Algorithm to Resolve Anaphoric Ambiguity of Text Summarization

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Software Engineering

Submitted By
Avnish Thakur
(Roll No. 801131007)

Under the supervision of:
Mr. Ravinder Kumar
Assistant Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

July 2013

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Algorithm to Resolve Anaphoric Ambiguity of Text Summarization"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Software Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Mr. Ravinder Kumar* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Avnish Thakur

(Avnish Thakur)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Mr. Ravinder Kumar)
Assistant Professor
Computer Science and
Engineering Department
Thapar University
Patiala

Countersigned by:

(Dr. Maninder Singh)
Associate Professor and Head
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

Acknowledgement

I would like to express my deep sense of gratitude to my supervisor, **Mr. Ravinder Kumar**, Assistant Professor, Computer Science and Engineering Department, Thapar University, Patiala, for his invaluable help and guidance during the course of thesis. I am highly indebted to him for constantly encouraging me by giving his critics on my work. I am grateful to him for giving me the support and confidence that helped me a lot in carrying out the research work in the present form. And for me, it's an honor to work under him.

I also take the opportunity to thank **Dr. Maninder Singh**, Associate Professor and Head, Computer Science and Engineering Department, Thapar University, Patiala, for providing us with the adequate infrastructure in carrying the research work.

I would also like to thank my parents and friends for their inspiration and ever encouraging moral support, which went a long way in successful completion of my thesis.

Above all, I would like to thank the almighty God for His blessings and for driving me with faith, hope and courage in the thinnest of the times.

Avnish Thakur

Avnish Thakur

(801131007)

Abstract

Automatic text summarization aims to address the information overload problem due to the availability of enormous amount of information on World Wide Web, by generating summary of the documents. A summary is a text that is produced from one or more texts, that contain only the significant information contents of the original text(s), and that is no longer than half of the original text. Text summarization can be classified into two categories: abstractive and extractive summarization.

An extractive summary is simply a subset of the sentences in the original text, it generally uses statistical methods to generate summary. These summaries do not guarantee a good narrative coherence, but they can conveniently represent an approximate content of the text.

In abstractive text summarization, first the text is understood using linguistic methods and then the main concepts are expressed in natural language, using new concepts and expression to best describe the most important information present in the original text document.

In this Thesis, a survey on extractive text summarization techniques has been presented and work is done to resolve anaphoric problems in text summarization.

Table of Contents

List of Figures

List of Algorithms

List of Tables

Chapter 1
Introduction

The rapid and continuous growth of World Wide Web (WWW) has lead to the availability of enormous amount of information to the users. Internet provides far more information than is needed, in response to a query or search. It's very difficult for human beings to manually summarize large text documents, to find out relevant documents from an overwhelming number of documents available on the internet. In order to solve this problem of selecting relevant texts and extracting key information content of each text, automatic text summarizer is a very useful tool. A text summarizer is a system that produces the condensed representation of its input text.

A summary is a text that is produced from one or more texts, that contain a significant portion of information in the original text(s), and that is no longer than half of the original text(s). Text here includes multimedia documents, on-line documents, etc. The goal of automatic text summarization is to create a condensed version of the source text, preserving its information content and overall meaning. Automatic text summarization can be classified into extractive and abstractive summarization.

An extractive text summarization method [1] consists of selecting important text segments (sentences, concepts etc.) of source document and arrange them in the same order as they exist in the original text, to best describe the information content of the document in short form. The importance of segments is decided based on various statistical and linguistic features. The earliest instances of research on summarizing scientific documents proposed paradigms for extracting important sentences from text using features like word and phrase frequency (Luhn, 1958) [2], position in the text (Baxendale, 1958) [3] and key phrases (Edmundson, 1969) [4].

In abstractive text summarization, first the text is understood using linguistic methods and then the main concepts are expressed in natural language, using new concepts and expression to best describe the most important information present in the original text document.

1.1 Features for Extractive Text Summarization

Some features to be considered for assigning weight to the terms and including sentences in the summary:

- **Font based feature:**

Sentences containing words appearing in bold, italic or underlined fonts are usually important.

- **Numerical data feature:**

Sentence that contains numerical data is an important sentence and is most likely to be included in the document summary.

- **Sentence location feature:**

First sentence of the first paragraph of a text document is important and is likely to be included in the summary.

- **Sentence length feature:**

Sentences having very large or very short length are mostly not included in the summary.

- **Biased word feature:**

If a word appearing in a sentence is from biased word list, then that sentence is considered important. Biased word list generally contain domain specific words.

- **Pronouns:**

Pronouns such as "he, she, they" cannot be included in the summary unless they are expanded into their respective nouns or their corresponding nouns are included in the summary so that pronouns do not create ambiguity in the summary.

- **Stigma words:**

Some sentences starts with

  - Conjunctions (*e.g. and, if*)
  - the verb *say* and its derivatives
  - quotation marks
  - pronouns such as *he, she, they etc.*

mostly cause discontinuity and ambiguity in the summaries, hence sentences containing stigma words are demoted to improve cohesion and coupling in the summaries.

- **Similarity to title:**

Title of the document is used as a query against all sentences of the document under consideration, similarity of document's title and each sentence is computed using cosine similarity measure. Sentences containing words appearing in the title are having greater chances to be included in the summary.

- **Similarity to keyword:**

Similarity between set of keywords of the document and each sentence of the document is computed using cosine similarity. Sentences having high keyword similarity value are likely to be included in the summary.

- **Sentence-to-Sentence cohesion:**

For each sentence **s**, the similarity between **s** and every other sentence in the document is calculated, then add up these similarity values to obtain raw value of this feature for **s**. The process is repeated for all sentences. High similarity score indicates high sentence-to-sentence cohesion.

- **Sentence-to-centroid cohesion:**

First compute vector representing the centroid of the document, then compute similarity between each sentence and centroid of the document. High similarity score of sentence indicates its high sentence-to-centroid cohesion.

- **Proper Noun feature:**

Proper noun is name of a person, place etc. Sentences containing proper nouns have greater chances to be included in the summary.

- **Cue-Phrase feature:**

Sentences containing cue phrases (e.g. "purpose", "this report", "in conclusion" etc.) are important and likely to be in summaries.

- **Presence of non-essential information:**

Certain words are indicators of non-essential information. These words are speech makers such as "because" and "furthermore", and generally occur in the beginning of the sentence.

- **Bushy path of the node:**

The business of a node (sentence) on a graph to represent the document is defined as the number of links connecting it to other nodes on the graph. Since highly bushy

node is linked to a number of other nodes, it has an overlapping vocabulary with several sentences in the text and is likely to discuss topics covered in many sentences.

- **Aggregate similarity:**

Aggregate similarity measures the importance of a sentence. Instead of counting the number of links connecting a node (sentence) to other nodes, aggregate similarity sums the weights on the links, in order to find the most effective sentences of the text. Sentences with high aggregate similarity are important and are to be included in the summary.

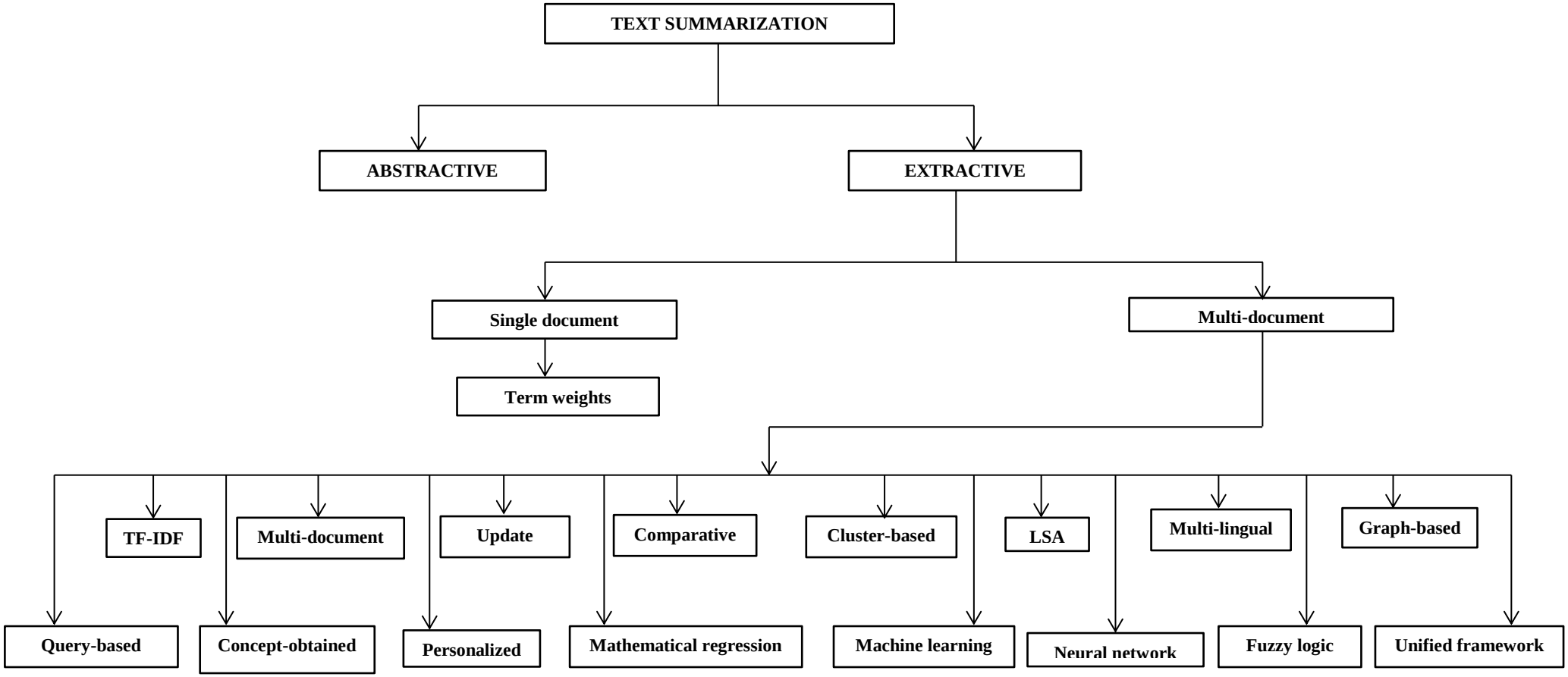

Figure 1.1 Categorization of text summarization techniques

Chapter 2
Literature Review

This chapter discuss various extractive text summarization methodologies in depth, mathematically and graphically.

2.1 Text Summarization using Term Weights

This technique is the evolution of proposed paradigms in the earliest research (Luhn, 1958) [2], to generate auto-abstracts of technical papers and magazines, the complete text of an article in machine-readable form is scanned by IBM 704 data-processing machine and analysed using standard program to derive information of word and phrase frequency. This statistical information of text is used by the machine to compute a relative measure of significance, first for individual words and then for sentences, highest scored sentences are extracted to form "auto-abstracts".

Text summarization using term weights [5] is generally used for single document summarization. In this method first the individual terms in the text are obtained by filtering out the stop words. Then frequencies of individual terms of the document are calculated and weight to each term is assigned as follows:

$$term_{weight} = \frac{frequency\ of\ the\ term}{total\ no.of\ terms\ in\ the\ document} \quad (1)$$

Boost factor is added to the special effect terms appearing in bold, italic, underlined, numeric, or any combination of these.

Boost factor is calculated as follows:

$$b = \frac{frequency\ of\ special\ effect\ term * boost\ value}{total\ no.of\ special\ effect\ terms\ in\ the\ document} \quad (2)$$

Now sentences are weighted as follows:

$$wt_s = \sum_{i=1}^{n} (wt_i)\ /n \quad (3)$$

where $\boldsymbol{wt_s}$ is the weight of the sentence, $\boldsymbol{wt_i}$ represent the weight of individual terms in that sentence and $\boldsymbol{n}$ is the total number of terms in the sentence.

