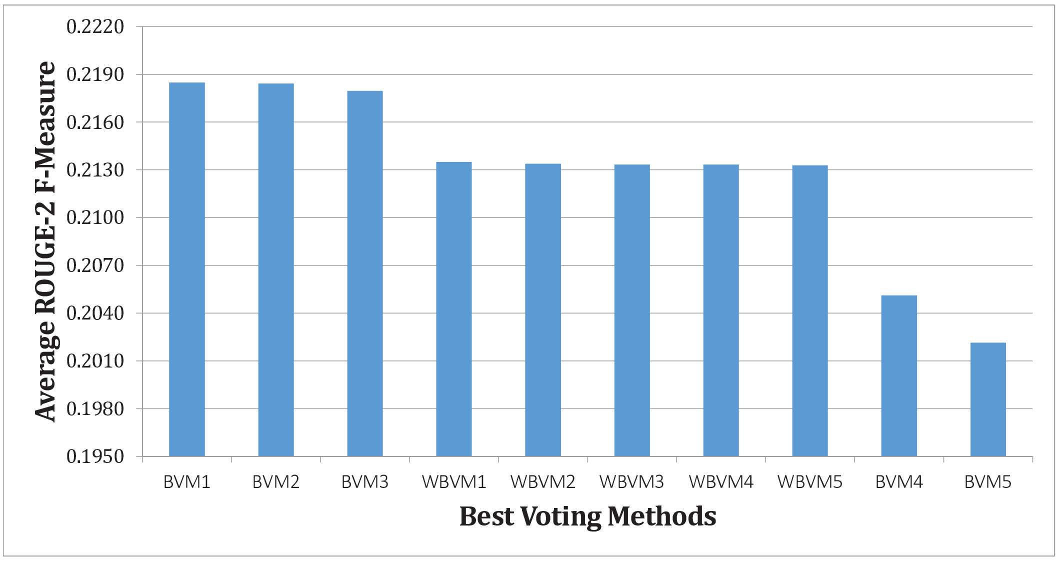


Figure 6.3: Effect of sentence filtering on voting methods using ROUGE-2 f-measure.

6.8 Comparison with Existing Approaches

Comparison in between best voting method (BVM1) and existing feature algorithms such as bushy path(FA1) [24], depth of the sentence in the tree(FA4) [84], sentence to sentence cohesion(FA17) [84], title similarity(FA23) [10] and sentence location1(FA15) [9] using ROUGE-1 and ROUGE-2 are shown in Figure 6.5 and 6.6, respectively.

Proposed voting method BVM1 performs better as compared to all other feature

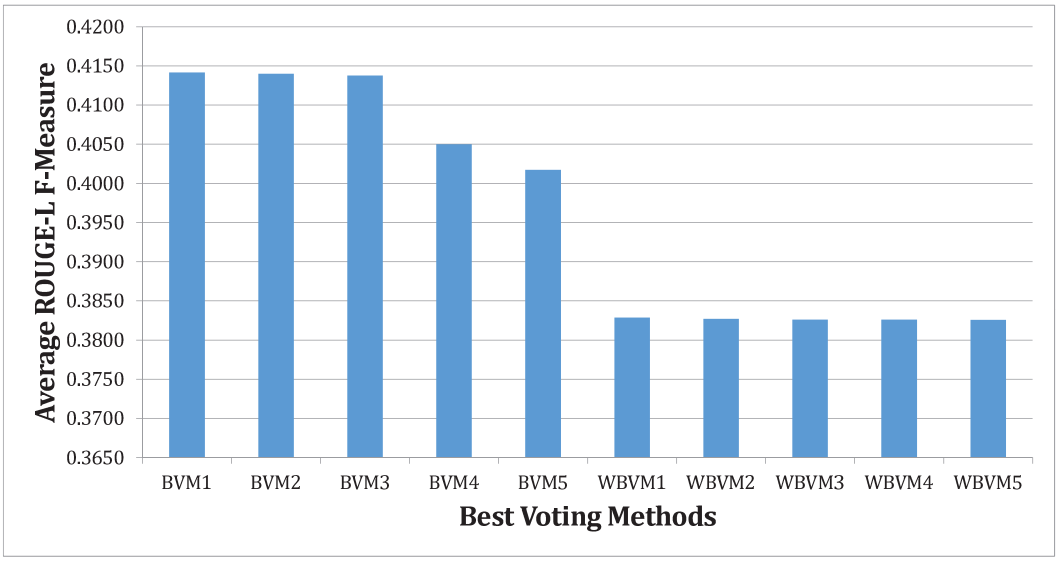


Figure 6.4: Effect of sentence filtering on voting methods using ROUGE-L f-measure.

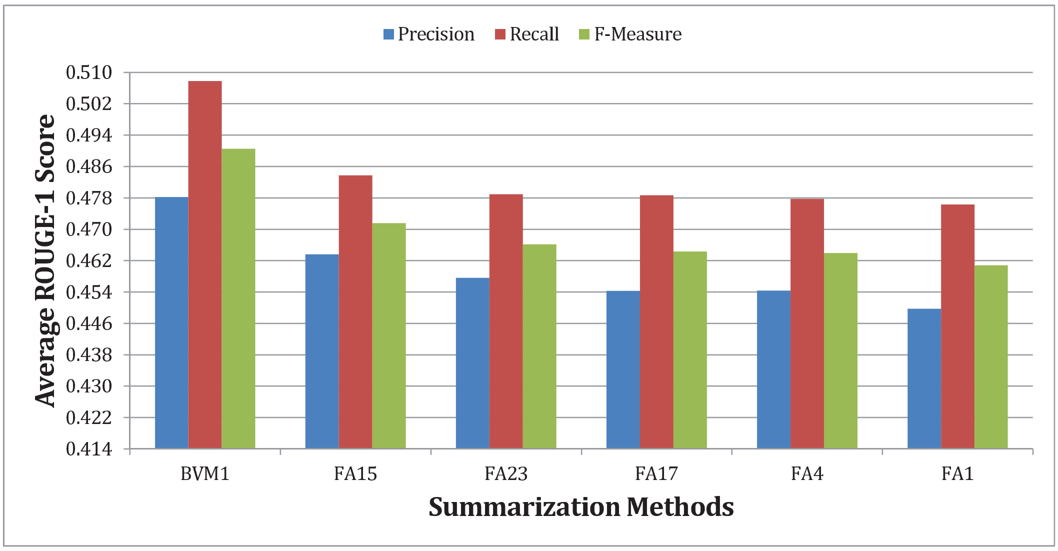


Figure 6.5: Comparison of existing best feature algorithms with BVM1 using ROUGE-1.