Finally, higher ranked sentences are extracted according to the percentage of summary required by the user and are displayed in the same sequence as in the source document. The relevancy of the summary calculated w.r.t. human judgment is 62.87% for Ms Word and 72.2% for the summarizer developed (Balabantaray et al., 2012) [5]. Since the summarization follows extraction of top weighted sentences for the summary, it might happen that one sentence contains a proper noun and the next sentence contains a pronoun. In this case if summary considers the second sentence without considering the first one, then it does not give its proper meaning. It's a big issue in automatic text summarization.

2.2 Query based Text Summarization

In query based text summarization method [6], sentences in a given document are scored based on the frequency count of terms (words or phrases) and degree of similarity between query and sentence. The sentence containing query phrases are given higher score than the ones containing single query words.

Each sentence of the document is assigned a score. The features considered for this purpose are *term frequency, position of the sentence, length of the sentence and query-sentence similarity.*

Query-Sentence similarity is computed using cosine similarity measure, as the inner product of query and sentence vectors. Query based methodology is one of the main ingredients of the searching algorithms in the search engines such as google and bing, to extract relevant documents corresponding to the user query from an overwhelming number of documents available on the internet. Search engines only display short extracts under the search results, e.g. two lines of text fragments which consist of query words and their surrounding text in accordance to the structure of the original text [7]. Automated summarization system for web search uses structural and linguistic information obtained from the documents both in the summarization process and in generating output summaries, to improve the search experience of the users.

Query specific opinion summarization system: QOS [8], generate summary with relevance to the input opinion questions. The QOS has several modules to accomplish this: a question analysis and query reformulation module, a latent semantic indexing

based sentence scoring module, a sentence polarity detection module, and a redundancy removal module.

BAYESUM [9] is a model for sentence extraction in query based summarization, works well in general scenario, when there are many relevant documents known for a single query.

2.3 Concept-Obtained Text Summarization approach

The idea of concept-obtained approach [10] is to obtain the concepts of words based on HowNet [11], and use these concepts as features instead of words. These Concepts are used to construct conceptual vector space model and then summary of documents is generated using various standard and hybrid algorithms [12].

HowNet is an online common-sense knowledgebase used for the semantic analysis of the text documents. HowNet unveil the inter-conceptual and inter-attribute relationships of concepts.

This approach consists of the following four main stages:

Stage 1: Obtain the concepts in the given text using HowNet and establish conceptual vector space model.

Stage 2: Measure the importance of concepts based on conceptual vector space model.

Stage 3: Measure the importance of sentences on the basis of concepts.

Stage 4: Extract important sentences and calculate the degree of similarity between extracted sentences in order to reduce the redundancy because sentences having high similarity measure describe the same theme.

Concept-obtained summarization has been implemented for various languages such as Chinese and Farsi [13].

2.4 Term Frequency-Inverse Document Frequency (TF-IDF) method

TF-IDF [14] is a multi-document text summarization technique. Let D be a collection of N documents, $\{d_0, d_1, \dots\dots, d_{N-1}\}$. Each document d ∈ D is represented by the set of M distinct terms $\{t_0, t_{1,\dots\dots\dots,} t_{M-1}\}$. Terms are words or *n*-grams (sequence of n consecutive words). Let $n_{i,j}$ be the number of occurrences of term t_i in document d_j. For a given document d_j and a term t_i , the *term frequency* TF(*i, j*) is directly

proportional to the number of times the term t_i occur in the document d_j divided by length of the document :

$$TF(i,j) = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{4}$$

Let $|D|$ be the cardinality of D, and $|\{d: t_i \in d\}|$ be the number of documents in which the term t_i appears. IDF for a term t_i, the *inverse document frequency* IDF (*i*) is the total number of documents divided by the number of documents containing term t_i :

$$IDF(i) = \log \frac{|D|}{|\{d:\ t_i\ \in\ d\}|} \tag{5}$$

IDF was proposed by Salton [15] for improving information retrieval systems. TF-IDF ranking function assigns higher weight to the terms in the document, which appears frequently in that document and rarely in other documents. The term frequency-inverse document frequency of term t_i in document d_j is defined as follows:

$$\text{TF-IDF}(i, j) = \text{TF}(i, j)*\text{IDF}(i) \tag{6}$$

The problem with TF-IDF is that in case of massive stream of documents TF-IDF weighting function require huge amount of memory space and it's also more time consuming, but gives good summary.

2.5 Personalized Text Summarization

Information overload is one of the most serious problems of present-day internet. Personalized text summarization aims to extract the most important information from a document, which can help readers to decide whether it is relevant for them or not. It takes into account differences in reader's characteristics and relevant domain specific terms. Method of personalized text summarization [16] based on Latent Semantic Analysis consists of following steps:

- Pre-processing
- Construction of a personalized terms by sentences matrix
- Singular value decomposition (SVD)
- Sentences selection

Park proposed automatic personalized text summarization agent using generic relevance weight based on non-negative matrix factorization [17]. User-adapted summarizer studies user interests [18] [19] [20] [21] [22].

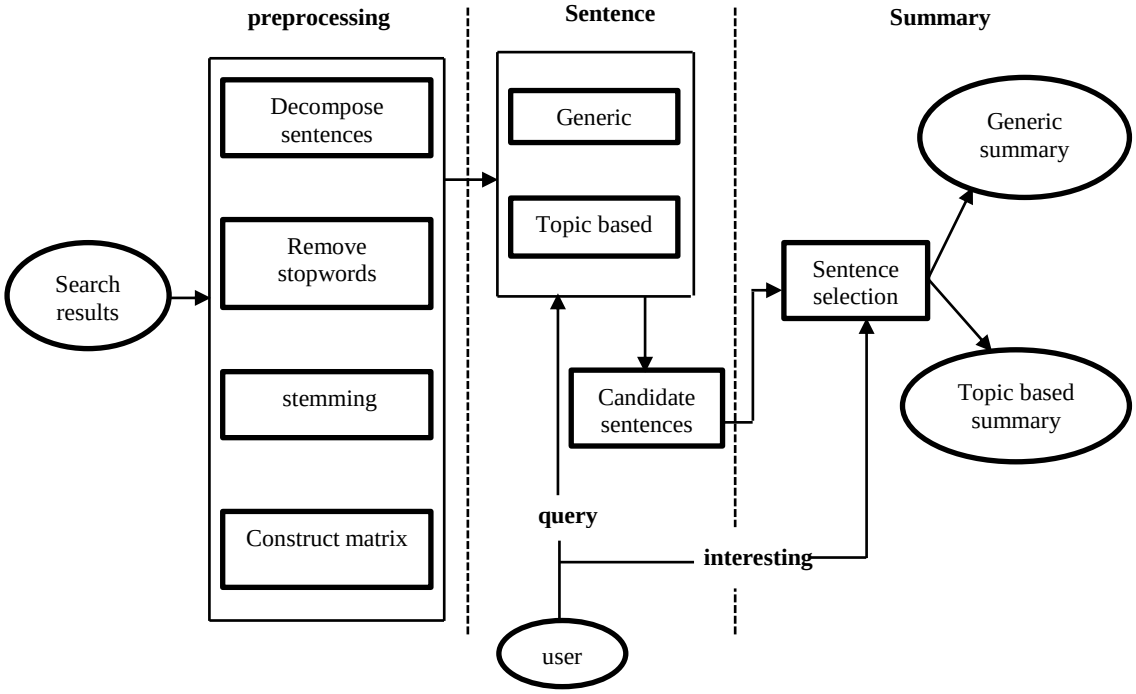

Figure 2.1 Personalized text summarization agent [17]

2.6 Multi-document Extractive Summarization

Multi-document extractive text summarization [23] deals with extraction of key information content from multiple texts written about the same topic. Generated final summary gives concise and comprehensive information about the topic, contained in a large cluster of documents. Given an input of a collection of sets of newspaper articles, NeATS [24] generates summaries in three stages: content selection, filtering and presentation.

Content selection identifies important concepts mentioned in a document collection. In a key step for locating important sentences, NeATS computes the likelihood ratio to identify key concepts in unigrams, bigrams and trigrams, using the on-topic document collection as the relevant set and off-topic document collection as the irrelevant set. This likelihood ratio is used to compute relevancy score of concepts for the given topic. With the individual key concepts available, these concepts are

clustered in order to identify major subtopics within the main topic. Clusters are formed through strict lexical connections.

The purpose content filtering is to remove redundancy and to improve cohesion and coherence. NeATS uses three different filters: sentence position, stigma words and maximum marginal relevancy. Sentence position is a simple sentence filter that only retains the lead 10 sentences. The purpose of stigma filter is to demote sentences containing stigma words to improve cohesion and coherence of summary.

Maximum marginal relevancy (MMR) filter handle redundancy issues. The content selection and above described filtering methods only concerned with individual sentences, they do not consider the concept overlap redundancy issue when two top ranked sentences refer to the similar concept. To address this problem MMR is used. A sentence is added to the summary if and only if its content has less than ***X*** percent overlap with the summary. The overlap ratio is computed using simple stemmed word overlap and the threshold ***X*** is computed empirically.

2.7 Update Summarization

Due to fast evolution of information on the internet, update summarization [25] [26] has become very important, it's a form of multi-document summarization. It is to summarize an evolutionary document collection at current time assuming that the user has already read some previous related documents. It's accomplished by assigning higher weights to the current event. It's one of the main implementations is in producing short update summary of newswire articles to inform reader about new information update regarding a particular topic.

Update summarization [27] is the task to generate summaries of texts while minimizing redundancy of previously read documents.

2.8 Comparative Summarization

Comparative summarization [25] [28] is to generate a short summary delivering the information differences in a collection of document groups by extracting the most discriminative sentences of each document group. A discriminative sentence selection method is used to extract the most discriminative sentences which represent the specific characteristics of each document group.

To implement this technique first the summary of each group of documents using traditional methods such as *Centroid-based method, LexPageRank, LSA(Latent Semantic Analysis)* is generated. These traditional methods extract candidate sentences from each group of documents and then selects the sentences from each group that are farthest from the candidates in other groups in terms of cosine similarity. These baselines are executed in an incremental manner. That is, they start from the most representative sentence in the first cluster, and select the sentence in the second cluster which is farthest from the first one and then select the sentence from the third cluster which is farthest from the two selected sentences, and so on.

2.9 Cluster based Text Summarization

Text clustering [29] is widely applicable in areas such as information retrieval and web mining. Sentence clustering based summarization approach [30] consists of three steps: first cluster the sentences based on semantic distance among sentences of the document, and then on each cluster calculate the accumulative sentence similarity based on the multiple feature fusion method, at last choose the topic sentences using extraction rules.

A. Similarity measure between sentences:

- ***Word form similarity:***

It's the similarity between two sentences measured by the number of same words in two sentences excluding stop words. If S_1 and S_2 are two sentences then their word form similarity is calculated as follows:

$$Sim1(S1, S2) = 2 * \left(\frac{SameWord(S1,S2)}{Len(S1)+Len(S2)}\right) \quad (7)$$

Here SameWord(S_1,S_2) is the number of same words in the sentences S_1 and S_2, excluding stop words. Len(S) is the number of words in the sentence S excluding stop words.

- ***Word order similarity:***

Different sequence of words stand for different meanings, the word order similarity is mainly used to describe the sequence similarity between two sentences.

Here we represent sentence using three vectors as follows:

$V_1 = \{d_{11}, d_{12}, \ldots\ldots, d_{1n1}\}$; weight d_{1i} in V_1 is the TF-IDF value of words in the given sentence.

$V_2 = \{ d_{21}, d_{22}, \ldots\ldots, d_{2n2}\}$; weight d_{2i} in V_2 is to represent the occurrence of bi-gram in the sentence, 0 stands for occurring and 1 stands for non-occurring.

$V_3 = \{ d_{31}, d_{32}, \ldots\ldots, d_{3n3}\}$; weight d_{3i} in V_3 is the tri-gram whether occur in the sentence.

- ***Word semantic similarity:***

The word semantic similarity is mainly used to describe the semantic similarity between two sentences.

- ***Sentence similarity:***

The sentence similarity is computed using the values of three types of similarities defined above, it's usually described as a number between zero and one, zero stands for non-similar and one stands for total sentence similarity. The larger the number is, the more the sentences are similar.

B. Estimating the number of clusters:

Determining the optimal number of sentence clusters in a text document is a difficult task and depends on the compression ratio of the summary and similarity measures used. The strategy based on distribution of words in the sentences can be used to determine the optimal number of clusters:

$$k = n\frac{|D|}{\sum_{i=1}^{n}|S_i|} = n\frac{|\bigcup_{i=1}^{n} S_i|}{\sum_{i=1}^{n}|S_i|} \quad (8)$$

Here $|D|$ is the number of terms in the document D, $|S_i|$ is the number of terms in sentence S_i and n is the number of sentences in document D.

C. Sentence clustering:

Once the number of sentence clusters is known, we can use K-means method to cluster the sentences of the document.

This algorithm can be described as follows:

Input: n sentences

K: the number of clusters

Output: clustered sentences

Step1: Select K sentences randomly to form K clusters, these sentences represent the initial central sentences of the clusters.

Step2: Assign each sentence to the cluster that has the closest central sentence.

Step3: After all sentences have been clustered into K clusters, recalculate the central sentence of each cluster. The central sentence is the one which has the lowest accumulative similarity.

Step4: Repeat steps 2 and 3 until there is no variation in the central sentence. This produces a separation of the sentences into K clusters.

D. Topic sentences extraction:

Based on the results of sentence clustering phase assume that the clusters formed are: $\{C_1, C_2, \ldots\ldots, C_k\}$. Assume that the similarity of the central sentence μ_i as 1, sort the sentences of each cluster based on their similarity weight and choose the high weight sentences as the topic sentences. Sentences should be extracted from every cluster to generate a good informative summary.

2.10 LSA method

LSA (Latent Sematic Analysis) method [31] [32] identifies the semantically important sentences to generate summary. LSA methodology generate summaries by extracting top ranked sentences of a document that are different from each other, to create summaries with a wider coverage of the document's main content and less redundancy.

First step to generate summary using LSA methodology is to create generic summary to provide an overall sense of the document content, then singular value decomposition (SVD) is applied to generic text summarization. SVD is a very powerful mathematical tool that can find principal orthogonal dimensions of multidimensional data. The process starts with the creation of terms by sentences matrix $\mathbf{A} = [A_1\ A_2\ \ldots..\ A_n]$ with each column vector A_i representing the weighted term-frequency vector of sentence i (S_i) of the document under consideration. If there are a total of m terms and n sentences in the document, then the matrix **A** for the given document will be of order $m \times n$, as every word does not appear in each sentence, the matrix **A** will be a sparse matrix.

Matrices are the key ingredient of various algorithms such as google page rank algorithm [33], which are backbone of information science.

For a given matrix **A** of order $m \times n$, where $m \geq n$ without the loss of generality as the number of terms are generally greater than the number of sentences in the document, the SVD of **A** is defined as:

$$\boldsymbol{A = U \sum V^T} \tag{9}$$

where $\mathbf{U} = [u_{i\,j}]$ is an $m \times n$ column-orthonormal matrix whose columns are called left singular vectors, $\sum$ is an $n \times n$ diagonal matrix, and $\mathbf{V} = [v_{i\,j}]$ is an $n \times n$ orthonormal matrix whose columns are right singular vectors.

SVD derives the latent semantic structure from the document represented by matrix **A.** LSA methodology aims to group documents which are semantically related to each other, even when they do not share common words, hence gives non-redundant and meaningful summaries with a wider coverage of the document contents.

2.11 Automatic Text Summarization using Machine Learning approach

Machine learning (ML) summarization procedure [34] is based on the application of trainable machine learning algorithms, sentences are extracted on the basis of statistical and linguistic features. Sentences are classified as summary and non-summary sentences based on features that they possess.

A trainable summarizer using a machine learning approach employ following set of features to weight sentences:

- Mean TF-ISF (Term Frequency-Inverse Sentence Frequency).
- Sentence Length.
- Sentence Position.
- Similarity to Title.
- Similarity to keywords.
- Sentence-to-Sentence cohesion.
- Sentence-to-Centroid cohesion.
- Occurrence of proper nouns.
- Occurrence of anaphors.
- Occurrence of non-essential information.

ML-based summarization framework consists of following steps:

- Standard pre-processing information retrieval (IR) methods such as stop word removal, case folding and stemming are applied to each document.
- Sentences are transformed into their vectorial representations.
- Various sentence features are calculated.
- Finally, machine learning trainable algorithm is employed [35] [36].

2.12 Text summarization using Regression for Estimating Feature Weights

In statistics, regression analysis is a statistical technique for estimating the relationship among variables. In this methodology [37], sentences of each document are transformed into vectors of features extracted from the text. The summarizer is expected to learn the patterns from reference summaries, to generate summary of the text document under consideration, by identifying relevant features. A summary is generated by extracting a set of high score sentences from the source text and present them in the same order as they exist in the original document. Text features used to weight sentences are as follows:

- Sentence position
- Positive keyword in the sentence
- Negative keyword in the sentence
- Sentence similarity with other sentences
- Sentence resemblance to the title
- Proper noun presence in the sentence
- Sentence inclusion of numerical data
- Sentence relative length
- Bushy path of a node (sentence)
- Aggregate similarity

Mathematical Regression (MR) model:

Mathematical regression is a good model to estimate the text feature weights [38]. In this model a mathematical function can relate output to input. The feature parameters of several manually summarized documents are used as independent input variables and the corresponding dependent outputs are specified in the training phase. Relationship between inputs and outputs is deduced to model the system. Then testing

data are introduced to the system model for evaluation of its efficiency. In matrix notation regression is represented as follows:

$$\begin{bmatrix} O_0 \\ O_1 \\ : \\ : \\ O_m \end{bmatrix} = \begin{bmatrix} I_{01} & I_{02} & I_{03} & \dots\dots\dots & I_{010} \\ : & : & : & : & : \\ : & : & : & : & : \\ : & : & : & : & : \\ I_{m1} & I_{m2} & I_{m3} & \dots\dots\dots & I_{m10} \end{bmatrix} . \begin{bmatrix} w_1 \\ w_2 \\ : \\ : \\ w_{10} \end{bmatrix} \quad (10)$$

where

[*O*] is the output vector.

[*I*] is the feature parameter matrix.

[*w*] is the linear statistical model of the system ($w_1, w_2, \dots\dots, w_{10}$ are weights corresponding to text features, obtained using Genetic Algorithm). Genetic Algorithm is the way of solving problems using approaches of natural processes.

2.13 Multilingual Extractive Text Summarization

Multilingual text summarization is a technique of summarizing the texts in different languages to the target language final summary. SimFinderML [39] is a framework for computing multilingual text similarity to identify similar pieces of text, using multiple features. SimFinderML improves the readability of English summaries of multilingual texts by replacing sentences in different languages with machine translated highly similar English sentences.

MINDS(Multilingual, Interactive Document Summarization) [40] integrates multi-lingual and multi-document summarization capabilities using a multi-engine, core summarization system. MINDS produce summaries both in English and in the original language of the document. It provides fast, interactive document access through hypertext summaries, hence operate in real time and it also produce document cross links as a byproduct.

MEAD [41] is a comprehensive, public domain, open source platform for multi-document multi-lingual text summarization and evaluation. Its source and documentation can be downloaded from http://www.summarization.com/mead. The platform implements multiple summarization algorithms such as position-based, centroid-based and query-based. MEAD's architecture consists of four stages**:** First, documents in a cluster are converted to MEAD's internal (XML-based) format.

Second, a number of features are extracted for each sentence of the cluster. Third, these features are combined into a composite score for each sentence. Fourth, these scores can be further refined after considering possible cross-sentence dependencies (e.g. chronological ordering, sentence repetition etc.). In addition to a number of command-line utilities, MEAD provides a Perl API which lets external programs to access its internal libraries.

The MEAD evaluation toolkit (MEADEval) uses various evaluation metrics such as precision, recall, cosine similarity and word overlap for the evaluation of summaries generated using MEAD.

2.14 Graph based Text Summarization

Initial step involved in the process of summarizing one or more documents using graph based methodology [42] is to identify the issues or topics addressed in the document. Graph theoretic representation of text document provides a method to identify these themes. After the common pre-processing steps of stop word removal and stemming, sentences in the document are represented as nodes in an undirected graph. Two sentences are connected with an edge if they share some common words or if their cosine similarity measure is above some threshold value.

Graphical representation of text document yields two results:

- Sub graphs that are not connected to other sub graphs, represent distinct topics covered in the text. In case of query-specific summaries, sentences may be chosen from the pertinent sub graph and in case of generic summaries representative sentences may be chosen from every sub graph.
- The nodes with high cardinality represent important sentences, hence are most likely to be included in the final summary.

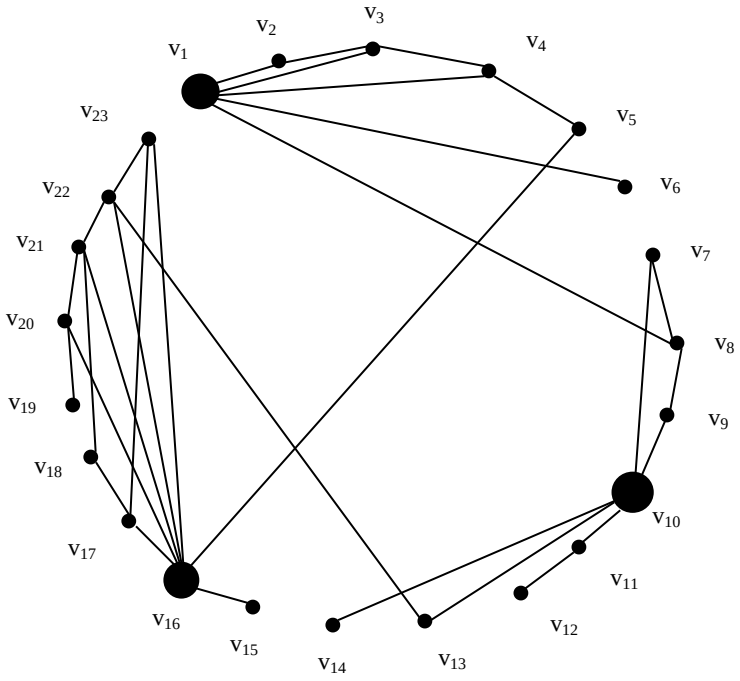

Figure 2.2 Graphical representation for a text

In figure 2.2 graphical representation for a text document is shown. It can be seen that there are three partitions in the graph, hence the document covers three distinct topics. The nodes that are large in size represent the most informative sentences of the document, since they share vocabulary with many other sentences in the document and are likely to discuss the information content of many other sentences. Summary can be generated by sorting all the nodes in the decreasing order of their global connectivity score (degree of node), and then extracting sentences corresponding to the *n* top-ranked nodes, where *n* is the target number of sentences in the summary. This idea is based on Salton et al.'s approach [43] that performs extraction at paragraph level. They suggested that nodes with high connectivity has an overlapping vocabulary with several other nodes, and is likely to discuss topics covered in many other paragraphs, such nodes are good candidates for extraction.

2.15 Text Summarization using Neural Networks

Text summarization using neural networks [44] is a machine learning approach that uses artificial neural networks. A neural network is trained on a corpus of articles to learn the relevant characteristics of sentences that should be included in the summary. Here a three layered feedforward neural network is used, which has been proved to be

a universal function approximator [45]. It can discover the patterns of any data to an accuracy of 100%. The neural network is then modified through feature fusion to generalize the relevant characteristics apparent in summary sentences. Finally, the modified neural network is used as a filter to summarize text documents. Every sentence of each document is represented using a vector [$f_1, f_2, ..., f_n$], composed of several features such as paragraph location in document, sentence location in paragraph, sentence length and number of thematic or title words in the sentence.

Summarization process consists of three main phases:

- **Neural network training:** In this phase neural network is trained to learn the type of sentences that should be included in the summary. This is accomplished by training the neural network with sentences in several test paragraphs where each sentence is identified as to whether it should be included in the summary or not. The three-layered feed forward neural network can discover the patterns and approximate the inherent functions of any data to an accuracy of 100%, as long as there are no contradictions in the dataset.