algorithms. Comparison in between best voting method (BVM1) and existing feature algorithms using ROUGE-L and ROUGE-W is shown in Figure 6.7 and 6.8, respectively. Here also it can be easily observed that BVM1 performs better as compared to bushy path(FA1) [24], depth of the sentence in the tree(FA4) [84], sentence to sentence cohesion(FA17) [84], title similarity(FA23) [10] and sentence location1(FA15) [9].

Feature algorithm combinations improves quality of summary using strengths of individual algorithms. Further, using combination as filtering along with voting

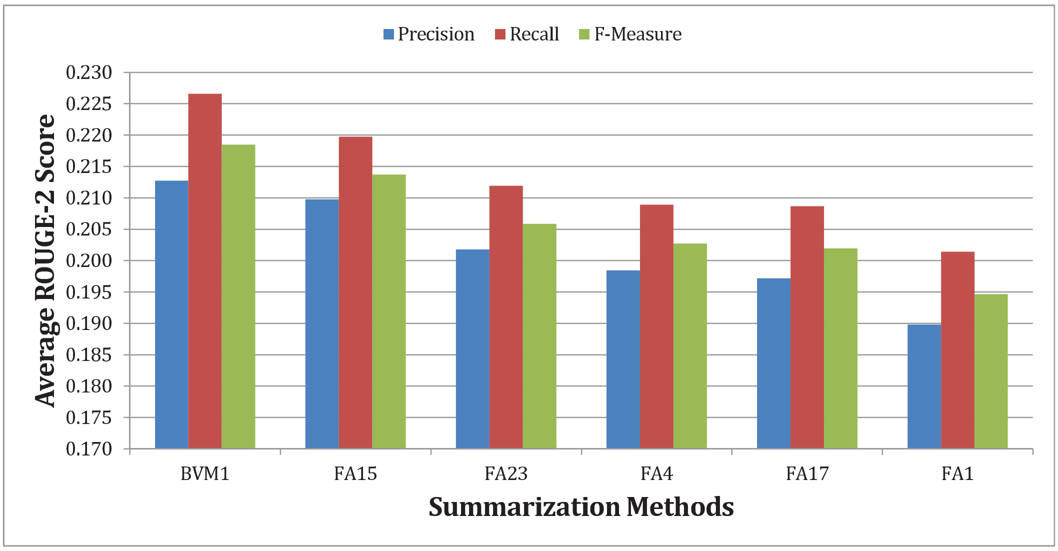


Figure 6.6: Comparison of existing best feature algorithms with BVM1 using ROUGE-2.

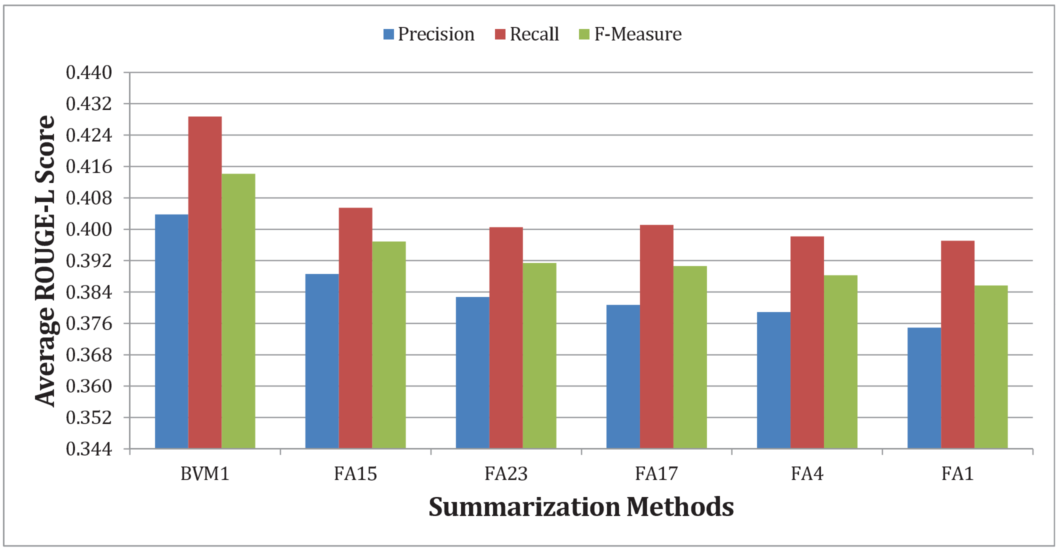


Figure 6.7: Comparison of existing best feature algorithms with BVM1 using ROUGE-L.

models improves the results. Voting use supports of each sentences from each of the other sentence in the given text. Sentences having higher support values are assigned higher scores.

Comparison in between best voting method (BVM1), best combination(COMB1), professional tool(MS Word and Copernic), LexRank, Cortex and LSA using ROUGE-1 and ROUGE-2 are shown in Figure 6.9 and 6.10, respectively.