  If sentence vector is composed of only five feature elements then the neural network after training is as follows:

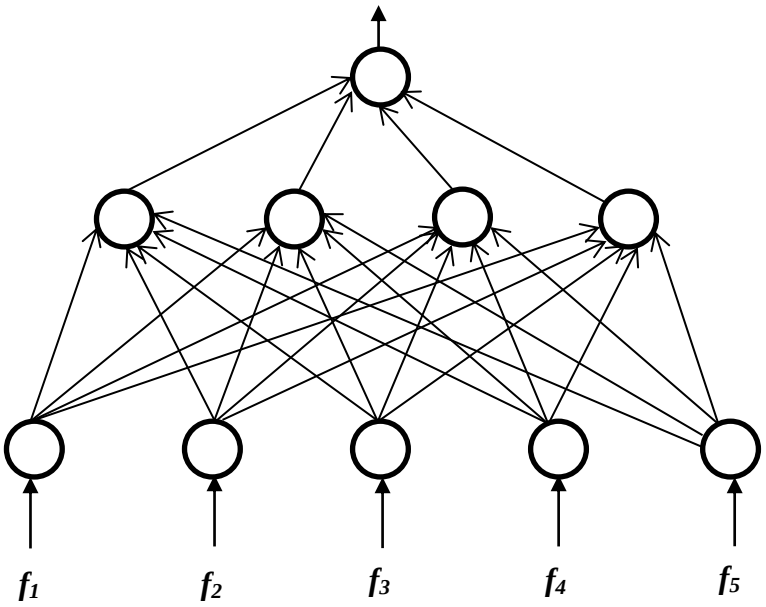

Figure 2.3 Neural network after training

The connections having low weights after training can be pruned without affecting the performance of the network. After the removal of low weight connections, input or hidden layer neuron having no emanating connections can be removed safely from the network, this leads to the removal of uncommon features from the network.

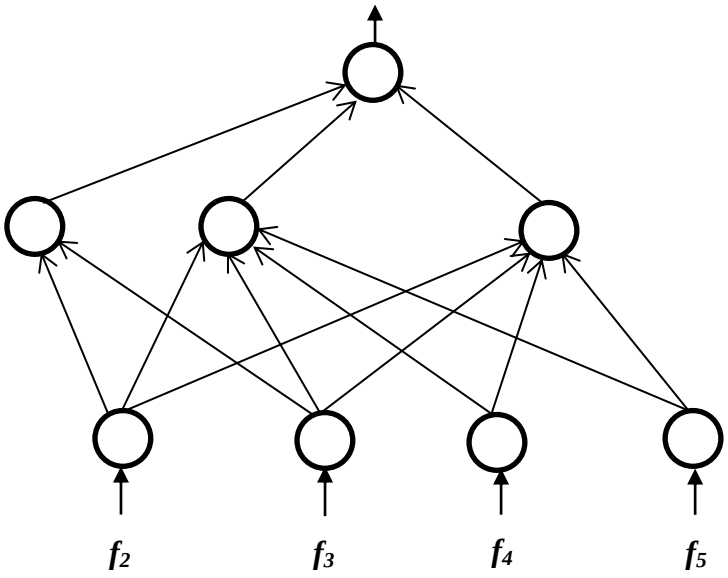

Figure 2.4 Neural network after pruning

- **Feature Fusion:** In this phase the trends and relationships among the features that are inherent in majority of summary sentences are discovered, this is accomplished in two steps: 1) eliminating uncommon features; and 2) collapsing the effect of common features.

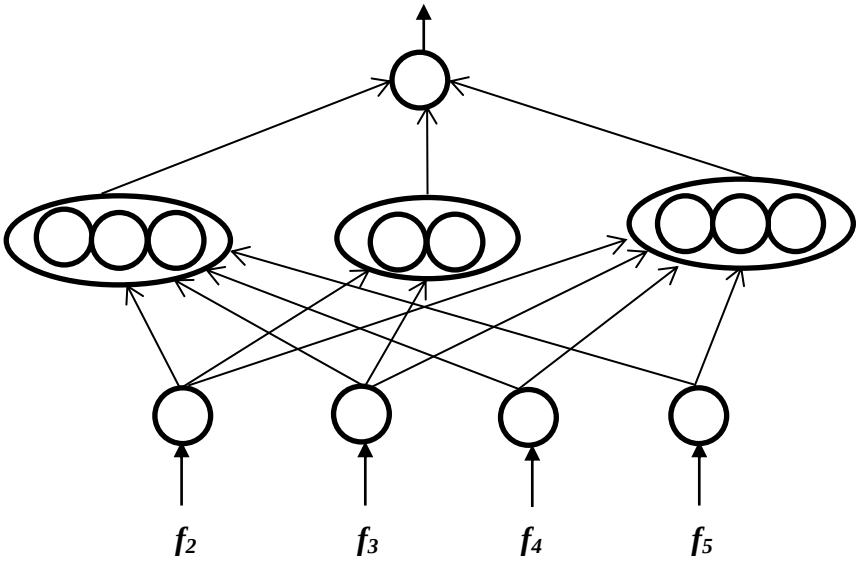

Figure 2.5 Neural network after feature fusion

The hidden layer activation values for each hidden layer neuron are clustered using an adaptive clustering technique. Each cluster is identified by its centroid and frequency. The activation value of each hidden layer neuron is replaced by the centroid of the cluster, this corresponds to collapsing the effects of common features.

The combination of pruning and feature fusion in neural network corresponds to generalizing the effect of features and providing control parameters for sentence ranking. Once the network has been trained, pruned and generalized, it can be used as a tool to filter sentences in any paragraph and determine which sentences to be included in the summary.

2.16 Text Summarization based on Fuzzy Logic

Fuzzy logic is a form of many valued or probabilistic logic. It deals with reasoning that is approximate rather than fixed and exact. Compared to traditional binary sets where variables may take on true or false values, fuzzy logic variables may have a truth value that ranges between 0 and 1. Fuzzy logic has been extended to handle the concept of partial truth, where the truth value ranges between completely true and completely false. Fuzzy logic allows approximate values as well as incomplete or ambiguous data (fuzzy data) as opposed to only relying on crisp data.

Text summarization approach based on fuzzy logic [46] considers various characteristic of a text such as sentence length, title feature, term weight, sentence position, sentence-to-sentence similarity, proper nouns, thematic words, and numerical data as input to the fuzzy system [47]. Fuzzy logic system design usually implicates selecting fuzzy rules and membership functions. This selection of fuzzy rules and membership functions directly affect the performance of the fuzzy logic system. The input membership function for each feature is divided into five fuzzy set which are composed of unimportant values (low and very low), median and important values (high and very high). The fuzzy logic system consists of four components: fuzzifier, inference engine, defuzzifier and the fuzzy knowledge base. In fuzzifier crisp inputs are translated into linguistic values. After fuzzification, the inference engine refers to the rule base containing fuzzy IF-THEN rules to derive the linguistic values. In last step, the output linguistic variables from the inference engine are converted to final crisp values by the defuzzifier using membership function for

representing the final sentence score. Text summarization using fuzzy logic is generally implemented using MATLAB.

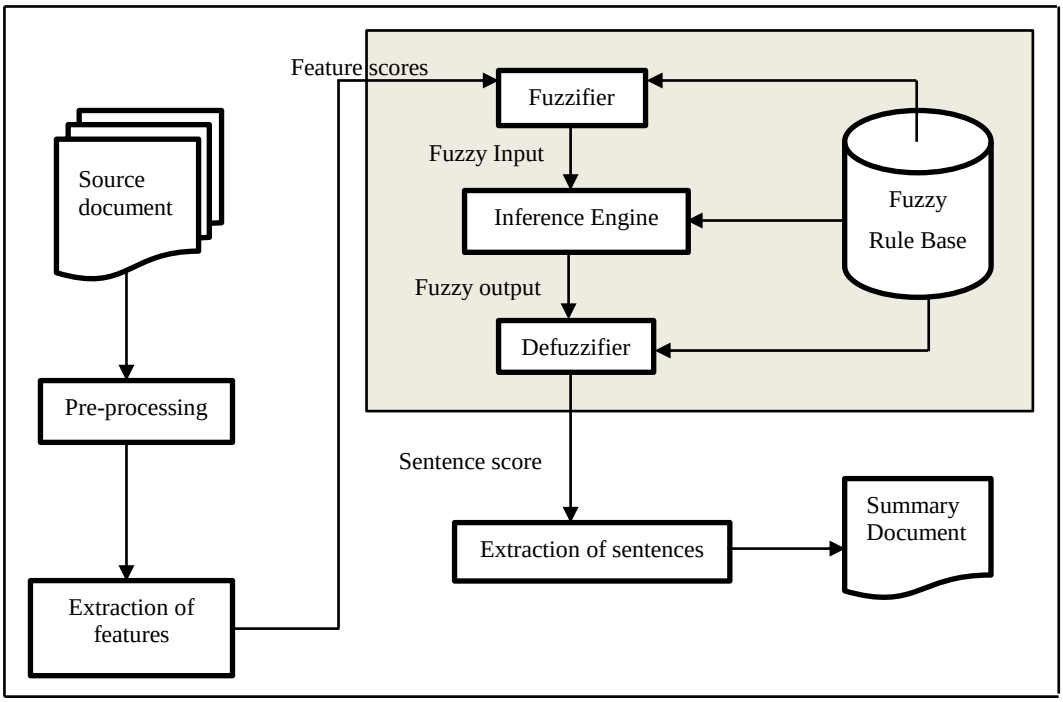

Figure 2.6 Fuzzy logic system architecture for text summarization [46]

According to ROUGE, the fuzzy summarizer reaches average precision of 0.49769, recall of 0.45706 and f-measure of 0.47181, while MS Word 2007 achieves the average precision of 0.47242, recall of 0.40778 and f-measure of 0.43026, on the basis of DUC2002 data set.

2.17 Unified Framework for Opinion Retrieval, Mining and Summarization

Unified framework [48] composed of three crucial components: information retrieval, opinion mining and text summarization.

Information Retrieval (IR) is the discipline that deals with the retrieval of information in response to a query. IR techniques are applied to different types of documents, such as geographic, images, audio and video data [49]. Opinion mining (OM) handles the semantic analysis task of analysing, classifying and extracting the subjective information associated to a specific target.

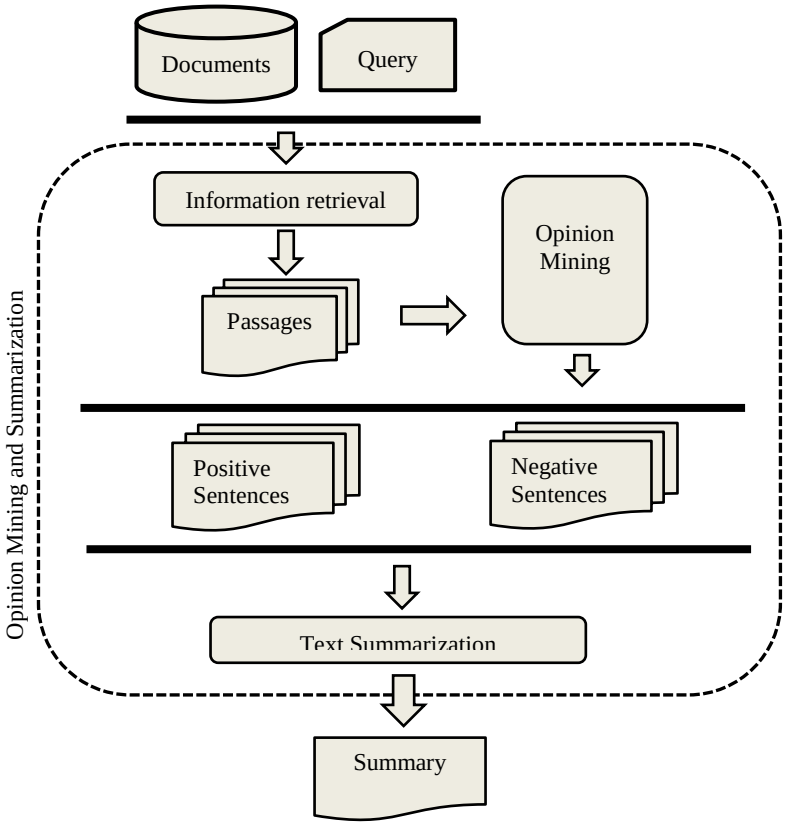

Figure 2.7 Unified framework [48]

In this manner, through this framework it would be possible to search and retrieve relevant content from Web, classify the information found in them, and in the last step, extract the most relevant information for generating summaries of a desired size.