Proposed voting method (BVM1) performs better as compared to all other approaches. Here also recall value of Copernic is high as compared to BVM1. This

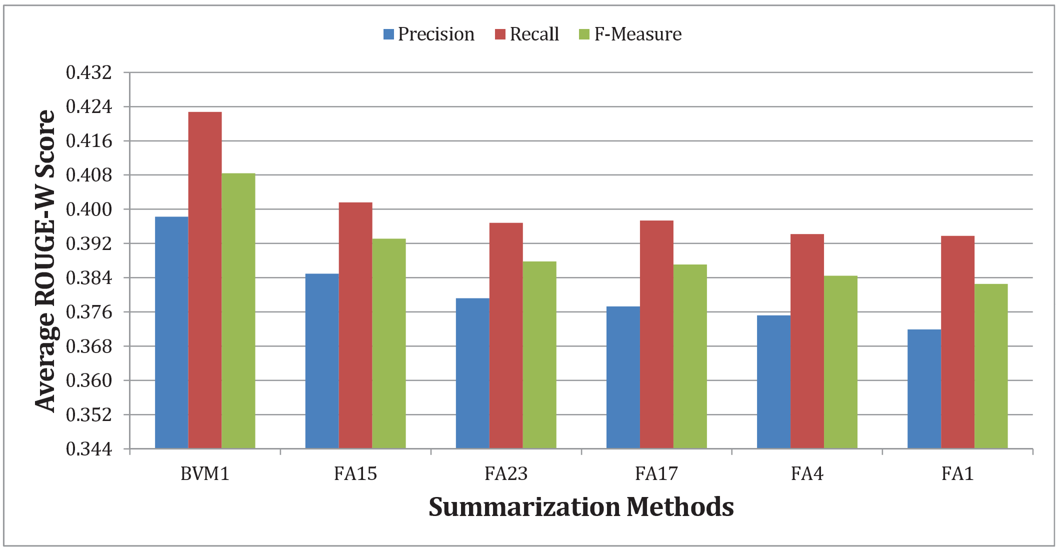


Figure 6.8: Comparison of existing best feature algorithms with BVM1 using ROUGE-W.

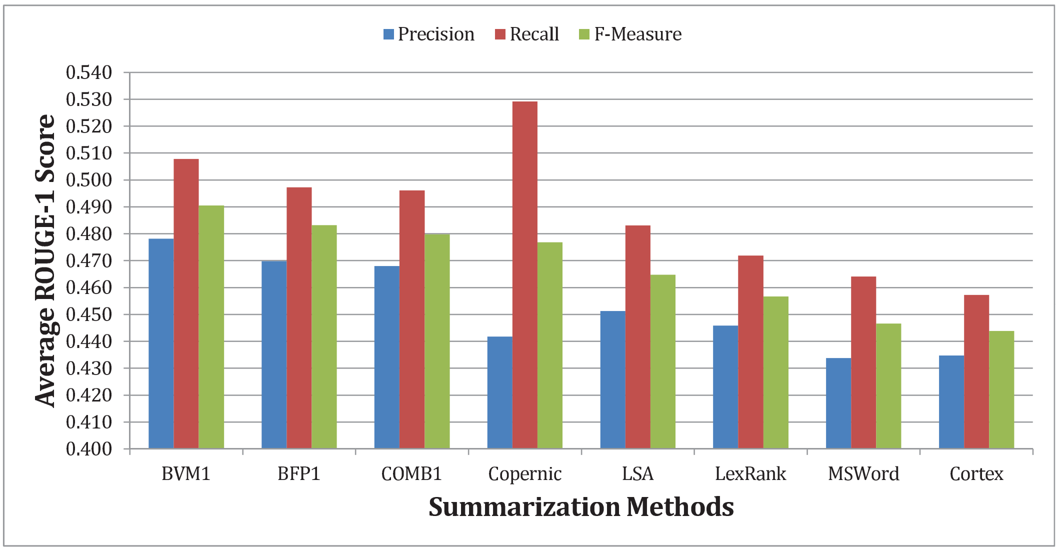


Figure 6.9: Comparison of existing approaches with BVM1 using ROUGE-1.

is due the the highers compression rate of the given text documents. More number of words in generated summary finally lowers the precision value and increases the value of recall. However, f-measure value of BVM1 is high as compared to Copernic. Comparison in between best voting method (BVM1), best combination(COMB1), professional tools(MSWord and Copernic), LexRank, Cortex and LSA using ROUGE-L and ROUGE-W are shown in Figure 6.11 and 6.12, respectively.

ROUGE-W f-measure of Copernic is higher as compared to BFP1. However, f-measure value is lower as compared to BVM1. Finally by comparing existing

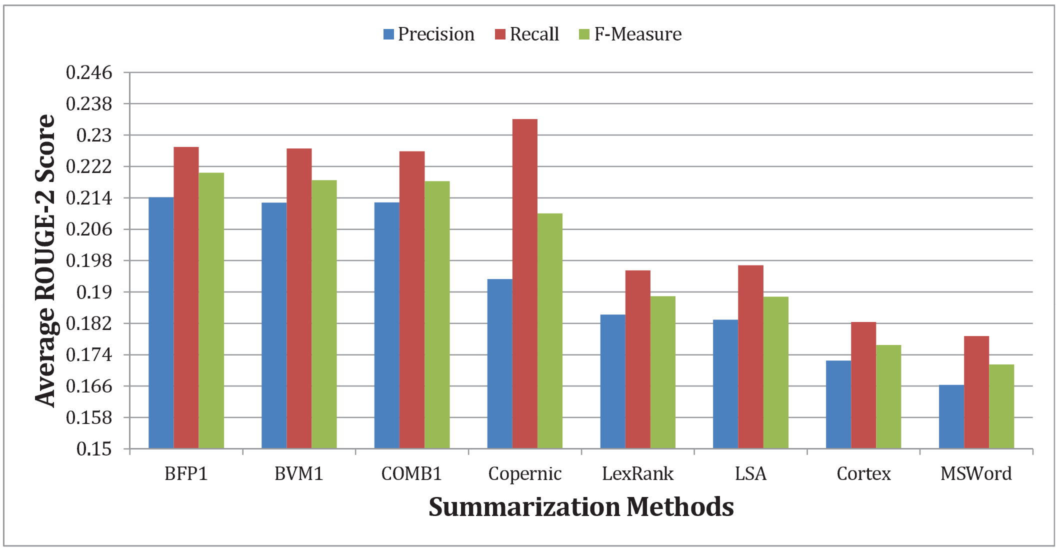


Figure 6.10: Comparison of existing approaches with BVM1 using ROUGE-2.

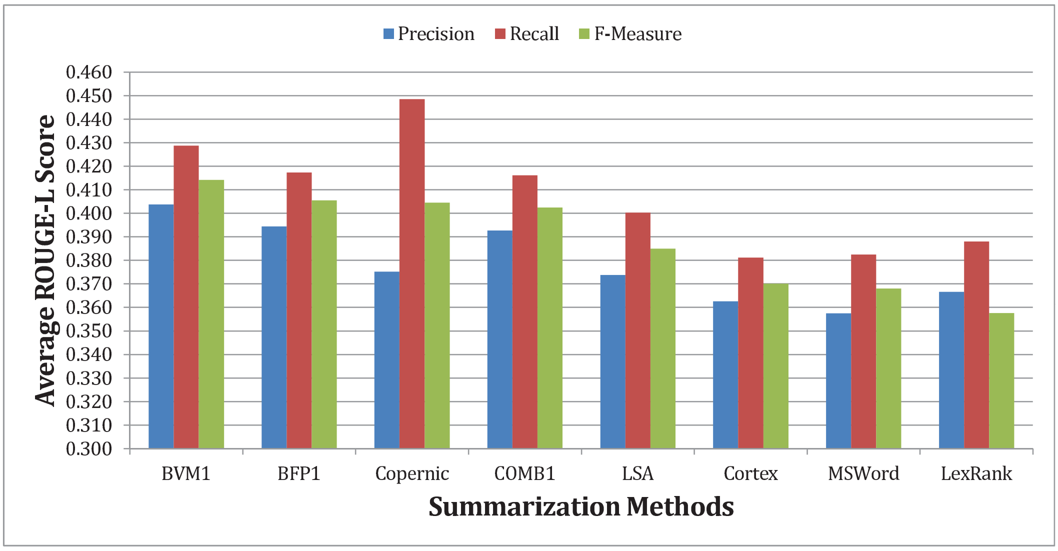


Figure 6.11: Comparison of existing approaches with BVM1 using ROUGE-L.

methods with best voting method (BVM1), it has been observed that the performance of BVM1 is better as compared to all other existing statistical and graph based methods. BVM1 use best feature combinations for sentence filtering, specific feature set for modified initial ranking and specific voting scheme. Professional tools also performs worse as compared to BVM1.

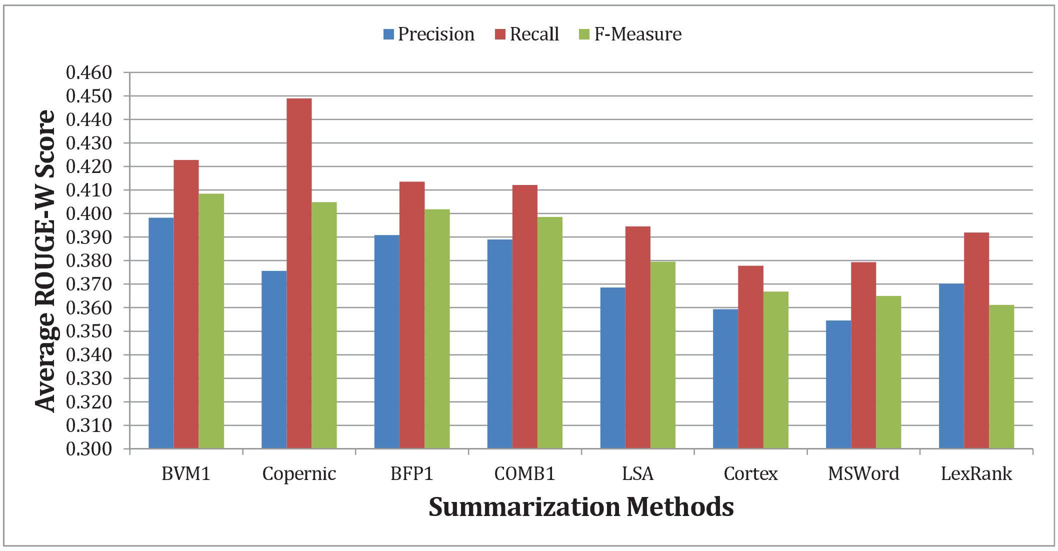


Figure 6.12: Comparison of existing approaches with BVM1 using ROUGE-W.

6.9 Summary

In this Chapter voting models with sentence filtering and modified initial ranking are used for generating text summaries. Voting models selects sentences that are of higher strength as compared to other sentences in the given text on the basis of inter relatedness of sentences. Experimental results using ROUGE evaluation measures shown that the use of sentence filtering as heuristic with voting models improves the performance of generated text summaries.

Chapter 7

Conclusions and Future Directions

In this chapter, conclusions drawn from the research work along with the possible directions for future work are presented. The major goal of this research is to present a better sentence scoring system for extractive automatic text summarization using statistical and graph based features. To achieve this objective, best feature combinations are identified using ROUGE evaluation matrices. In the presented work, voting and sentence filtering techniques are used for sentence scoring with best feature combinations. A text summarization system is sensitive to the sentence scoring methods employed to score the sentences. This thesis reports a novel extractive automatic text summarization method using filtering and voting techniques. Experimental results using ROUGE evaluation measures shown that proposed method performs better as compared to single feature algorithms, other state of the art statistical methods and professional tools such as MS Word and Copernic. The key findings of this research work are summarized as follows:

- Single feature algorithms score sentences using a specific criterion, therefore, do not improve the quality of text summaries significantly. Feature combinations also select sentences according to specific sentence weighing. A single combination can not be justified as best; instead a set of features combinations can be termed as best feature combinations. Interestingly this set of best feature combinations contains features like bushy path, cosine similarity with title, depth of the sentence in the tree, lexrank, aggregate similarity, sentence location1, sentence to sentence cohesion and title similarity. Analysis, performed on the DUC 2002 dataset demonstrates that stemming improves ROUGE scores significantly, and degrades as the size of given text document increases.