2.18 Summary Evaluation

Summary evaluation is done to decide the quality of summary generated by the summarizer. In the evaluation, summaries generated by different methods are compared with human generated summaries using various evaluation measures as follows:

- ROUGE toolkit: ROUGE [50] stands for Recall-Oriented Understudy for Gisting Evaluation. It's widely applied by Document Understanding Conference (DUC) for performance evaluation of text summarization. It measures the quality of summary by counting the unit overlaps between the candidate summary and a set of reference summaries. Several evaluation methods are implemented in ROUGE, such as ROUGE-N, ROUGE-L, ROUGE-W and ROUGE-SU. ROUGE-N is an *n*-gram recall computed as follows:

$$ROUGE - N = \frac{\sum_{S \in ref} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in ref} \sum_{gram_n \in S} Count(gram_n)} \quad (11)$$

where ***n*** is the length of *n*-gram, and ***ref*** stands for set of reference summaries. $Count_{match}(gram_n)$ is the maximum number of *n*-grams co-occurring in candidate summary and the reference summaries, and $Count(gram_n)$ is the number of *n*-grams in the reference summaries. *ROUUGE-L* uses the longest common subsequence (LCS) statistics, while *ROUGE-W* is based on weighted LCS, and *ROUGE-SU* is based on skip-bigram plus unigram. Each of these evaluation methods can generate three scores: recall (R), precision (P), and F-measure (F), defined as follows:

$$R = \frac{|S_{ref} \cap S_{cand}|}{|S_{ref}|} \quad (12)$$

$$P = \frac{|S_{ref} \cap S_{cand}|}{|S_{cand}|} \quad (13)$$

$$F = \frac{2RP}{R + P} \quad (14)$$

A good summary should have high recall and precision measures, tends to unity.

- Compression ratio (C_r): A good summary is a highly compressed informative representation of its source text. Compression ratio is measured as follows:

  $$C_r = \frac{S_{len}}{T_{len}} \quad (15)$$

  where S_{len}, T_{len} are the length of summary and source text respectively. A good summary is the one having small $C_r$value, tends to zero.

- Retention ratio (R_r): This measure represents the extent to which a summary represent the information content of its source document. An informative summary is the one having high retention ratio (tends to unity), measured as follows:

  $$R_r = \frac{S_{info}}{T_{info}} \quad (16)$$

  S_{info}, T_{info} represent the information content in the summary and corresponding source text document respectively.

Summarization Techniques	Statistical approach	Linguistic approach	Redundancy	Ambiguity	Retention ratio	System/Model	Multi-document effectiveness	Semantics	Trainable Summarizer	Memory usage	Time	Quality of summary
Query-based	Yes	No	High	High	Medium	BAYESUM	Medium	Medium	No	Medium	Medium	Medium
Term weights	Yes	No	High	High	Low	-	Low	Low	No	Medium	Low	Low
Concept-obtained	Yes	Yes	Low	Medium	High	HowNet	Medium	High	No	Medium	High	High
TF-IDF	Yes	No	Medium	High	High	-	High	Medium	No	High	High	High
Personalized	Yes	Yes	Low	Low	High	-	Medium	Medium	No	Medium	Medium	Medium
Multi-document	Yes	Yes	Low	Low	Medium	NeATS	High	Medium	No	High	High	Medium
Update	Yes	Yes	Low	High	Medium	-	High	Medium	No	Low	Medium	Medium
Comparative	Yes	Yes	Low	High	High	-	High	Medium	No	High	High	High
Cluster-based	Yes	Yes	Low	Low	High	-	High	Medium	No	Medium	High	High
LSA	Yes	Yes	Low	Low	High	-	High	High	No	High	High	High
Machine learning	Yes	Yes	Low	Low	High	-	High	High	Yes	Medium	High	High
Mathematical regression	Yes	No	Low	Low	High	-	Medium	Medium	Yes	Medium	High	High
Multi-lingual	Yes	No	Low	Low	Medium	SimFinderML, MINDS, MEAD	High	Medium	No	Medium	High	High
Graph-based	Yes	No	Low	Low	High	-	High	Medium	No	Medium	High	High
Neural network	Yes	No	Low	Medium	Medium	-	Medium	Medium	Yes	Medium	Medium	Medium
Fuzzy logic	Yes	No	Low	Medium	High	-	Medium	Medium	Yes	Medium	High	High
Unified framework	Yes	No	Low	Medium	High	-	High	Medium	No	Medium	High	High

Table 2.1 Analysis of extractive text summarization techniques

Chapter 3
Problem Statement

Since extractive text summarization follows the extraction method, when it extracts the important sentences it might happen that one sentence contains a proper noun and some next sentence contains a pronoun as a reference of the proper noun. In that case, if the summary considers the second sentence without considering the first one, then it does not give its proper meaning. It will create ambiguity in the generated summary, as reader will not come to know that to whom (proper noun) this pronoun refers. It's a big issue in automatic text summarization. In this thesis work is done to resolve this type of anaphoric problems in text summarization.

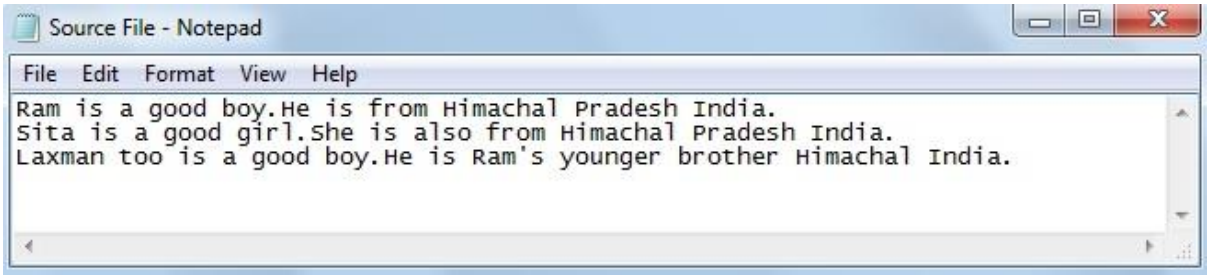

Figure 3.1 Source file

Figure 3.2 Generated summary

As we can see that the summary generated in Figure 9 of source file in Figure 8 is ambiguous, as reader of the of the summary will not come to know who is "She" in the third line of the summary. It happened because the sentence containing "Sita" do not have enough weight to be included in the summary i.e. 50% top weighted sentences of the source file.

Chapter 4
Implementation

This chapter discusses how the problem stated in previous chapter can be solved using proposed method.

Steps involved in generating unambiguous summary:

1. Open source file in read mode.
2. Read the file character by character and store each word of the file in separate 2D array location.
3. Close source file.
4. Enter boost words.
5. Term weighting.
6. Sentence weighting.
7. Generate unambiguous summary.

4.1 Storing Source File Words

Algorithm 4.1, parse the input file and store the words in a two-dimensional character array. In algorithm 4.1, A and D are two-dimensional character arrays and fp is the file pointer. File operation File-Read(A, fp) reads the input source file using file pointer fp and store the individual words of the file in two dimensional array A. Operation Copy-2Darray(D, A) copy the elements of array A and stores in array D.

Algorithm 4.1: Store words of source file in a two-dimensional array

```
1: procedure store-file-words(A, fp, D)
2: begin
3: File-Read(A, fp)
4: Copy-2Darray(D, A)
5: end
```

4.2 Calculation of Term Weights

Algorithm 4.2, filter out stop words and calculate the weight of content words in the input file, on the basis of their frequency of repetition in the file. In algorithm 4.2, A and C are two-dimensional character arrays and termwt is an integer array. Stop words are non-significant words like is, am, are etc. Boost words are domain specific

significant words. Variable tcount is used as a counter to measure the frequency of content words in the input file and array termwt store the weight of these content words. Operation Copy-String(C[z], A[h]) copies the element at location h of array A into location z of array C. Array C stores distinct content words of the file.

Algorithm 4.2: Calculate the weight of terms in the source file [14]

```
1: procedure term-weighting(A, termwt, C, i)
2: h ≔ 1
3: z ≔ 1
4: v ≔ 1
5: while h≤i do
6:       if A[h] is a stop word then
7:            A[h][1] ≔ '*'
8:            h ≔ h+1
9:       else if A[h][1] = '*' || A[h][1] = '.' then
10:           h ≔ h+1
11:      else
12:           Copy-String(C[z], A[h])
13:           z ≔ z+1
14:           m ≔ h+1
15:           while m ≤ i do
16:                if A[m][1] ≔ '*' || A[m][1] ≔ '.' then
17:                     m ≔ m+1
18:                else if A[h] = A[m] then
19:                     tcount ≔ tcount+1
20:                     A[m][1] ≔ '*'
21:                     m ≔ m+1
22:                else
23:                     m ≔ m+1
24:           if A[h] is a boost word then
25:                termwt[v] ≔ 2 * tcount
26:           else
27:                termwt[v] ≔ tcount
28:           v ≔ v+1
29:      h ≔ h+1
```

4.3 Calculation Sentence Weights

Algorithm 4.3 is used to calculate the weight of sentences in the input file on the basis of term weights calculated in algorithm 4.2. In algorithm 4.3, D and C are two-dimensional character arrays and sentencewt is a one-dimensional array of type float. Variable sweight of type float is used to calculate the weight of sentences in the input file and array sentencewt stores weight of these sentences.

Algorithm 4.3: Calculate the weight of sentences in the source file [14]

```
1: procedure sentence-weighting(D, C, termwt, sentencewt, i, z)
2: sweight :=0
3: k := 1
4: b := 1
5: while k ≤ i do
6:       if  D[k][i] = '.' then
7:             sentencewt[b] := sweight/n
8:             n := 0
9:             b := b+1
10:            sweight := 0
11:            k := k+1
12:      else
13:            v := 1
14:            while v < z do
15:                  if D[k] = C[v] then
16:                        sweight := sweight + termwt[v]
17:                        n := n+1
18:                        v := z
19:                  else
20:                        v := v+1
21:      k := k+1
```

4.4 Summary Generation

Algorithm 4.4 is used to generate the final summary of the input source file on the basis of sentence weights calculated in algorithm 4.3. In algorithm 4.4, sentencewt, Sswt, ns and ss are one-dimensional integer type arrays, sentencewt is used to store the sentence weights in the same sequence as in file, sorted sentence weights are stored in Sswt, ns is used to store sequence number of the sentences containing nouns and ss is used to store the sequence number of the summary sentences in the input source file. Operations noun-class(ui, nd, nc, noun) and display-summary(sn, g, k, D, nclass) are explained using algorithm 5 and algorithm 6 respectively.

Algorithm 4.4: Generates summary of source file

```
1: procedure Generate-Summary(sentencewt, Sswt, ns, ss, b, nsn)
2: j ≔ b ∗ p; h ≔ 1; g ≔ 1; ui ≔ 1; sn ≔ 1; nclass ≔ 0
3: while g ≤ j do
4:       k ≔ 1
5:       while k ≤ j do
6:            if sentencewt[h] = Sswt[k] then
7:                 ss[sn] ≔ h
8:                 if ss[sn] ≔ ns[ui] then
9:                      display-summary(sn, g, k, D, nclass)
10:                else if ss[sn] > ns[ui] then
11:                     while ss[sn] > ns[ui] do
12:                          ui ≔ ui+1
13:                          if ui > nsn then
14:                               ui ≔ ui-1
15:                               if temp ≔ ui then
16:                                    display-summary(sn, g, k, D, nclass)
17:                               x ≔ 1
18:                               while x < sn do
19:                                    if ss[x] = ns[ui] then
20:                                         display-summary(sn, g, k, D, nclass)
21:                                    else
22:                                         x ≔ x+1
23:                               noun-class(ui, nd, nc, noun)
24:                               display-summary(sn, g, k, D, nclass)
25:                          else if ns[ui] = ss[sn] then
26:                               display-summary(sn, g, k, D, nclass)
27:                          else
28:                               ui ≔ ui-1
29:                               x ≔ x+1
30:                               while x < sn do
31:                                    if ss[x] = ns[ui] then
32:                                         display-summary(sn, g, k, D, nclass)
33:                                    else
34:                                         x ≔ x+1
35:                               if temp = ui then
36:                                    display-summary(sn, g, k, D, nclass)
37:                               noun-class(ui, nd, nc, noun)
38:                               display-summary(sn, g, k, D, nclass)
39:                else
40:                     dispay-summary(sn, g, k, D, nclass)
41:           else
42:                k ≔ k+1
43:      h ≔ h+1
```

Algorithm 4.5: Identify the class of nouns

```
1: procedure noun-class(ui, nd, nc, noun)
2: x ≔ 1
3: while noun[ui] ≠ nd[x] do
4:       x ≔ x+1
5: if nc[x] is male then
6:       nclass ≔ 1
7: else if nc[x] is female then
8:       nclass ≔ 2
9: else if nc[x] is place then
10:      nclass ≔ 3
11: else
12:      nclass ≔ 0
```

Algorithm 4.5, associates a class with the noun. In algorithm 4.5, nd, noun , nc are two-dimensional character arrays. Array noun store all the nouns of file in the same sequence as they exist in the file. Array nd is used as a noun dictionary and nc stores the corresponding class of the nouns stored in nd. Algorithm 4.6 is used to display the unambiguous summary. Operation display() is used to print the element of the array passed as argument on the output screen.

Algorithm 4.6: Displays summary

```
1: procedure display-summary(sn, g, k, D, nclass)
2: sn ≔ sn+1; g ≔ g+1; i ≔ 1; t ≔ 1
3: while t ≤ h do
4:       if D[i][1] = '.' then
5:             t ≔ t+1
6:       else
7:             i ≔ i+1
8: while D[i][1] ≠ '.' do
9:       if nclass ≠ 0 then
10:            if D[i] ∈ pronoun then
11:                  temp ≔ ui
12:                  display(D[i])
13:                  display((noun[ui]))
14:                  nclass ≔ 0
15:                  i ≔ i+1
16:                  continue
17:            else
18:                  display(D[i])
19:                  i ≔ i+1
20:                  continue
21:      else
22:            display(D[i])
23:            i ≔ i+1
24:            continue
```

Chapter 5
Testing and Results

In this thesis, algorithm developed to resolve anaphoric problems in text summarization is implemented practically in C language.

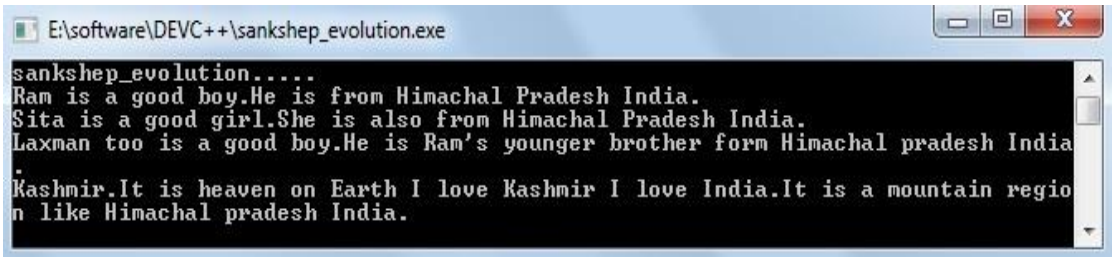

Figure 5.1 Parsed input source file

The figure 5.1 displays the input source file parsed character by character till the end of file is not detected and figure 5.2 shows the words of the parsed input file, which are stored using a two-dimensional character array.

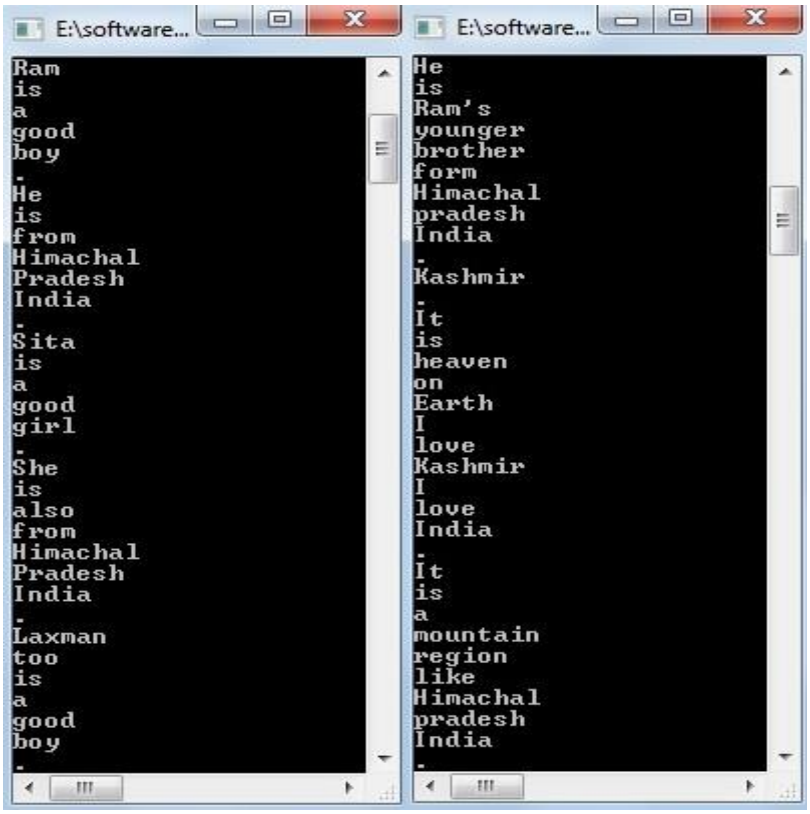

Figure 5.2 File words

Figure 5.3 shows the creation of noun database. In the developed text summarizer sankshep_evolution using the proposed algorithm, its used as a noun dictionary to identify nouns and their corresponding noun class.

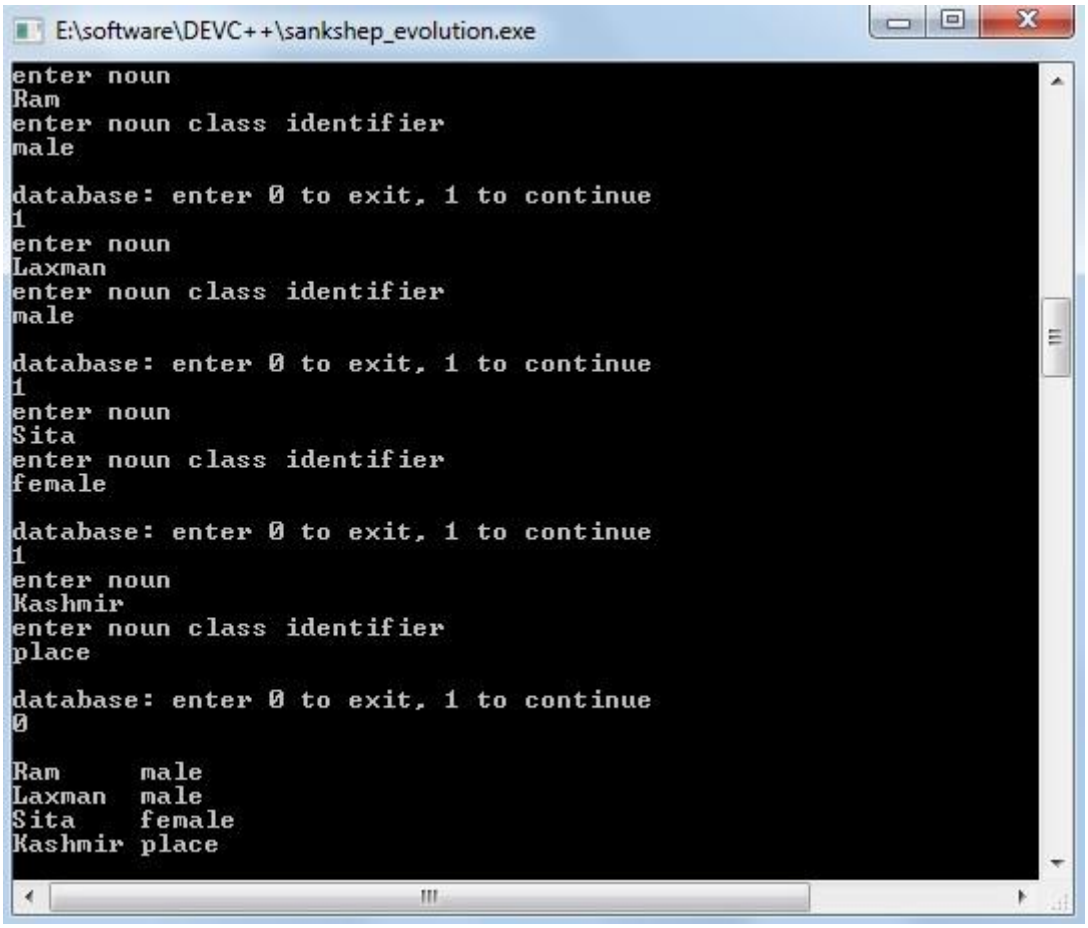

Figure 5.3 Noun database

In figure 5.4, nouns occurring in input file are displayed along with their corresponding sentence numbers, this is accomplished using noun database.

Figure 5.4 Source file nouns

Figure 5.5 shows manual entry of boost words by the user to generate summary of user's interest. Boost words are generally domain specific significant words.

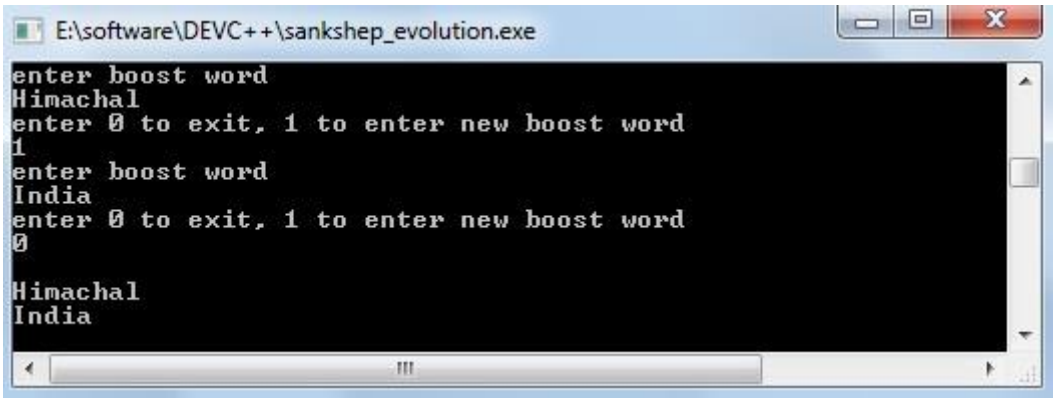

Figure 5.5 Boost words

In figure 5.6, list of all the distinct content words in the source file is displayed. Content words are the terms left after filtering out the stop words.