- Best feature combinations are used for sentence filtering as filtering permutation sequences. Specific filtering permutation COMB12 → COMB6 → COMB18 → COMB1 improves the quality of text summaries using strengths of feature combinations and multiple levels of sentence filtering. COMB12 scores each sentence using linear sum of feature algorithms such as depth of the sentence in the tree [84], term frequency [8], lexrank [25] and sentence location1 [9]. A specific number of sentences are filtered out by using this feature algorithm combination. Experimental results demonstrated that best feature permutation (BFP1) performs even better as compared to other state of the art methods.

- Sentence filtering is further integrated with voting to support information rich sentences. Voting models selects sentences with higher strength on the basis of the interrelatedness of sentences. By applying sentence filtering as heuristic with voting models further improves the performance of generated text summaries. Best voting method (BVM1) performs better as compared to other state of the art methods for text summarization. Feature combination COMB12 is used in BVM1 for sentence filtering.

In continuation to the above contributions, some of the future research plans are as follows:

- Pragmatic analysis may further improve the quality of text summaries. For the purpose, certain additional features may be found to extract the information rich sentences.

- In the presented voting based sentence filtering, sentence rejection starts after computing the score of each sentence for all single features and feature combinations. In future, the score of sentences may be calculated on the remaining sentences after each successive level of sentence filtering. This scoring method after sentence filtering might further improve the performance.

- For resolving the issue of anaphora, efficient post-processing may be applied to the generated summary.

List of Contributions

A. Journal Publications

[J-1] Yogesh Kumar Meena and Dinesh Gopalani, "Efficient Voting Based Extractive Automatic Text Summarization Using Prominent Feature Set", **Communicated (SCIE)** to IETE Journal of Research.

[J-2] Yogesh Kumar Meena and Dinesh Gopalani, "Improvement in Quality of Text Summaries using Feature Priority Based Sentence Filtering", **Communicated (SCIE)** to ELEKTRONIKA IR ELEKTROTECHNIKA.

[J-3] Yogesh Kumar Meena and Dinesh Gopalani, "Optimal Cluster Priority Based Sentence Ranking for Efficient Extractive Text Summaries", Electrical & Computer Engineering: An International Journal (ECIJ), Wireilla, Volume 4:1, 36-43, 2015.

B. Conference Publications

[C-1] Yogesh Kumar Meena and Dinesh Gopalani, "Improvement in Quality of Extractive Text Summaries using Modified Reciprocal Ranking", in Proceedings of International Conference on Information and Communication Technology for Sustainable Development (ICT4SD), Ahmadabad, India, July 3-4, 2015.

[C-2] Yogesh Kumar Meena, Peeyush Deoliya and Dinesh Gopalani, "Optimal Features Set For Extractive Automatic Text Summarization", in Proceedings of 5th IEEE International Conference on Advanced Computing & Communication, Rohtak, India, 21-22 February, 2015.

[C-3] Yogesh Kumar Meena and Dinesh Gopalani, "Domain Independent Framework for Automatic Text Summarization", in Proceedings of International Conference on Intelligent Computing, Communication & Convergence (ICCC-2014), ELSEVIER Procedia Computer Science, Bhubaneswar, Odisha, 27-28, December 2014.

[C-4] Yogesh Kumar Meena and Dinesh Gopalani, "Feature Priority Based Sentence Filtering Method for Extractive Automatic Text Summarization",in Proceedings of International Conference on Intelligent Computing, Communication & Convergence (ICCC-2014), ELSEVIER Procedia Computer Science, Bhubaneswar, Odisha, 27-28, December 2014.

[C-5] Yogesh Kumar Meena and Dinesh Gopalani, "Analysis of Sentence Scoring Methods for Extractive Automatic Text Summarization",in Proceedings of International Conference on Information and Communication Technology for Competitive Strategies (ICTCS '14). ACM, New York, NY, USA, , Article 53 , pp. 1-6, 2014.

[C-6] Yogesh Kumar Meena, Ashish Jain and Dinesh Gopalani, "Survey on Graph and Cluster Based Approaches in Multi-document Text Summarization", in Proceedings of IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE), 2014 , vol., no., pp.1-5, 9-11 May 2014.

C. Book Chapter

[B-1] Yogesh Kumar Meena and Dinesh Gopalani, "Statistical Features for Extractive Automatic Text Summarization", **Communicated-Proposal Approved** to Enterprise Big Data Engineering, Analytics, and Management Book, IGI Global.

Bibliography

[1] Karen Sparck Jones. Automatic summarising: Factors and directions. In *Advances in Automatic Text Summarization*, pages 1–12. MIT Press, 1998.

[2] Eduard Hovy and Chin Yew Lin. Automated multilingual text summarization and its evaluation. Technical report, Technical report Information Sciences Institute, University of Southern California, 1999.

[3] Inderjeet Mani and Mark T Maybury. *Advances in automatic text summarization*, volume 293. MIT Press, 1999.

[4] Dipanjan Das and André FT Martins. A survey on automatic text summarization. *Literature Survey for the Language and Statistics II course at CMU*, 4:192–195, 2007.

[5] Vishal Gupta and Gurpreet Singh Lehal. A survey of text summarization extractive techniques. *Journal of Emerging Technologies in Web Intelligence*, 2(3):258–268, 2010.

[6] Elena Lloret and Manuel Palomar. Text summarisation in progress: A literature review. *Artif. Intell. Rev.*, 37(1):1–41, January 2012.

[7] Majid Ramezani and Mohammad-Reza Feizi-Derakhshi. Automated text summarization: An overview. *Applied Artificial Intelligence*, 28(2):178–215, 2014.

[8] Hans Peter Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165, April 1958.

[9] P. B. Baxendale. Machine-made index for technical literature: An experiment. *IBM J. Res. Dev.*, 2(4):354–361, October 1958.