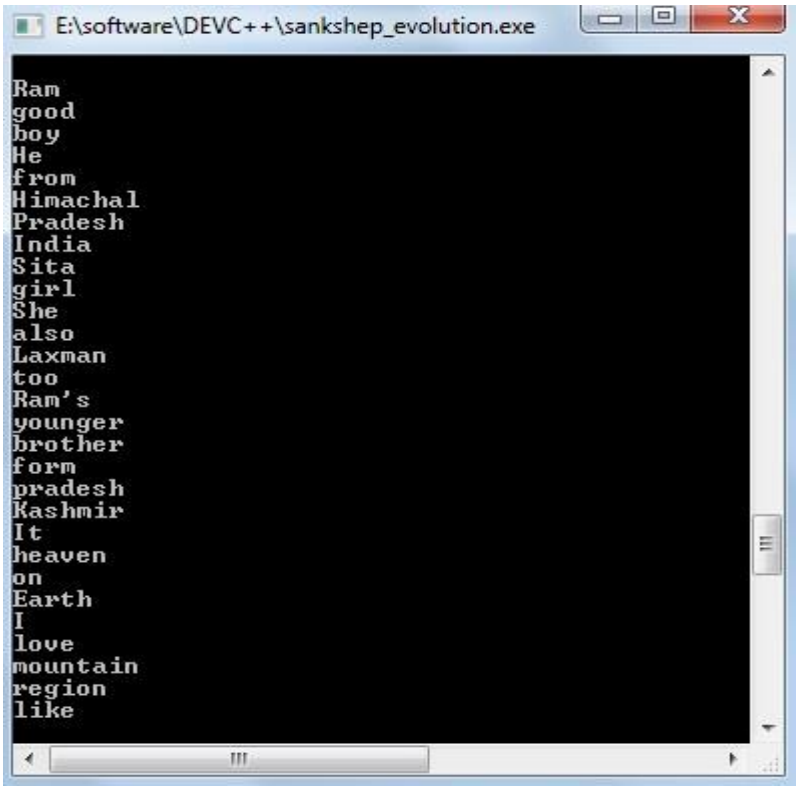

Figure 5.6 Content words

Figure 5.7 shows the frequency of occurrence of content words in the source file, this is used in assigning weight to the individual content words.

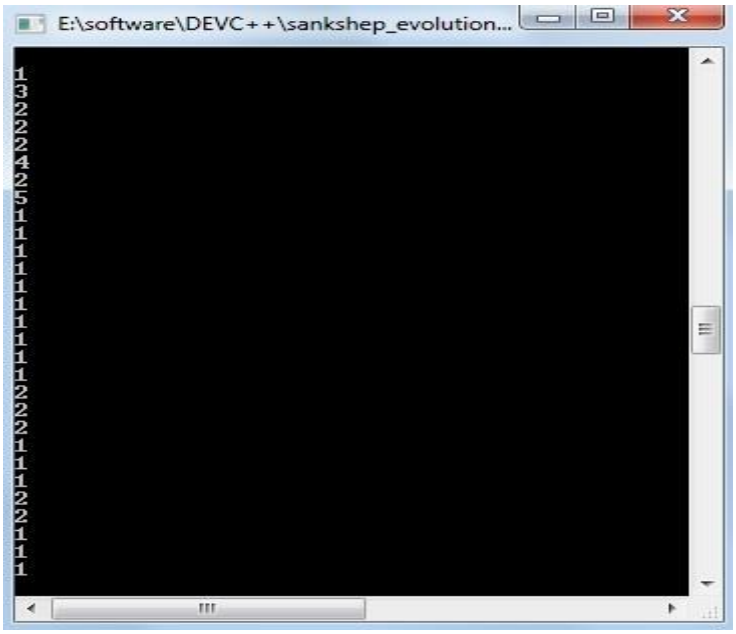

Figure 5.7 Frequency of content words

In figure 5.8, weight of content words are displayed. These weights are calculated using frequency of the content word and boost word list.

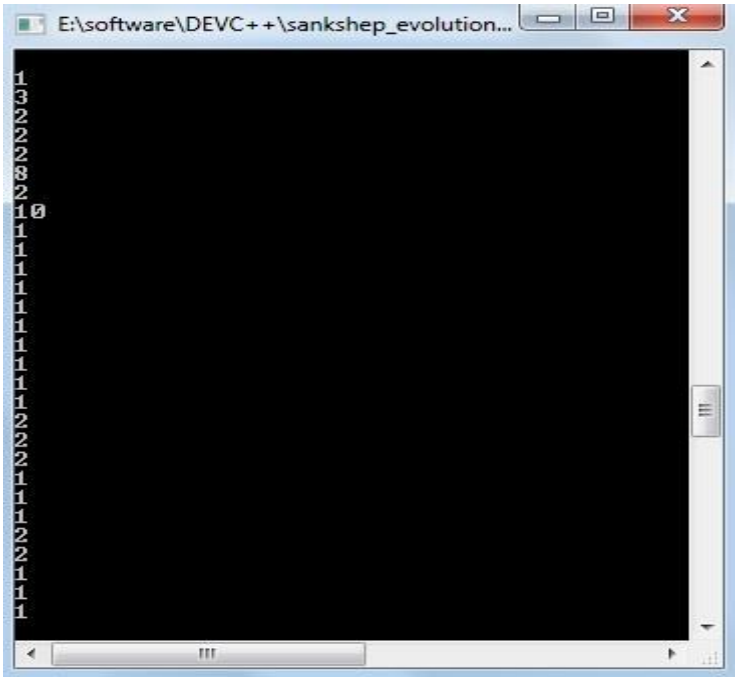

Figure 5.8 Weight of content words

In figure 5.9, lists the weight of sentences in sequence orders and sorted order. Sentence weights are calculated by summing up the weight of all content words in the sentence divided by total number of content words in that sentence.

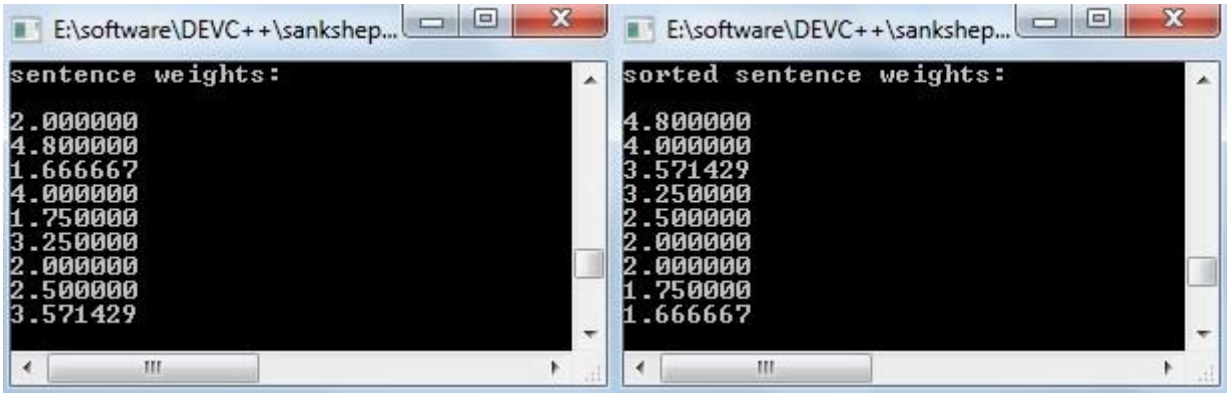

Figure 5.9 Sentence weights

Figure 5.10, displays 100% of the document as summary i.e. the whole document.

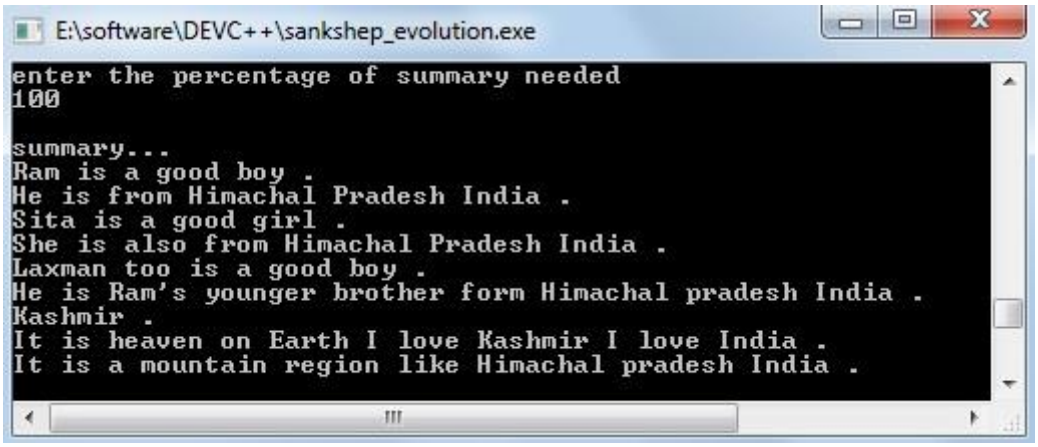

Figure 5.10 Full input file as 100% summary

In figure 5.11, 50% top weighted sentences of the file are displayed as summary, it can also be seen that the proposed algorithm resolves all the anaphoric ambiguities in the generated summary and once the ambiguity has been removed for an ambiguous pronoun then the further related pronouns to that noun will be displayed as it is, as in case of Kashmir in this case.

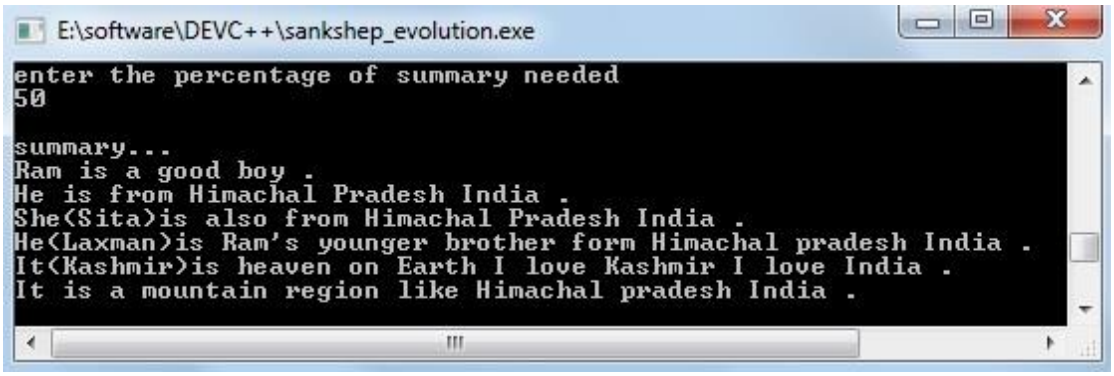

Figure 5.11 Unambiguous 50% summary

In figure 5.12, top weighted 70% sentences are displayed as summary in accordance to the user's value of the summary required.

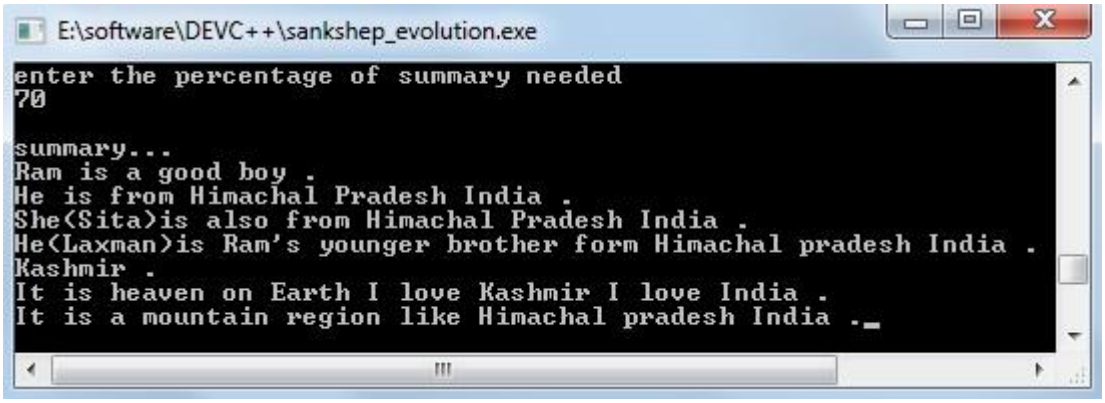

Figure 5.12 Unambiguous 70% summary

In figure 5.13, 40% top weighted sentences are displayed as summary.