[10] H. P. Edmundson. New methods in automatic extracting. *J. ACM*, 16(2):264–285, April 1969.

[11] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into texts. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 404–411, Barcelona, Spain, July 2004.

[12] Julian Kupiec, Jan Pedersen, and Francine Chen. A trainable document summarizer. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '95, pages 68–73, New York, NY, USA, 1995.

[13] Regina Barzilay and Michael Elhadad. Using lexical chains for text summarization. In *Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization*, pages 10–17, 1997.

[14] Chin Yew Lin. Rouge: a package for automatic evaluation of summaries. pages 25–26, 2004.

[15] Richard Tucker. *Automatic summarising and the CLASP system*. University of Cambridge, Computer Laboratory, 2000.

[16] Kathleen McKeown and Dragomir R Radev. Generating summaries of multiple news articles. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 74–82. ACM, 1995.

[17] Noemie Elhadad. User-sensitive text summarization thesis summary. In *Proceedings of the 19th national conference on Artifical intelligence*, pages 987–988. AAAI Press, 2004.

[18] Ronald Brandow, Karl Mitze, and Lisa F. Rau. Automatic condensation of electronic publications by sentence selection. *Inf. Process. Manage.*, 31(5):675–685, September 1995.

[19] Chin Yew Lin. Training a selection function for extraction. In *Proceedings of CIKM*, pages 55–62, 1999.

[20] Tatsunori Mori. Information gain ratio as term weight: The case of summarization of ir results. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA, 2002.

[21] Chikashi Nobata, Satoshi Sekine, Hitoshi Isahara, and Ralph Grishman. Summarization system integrated with named entity tagging and ie pattern discovery. In *Proceedings of the Third International Conference on Language*

Resources and Evaluation (LREC'02). European Language Resources Association (ELRA), 2002.

[22] Xiaoyue Liu, Jonathan J. Webster, and Chunyu Kit. An extractive text summarizer based on significant words. In *Proceedings of the 22Nd International Conference on Computer Processing of Oriental Languages. Language Technology for the Knowledge-based Economy*, ICCPOL '09, pages 168–178, Berlin, Heidelberg, 2009.

[23] Gerard Salton, Edward A. Fox, and Harry Wu. Extended boolean information retrieval. *Commun. ACM*, 26(11):1022–1036, November 1983.

[24] Gerard Salton, Amit Singhal, Mandar Mitra, and Chris Buckley. Automatic text structuring and summarization. *Inf. Process. Manage.*, 33(2):193–207, March 1997.

[25] Günes Erkan and Dragomir R. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479, December 2004.

[26] Chinatsu Aone, Mary Ellen Okurowski, and James 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*, ACL '98, pages 62–66, Stroudsburg, PA, USA, 1998.

[27] John M Conroy and Dianne P O'leary. Text summarization via hidden markov models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 406–407. ACM, 2001.

[28] Tsutomu Hirao, Yutaka Sasaki, Hideki Isozaki, and Eisaku Maeda. Ntts text summarization system for duc-2002. In *Proceedings of the Document Understanding Conference 2002*, pages 104–107. Citeseer, 2002.

[29] Krysta Marie Svore, Lucy Vanderwende, and Christopher JC Burges. Enhancing single-document summarization by combining ranknet and third-party sources. In *EMNLP-CoNLL*, pages 448–457, 2007.

[30] Chris Burges, Tal Shaked, Erin Renshaw, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *In ICML*, pages 89–96, 2005.

[31] Frank Schilder and Ravikumar Kondadadi. Fastsum: fast and accurate query-based multi-document summarization. In *In Proceedings of ACL-08: HLT*, pages 985–992, 2008.

[32] Kam-Fai Wong, Mingli Wu, and Wenjie Li. Extractive summarization using supervised and semi-supervised learning. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 985–992. Association for Computational Linguistics, 2008.

[33] John H Holland. Genetic algorithms and the optimal allocation of trials. *SIAM Journal on Computing*, 2(2):88–105, 1973.

[34] Haleh Vafaie and Kenneth De Jong. Genetic algorithms as a tool for feature selection in machine learning. In *Tools with Artificial Intelligence, 1992. TAI'92, Proceedings., Fourth International Conference on*, pages 200–203. IEEE, 1992.

[35] Carlos N Silla Jr, Gisele L Pappa, Alex A Freitas, and Celso AA Kaestner. Automatic text summarization with genetic algorithm-based attribute selection. pages 305–314, 2004.

[36] John R Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. 1992.

[37] Zhuli Xie, Xin Li, Barbara Di Eugenio, Peter C Nelson, Weimin Xiao, and Thomas M Tirpak. Using gene expression programming to construct sentence ranking functions for text summarization. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1381. Association for Computational Linguistics, 2004.

[38] Kenneth Price and Storn. *Differential evolution: a practical approach to global optimization.* Springer Science & Business Media, 2006.

[39] Hamid Khosravi. Text summarization based on genetic programming. *International Journal of Computing and ICT Research*, page 57, 2009.

[40] Oscar Cordón, Félix de Moya, and Carmen Zarco. Fuzzy logic and multiobjective evolutionary algorithms as soft computing tools for persistent query learning in text retrieval environments. In *Fuzzy Systems, 2004. Proceedings. 2004 IEEE International Conference on*, volume 1, pages 571–576. IEEE, 2004.

[41] Arman Kiani and Mohammad R Akbarzadeh. Automatic text summarization using hybrid fuzzy ga-gp. In *Fuzzy Systems, 2006 IEEE International Conference on*, pages 977–983. IEEE, 2006.

[42] Sylvain Lamprier, Tassadit Amghar, Bernard Levrat, and Frederic Saubion. Seggen: A genetic algorithm for linear text segmentation. In *IJCAI*, volume 2007, pages 1647–1652, 2007.

[43] Mohammed S Binwahlan, Naomie Salim, and Ladda Suanmali. Fuzzy swarm based text summarization. *journal of computer science*, 5(5):338, 2009.

[44] Wei Song and Choi. Fuzzy evolutionary optimization modeling and its applications to unsupervised categorization and extractive summarization. *Expert Systems with Applications*, 38(8):9112–9121, 2011.

[45] Yeni Herdiyeni Aristoteles, Ahmad Ridha, and Julio Adisantoso. Text feature weighting for summarization o text feature weighting for summarization of document f document f documents in bahasa indonesia using genetic algorithm bahasa indonesia using genetic algorithm bahasa indonesia using genetic algorithm. 2012.

[46] Albaraa Abuobieda, Naomie Salim, and Binwahlan. Differential evolution cluster-based text summarization methods. In *Computing, Electrical and Electronics Engineering (ICCEEE), 2013 International Conference on*, pages 244–248. IEEE, 2013.

[47] Martha Mendoza, Susana Bonilla, and Noguera. Extractive single-document summarization based on genetic operators and guided local search. *Expert Systems with Applications*, 41(9):4158–4169, 2014.

[48] G Hirst and D St Onge. Lexical chains as representations of context for the detection and correction of malapropisms. wordnet. c. fellbaum, 1995.

[49] Stairmand M. A computational analysis of lexical cohesion with applications in information retrieval. In *Ph.D. Dissertation, Center for Computational Linguistics UMIST, Manchester*, 1996.

[50] Meru Brunn, Yllias Chali, and Christopher J Pinchak. Text summarization using lexical chains. In *Proc. of Document Understanding Conference*. Citeseer, 2001.

[51] Joe Carthy and Michael Sherwood-Smith. Lexical chains for topic tracking. In *Systems, Man and Cybernetics, 2002 IEEE International Conference on*, volume 7, pages 5–pp. IEEE, 2002.

[52] Dan Moldovan and Adrian Novischi. Lexical chains for question answering. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002.

[53] Michel Galley and Kathleen McKeown. Improving word sense disambiguation in lexical chaining. In *IJCAI*, volume 3, pages 1486–1488, 2003.

[54] Doran William, Stokes Nicola, Carthy Joe, and Dunnion John. Comparing lexical chain based summarisation approaches using an extrinsic evaluation. In *Proceedings of the 5th International conference on Intelligent Text Processing and Computational Linguistics CICLing2004*, pages 112–117. MIT Press, 2004.

[55] Olena Medelyan. Computing lexical chains with graph clustering. In *Proceedings of the 45th Annual Meeting of the ACL: Student Research Workshop*, pages 85–90. Association for Computational Linguistics, 2007.

[56] Pierre-Etienne Genest and Guy Lapalme. Fully abstractive approach to guided summarization. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 354–358. Association for Computational Linguistics, 2012.

[57] Liang Zhou and Eduard Hovy. Templatefiltered headline summarization. *Text Summarization Branches Out: Pr ACL-04 Wkshp, July*, 2004.

[58] Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd international conference on computational linguistics*, pages 340–348. Association for Computational Linguistics, 2010.

[59] Hideki Tanaka, Akinori Kinoshita, Takeshi Kobayakawa, Tadashi Kumano, and Naoto Kato. Syntax-driven sentence revision for broadcast news summarization. In *Proceedings of the 2009 Workshop on Language Generation and Summarisation*, pages 39–47. Association for Computational Linguistics, 2009.

[60] Pierre-Etienne Genest and Guy Lapalme. Framework for abstractive summarization using text-to-text generation. In *Proceedings of the Workshop on*

Monolingual Text-To-Text Generation, pages 64–73. Association for Computational Linguistics, 2011.

[61] Ibrahim F Moawad and Mohammadreza Aref. Semantic graph reduction approach for abstractive text summarization. In *Computer Engineering & Systems (ICCES), 2012 Seventh International Conference on*, pages 132–138. IEEE, 2012.

[62] NR Kasture, Neha Yargal, Neha Nityanand Singh, Neha Kulkarni, Vijay Mathur, Chandrika Kapre, Aparna Harikumar, Rajeshwari Chandratre, Dhaval S Adhav, P Sowmiyaa, et al. A survey on methods of abstractive text summarization. *International Journal for Emerging Science and Technology*, 1(6):53–57, 2014.

[63] Neelima Bhatia and Arunima Jaiswal. Article: Trends in extractive and abstractive techniques in text summarization. *International Journal of Computer Applications*, 117(6):21–24, May 2015.

[64] Yuta Kikuchi, Tsutomu Hirao, Hiroya Takamura, Manabu Okumura, and Masaaki Nagata. Single document summarization based on nested tree structure. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 315–320, 2014.

[65] Regina Barzilay and Kathleen R McKeown. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328, 2005.

[66] Yulia Ledeneva and Grigori Sidorov. Special issue: Computational linguistics and its applications. 2010.

[67] Atefeh Farzindar and Guy Lapalme. Legal text summarization by exploration of the thematic structures and argumentative roles. In *Text Summarization Branches Out Workshop held in conjunction with ACL*, pages 27–34, 2004.

[68] Ben Hachey and Claire Grover. A rhetorical status classifier for legal text summarisation. In *Proceedings of the ACL-2004 Text Summarization Branches Out Workshop*, 2004.

[69] Simon Corston-Oliver, Eric Ringger, Michael Gamon, and Richard Campbell. Task-focused summarization of email. In *ACL-04 Workshop: Text Summarization Branches Out*, pages 43–50, 2004.

[70] Lokesh Shrestha and Kathleen McKeown. Detection of question-answer pairs in email conversations. In *Proceedings of the 20th international conference on*

Computational Linguistics, page 889. Association for Computational Linguistics, 2004.

[71] Stephen Wan and Kathy McKeown. Generating overview summaries of ongoing email thread discussions. In *Proceedings of the 20th international conference on Computational Linguistics*, page 549. Association for Computational Linguistics, 2004.

[72] Qian Diao and Jiulong Shan. A new web page summarization method. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 639–640. ACM, 2006.