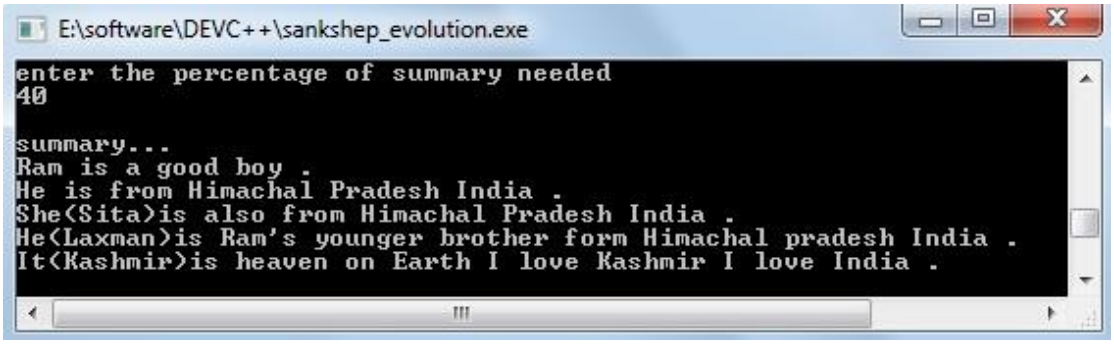

Figure 5.13 Unambiguous 40% summary

Chapter 6
Conclusion and Future Scope

6.1 Conclusion

In today's fast growing information era, there is huge availability of information content on internet. It's very difficult to find out relevant documents from an overwhelming number of documents on internet corresponding to a user query. In order to solve this problem of selecting relevant texts and extracting key significant information content of each text, automatic text summarizer is a very useful tool. Summary construction is, in general, a complex task which would involve deep natural language processing. In order to simplify this problem, current research is focused on extractive summarization methodology. An extractive summary is simply a subset of the sentences of the original text. These summaries do not guarantee a good narrative coherence, but they can conveniently represent an approximate content of the text. Different type of techniques are used for different purposes such as update and comparative summarization technique is widely used in developing mobile applications to generate short updates on newswire articles, multilingual summarization technique is used for summarizing text documents of different languages, some technique are useful in generating non-redundant and unambiguous summaries, some techniques are effective only for single document summarization while some are also good in generating summaries of multiple documents. Text summarizers nowadays are developed by fusing different text summarization methodologies together as in MS Word, to generate coherent and cohesive summaries.

Features are the pillars of text summarization methodologies, so deciding proper weight of individual features is very important as the quality of summary depends on it. Summaries can be evaluated using various evaluation measures like precision, recall and retention.

6.2 Thesis Contribution

In this thesis a detailed description of seventeen different extractive text summarization methodologies has been presented and work is done to resolve anaphoric problems in text summarization. The comparison between seventeen different extractive text summarization techniques is done using twelve different parameters, which can help to choosing the best suited techniques in accordance to the resources available for the development of various applications such as multi-document and multi-lingual text summarizers, news update mobile applications etc.

6.3 Future Scope

In future the limitations of the algorithm implemented can be improved in a wide range of fields:

- Work can be extended by linking the application developed to an online extensive noun database.
- Abstractive approach along with wide range of suitable features can be used to improve the coherence and cohesiveness of the generated summaries.
- Work can be done to reduce time complexity and memory usage, it can also be extended for multi-documents, documents in multiple languages, audio etc.
- Work can be extended for improving the methodology proposed to remove anaphoric ambiguities in the summary to a wide range of anaphoric classes.
- Boost in the weight of terms appearing in italic, bold, underlined and numeric data, as they are generally important.

References

[1] V. Gupta and G. S. Lehal, "A Survey of Text Summarization Extractive Techniques," *Journal of Emerging Technologies in Web Intelligence*, vol. 2, no. 3, pp. 258-268, 2010.

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

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

[4] H. P. Edmundson, "New methods in automatic extracting," *Journal of ACM*, vol. 16, no. 2, pp. 264-285, 1969.

[5] R. C. Balabantaray, D. K. Sahooo, B. Sahoo and M. Swain, "Text Summarization using Term Weights," *International Journal of Computer Applications*, vol. 38, no. 1, pp. 10-14, 2012.

[6] T. J. Siddiqui and U. S. Tiwary, "Query based summary for assessing document relevance," in *1st International Conference on Digital Information Management*, bangalore, 2006.

[7] F. C. Pembe and T. Güngör, "Automated Query-biased and structure-preserving text summarization on web documents," in *Proceedings of International Symposium on Innovations in Intelligent Systems and Applications*, Istanbul, 2007.

[8] F. Jin, M. Huang and X. Zhu, "A query-specific opinion summarization system," in *8th IEEE International Conference on Cognitive Informatics* , 2009.

[9] D. Marcu and H. D. III, "Bayesian query-focused summarization," in *ACL-44 proceedings of the 21st International Conference on Computational Linguistics and the 44th annnual meeting of the Association of Computational Linguistics*, 2006.

[10] M. Wang, X. Wang and C. Xu, "An approach to concept-obtained text summarization," in *IEEE International Symposium on Communications and Informationn Technology,2005.ISCIT 2005.*, 2005.

[11] Z. Dong, Q. Dong and C. Hao, "HowNet and its computation of meaning," in *COLING '10 Proceedings of 23rd International Conference on Computational Linguistics*, Beijing, 2010.

[12] W. Meng and T. Xinlai, "Extract summarization using concept-obtained and hybrid parallel genetic algorithm," in *Eighth International Conference on National Computation*, 2012.

[13] A. Zamanifar, B. Minaei-Bidgoli and M. Sharifi, "A new hybrid Farsi text summarization technique based on term co-occurrence and conceptual property of the text," in *Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, IEEE*, Iran, 2008.

[14] R. C. Balabantaray, D. k. Sahoo, B. Sahoo and M. Swain, "Text summarization using term weights," *International Journal of Computer Applications*, vol. 38, no. 1, pp. 10-14, 2012.

[15] G. Salton and C. Buckley, Term-weighting approaches in automatic text retrieval, 1988, pp. 513-523.

[16] R. Móro and M. Bielikov", "Personalized text summarization based on important terms identification," in *23rd International Workshop on Database and Expert Systems Applications*, 2012.

[17] S. Park, "Automatic Personalized text summarization agent using generic relevance weight based on NMF," in *23rd International conference on Information Networking*, 2009.

[18] A. Díaz and P. Gervás, "User-model based personalized summarization," *Information Processing and Management*, vol. 43, no. 6, pp. 1715-1734, 2007.

[19] S. Park , J.-W. Lee and J.-W. Song, "Automatic personalized summarization using non-negative matrix factorization and relevance measure," in *IEEE International workshop on semantic computing and application*, 2008.

[20] S. Park, "Personalized document summarization using non-negative semantic feature and non-negative semantic variable," in *proceedings of IDEAL*, 2008.

[21] S. Park, "Personalized summarization agent using non-negative matrix factorization," in *PRICAI2008: Trends in Artificial Intelligence*, Springer, 2008, pp. 1034-1038.

[22] M. Sanderson, "Accurate user directed summarization from existing tools," in *proceedings of the international conference on information and knowledge management*, 1998.

[23] C. Lin and E. Hovy, "From single to multi-document summarization: A prototype system and its evaluation," in *40th annual meeting of the Association for Computational Linguistics*, Philadelphia, 2002.

[24] C. Lin and E. Hovy, "Automated multi-document summarization in NeATS," in *2nd International conference on Human Laguage Technology Research*, 2002.

[25] J. Li, L. Li and T. Li, "Multi-document summarization via submodularity," *Applied Intelligence,* vol. 37, no. 3, pp. 420-430, 2012.

[26] X. Li, L. Du and Y. Shen, "Update summarization via graph-based sentence ranking," *IEEE transactions on Knowledge and Data Engineering,* vol. 25, no. 99, pp. 1162-1174, 2012.

[27] M. Peng, X. Ma, Y. Tian, M. Yang, H. Long, Q. Lin and X. Xia, "The web information extraction for update summarization based on shallow parsing," in *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, 2011.

[28] D. Wang, S. Zhu, T. Li and Y. Gong, "Comparative document summarization via discriminative sentence selection," *ACM Transaction on Knowledge Discovery from Data,* vol. 7, no. 1, 2013.

[29] M. Chen and Y. Song, "Summarization of text clustering based vector space model," in *IEEE 10th International Conference on CAID CD*, 2009.

[30] Z. Pei-ying and L. Cun-he, "Automatic text summarization based on sentences clustering and extraction," in *IEEE International Conference on Computer Science and Information Technology*, 2009.

[31] Y. Gong and X. Liu, "Generic text summmarization using relevance measure and latent semantic analysis," in *24th International ACM SIGIR Conference on research and development in information retrieval*, 2001.

[32] J. Steinberger and K. Jažek, "Text summarization and singular value decomposition," in *Advances in Information Systems*, Heidelberg, Springer Berlin, 2005, pp. 245-254.

[33] N. HareshKumar and D. Garg, "Random web surfer pagerank algorithm," *International Journal of Computer Applications*, vol. 35, no. 11, pp. 36-41, 2011.

[34] J. L. Neto, A. A. Freitas and C. A. A. Kaestner, "Automatic text summarization using a machine learning approach," in *16th Brazilian Symposium on Artificial Intelligence*, 2002.

[35] T. M. Mitchell, Machine learning, New york, USA: McGraw-Hill, 1997.

[36] S. L. Salzberg, "C4.5: Programs for machine learning," *Machine Learning*, vol. 16, no. 3, pp. 235-240, 1994.

[37] M. A. Fattah and F. Ren, "Automatic text summarization," *International Journal of Computer Science*, vol. 3, no. 1, pp. 25-28, 2008.

[38] B. Jann, "making regression tables from stored estimates," *Stata Journal*, vol. 5, no. 3, pp. 288-308, 2005.

[39] D. K. Evans, "Identifying Similarity in Text: Multi-Lingual Analysis for Summarization," PhD thesis, Graduate School of Arts and Sciences, Columbia University, 2005.

[40] J. Cowie, K. Mahesh, S. Nirenburg and R. Zajac, "MINDS - Multi-lingual Interactive Document Summarization".

[41] D. Radev, T. Allison, S. Blair-Goldensohn, J. Blitzer, A. Celebi, S. Dimitrov, E. Drabek, A. Hakim, W. Lam, D. Liu, J. Otterbacher, H. Qi, H. Saggion, S. Teufel, M. Topper, A. Winkel and Z. Zhang, "MEAD - a platform for multidocument multilingual text summarization," in *Proceedings of LREC*, Lisbon,Portugal, 2004.

[42] C. Kruengkrai and C. Jaruskulchai, "Generic text summarization using local and global properties of sentences," in *Proceedings of IEEE International Conference on Web Intelligence*, 2003.

[43] G. Salton, A. Singhal, M. Mitra and C. Buckley, "Automatic text structuring and summarization," *Information Processing & Management,* vol. 33, no. 2, pp. 193-207, 1997.

[44] K. kaikhah, "Text summarization using neural networks," in *Proceedings of 2nd International Conference on Intelligent Systems*, 2004.

[45] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards,* vol. 49, no. 6, pp. 409-436, 1952.

[46] L. Suanmali, N. Salim and M. S. Binwahlan, "Fuzzy logic based method for improving text summarization," *International Journal of Computer Science and Information Security,* vol. 2, no. 1, 2009.

[47] F. kyoomarsi, H. Khosravi, E. Eslami, P. K. Dehkordy and A. Tajoddin, "Optimizing text summarization based on fuzzy logic," in *proceedings of seventh IEEE/ACIS International Conference on Computer and Information Science*, 2008.

[48] E. Lloret, A. Balahur, J. M. Gómez, A. Montoyo and M. Palomar, "Towards unified framework for opinion retieval, mining and summarization," *Journal of Intelligent Information Systems,* pp. 1-37, 2012.

[49] Y. A. Aslandogan and C. T. Yu, "Techniques and systems for image and video retrieval," *IEEE TRANSACTIONS on KNOWLEDGE and DATA ENGINEERING,* vol. 11, no. 1, pp. 56-63, 1999.

[50] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Proceedings of the ACL-04 Workshop on Text Summarization Branches Out*, barcelona, 2004.

List of Publications

[1] Avnish Thakur, Ravinder Kumar & Neeraj Kumar, "Extractive Text Summarization Techniques: A Survey", *ACM Computing Surveys* [communicated].