[73] Jahna Otterbacher, Dragomir Radev, and Omer Kareem. News to go: hierarchical text summarization for mobile devices. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 589–596. ACM, 2006.

[74] Kathleen McKeown, Regina Barzilay, John Chen, David K Elson, David Evans, Judith L Klavans, Ani Nenkova, Barry Schiffman, and Sergey Sigelman. Columbias newsblaster: new features and future directions. 2003.

[75] Ani Nenkova, Advaith Siddharthan, and Kathleen McKeown. Automatically learning cognitive status for multi-document summarization of newswire. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 241–248. Association for Computational Linguistics, 2005.

[76] D Evans and K McKeown. Identifying similarities and differences across arabic and english news. In *Proc. of the International Conference on Intelligence Analysis. McLean, VA*, 2005.

[77] José M Perea-Ortega, Elena Lloret, L Alfonso Ureña-López, and Manuel Palomar. Application of text summarization techniques to the geographical information retrieval task. *Expert Systems with Applications*, 40(8):2966–2974, 2013.

[78] A. Abuobieda, N. Salim, A.T. Albaham, A.H. Osman, and Y.J. Kumar. Text summarization features selection method using pseudo genetic-based model. In *Information Retrieval Knowledge Management (CAMP), 2012 International Conference on*, pages 193–197, March 2012.

[79] Rajesh Shardanand Prasad, Nitish Milind Uplavikar, and Sanket Shantilals Wakhare. Feature based text summarization. In *International Journal of Advances in Computing and Information Researches*, pages 15–18, 2012.

[80] P. R. Shardanand and Uday Kulkarni. Implementation and evaluation of evolutionary connectionist approaches to automated text summarization. In *Journal of Computer Science*, pages 1366–1376, Feb 2010.

[81] P. Gupta, V.S. Pendluri, and I. Vats. Summarizing text by ranking text units according to shallow linguistic features. In *Advanced Communication Technology (ICACT), 2011 13th International Conference on*, pages 1620–1625, Feb 2011.

[82] Mohamed Abdel Fattah and Fuji Ren. Ga, mr, ffnn, pnn and gmm based models for automatic text summarization. *Computer Speech and Language*, 23(1):126 – 144, 2009.

[83] Juan-Manuel Torres-Moreno, Patricia Velázquez-Morales, and Jean-Guy Meunier. Condensés de textes par des méthodes numériques. In *JADT*, volume 2, pages 723–734. JADT, 2002.

[84] JoelLarocca Neto, AlexA. Freitas, and CelsoA.A. Kaestner. Automatic text summarization using a machine learning approach. In Guilherme Bittencourt and GeberL. Ramalho, editors, *Advances in Artificial Intelligence*, volume 2507 of *Lecture Notes in Computer Science*, pages 205–215. Springer Berlin Heidelberg, 2002.

[85] Rafael Ferreira, Luciano de Souza Cabral, Rafael Dueire Lins, Gabriel Pereira e Silva, Fred Freitas, George DC Cavalcanti, Rinaldo Lima, Steven J Simske, and Luciano Favaro. Assessing sentence scoring techniques for extractive text summarization. *Expert systems with applications*, 40(14):5755–5764, 2013.

[86] R. Ferreira, F. Freitas, L. De Souza Cabral, R. Dueire Lins, R. Lima, G. Franca, S.J. Simske, and L. Favaro. A context based text summarization system. In *Document Analysis Systems (DAS), 2014 11th IAPR International Workshop on*, pages 66–70, April 2014.

[87] J. E. Rush, R. Salvador, and A. Zamora. Automatic abstracting and indexing. ii. production of indicative abstracts by application of contextual inference and syntactic coherence criteria. *Journal of the American Society for Information Science*, 22(4):260–274, 1971.

[88] J. J. Pollock and A. Zamora. Automatic Abstracting Research at Chemical Abstracts Service. *Chemical Information and Computer Sciences*, 15(4):226–232, November 1975.

[89] Hongyan Jing. Sentence reduction for automatic text summarization. In *Proceedings of the Sixth Conference on Applied Natural Language Processing*, ANLC '00, pages 310–315, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[90] Marco Bonzanini, Miguel Martinez-Alvarez, and Thomas Roelleke. Extractive summarisation via sentence removal: Condensing relevant sentences into a short summary. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 893–896, New York, NY, USA, 2013.

[91] Craig Macdonald and Iadh Ounis. Voting techniques for expert search. *Knowledge and information systems*, 16(3):259–280, 2008.

[92] Min Zhang, Ruihua Song, Chuan Lin, Shaoping Ma, Zhe Jiang, Yijiang Jin, Yiqun Liu, Le Zhao, and S Ma. Expansion-based technologies in finding relevant and new information: Thu trec 2002: Novelty track experiments. *NIST SPECIAL PUBLICATION SP*, (251):586–590, 2003.

[93] Javed A Aslam and Mark Montague. Models for metasearch. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 276–284. ACM, 2001.

[94] Joseph A. Shaw, Edward A. Fox, Joseph A. Shaw, and Edward A. Fox. Combination of multiple searches. In *The Second Text REtrieval Conference (TREC-2*, pages 243–252, 1994.

[95] Paul Ogilvie and Jamie Callan. Combining document representations for known-item search. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 143–150. ACM, 2003.

[96] Craig Macdonald and Iadh Ounis. Voting for candidates: adapting data fusion techniques for an expert search task. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 387–396. ACM, 2006.

[97] Yogan Jaya Kumar, Naomie Salim, Albaraa Abuobieda, and Ameer Tawfik. Multi document summarization based on cross-document relation using voting technique. In *Computing, Electrical and Electronics Engineering (ICCEEE), 2013 International Conference on*, pages 609–614. IEEE, 2013.

[98] Yogan Jaya Kumar, Ong Sing Goh, Abd Ghani, Mohd Khanapi, Naomie Salim, and Ameer Taufik Albaham. Voting models for summary extraction from text documents. In *IT Convergence and Security (ICITCS), 2014 International Conference on*, pages 1–4. IEEE, 2014.