Automatic Summarization Tool For Text Documents

Project Report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology.

in

Computer Science & Engineering

under the Supervision of

Dr. Pardeep Kumar

By

Kamal Kumar(111260)

to

Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh

Certificate

This is to certify that project report entitled "Automatic Summarization Tool For Text Documents", submitted by Kamal Kumar in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science & Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date: **Dr. Pardeep Kumar**

Assistant Professor

Acknowledgement

There are many people who are associated with this project directly or indirectly whose help and timely suggestions are highly appreciable for completion of this project. First of all, I would like to thank **Prof. Dr. RMK Sinha**, Dean, Department of Computer Science Engineering and Prof. Dr. SP Ghrera, Head, Department of Computer Science Engineering for his kind support and constant encouragements, valuable discussions which is highly commendable.

I would like to express my sincere gratitude to my supervisor Dr. Pardeep Kumar, for his super vision, encouragement, and support which has been instrumental for the success of this project. It was an invaluable experience for me to be one of his students. Because of him, I have gained a careful research attitude.

Thanks to those who are also the part of this project whose names could have not been mentioned here.

Lastly, I would also like to thank my parents for their love and affection and especially their courage which inspired me and made me to believe in myself.

Date: KAMAL KUMAR
Roll No. 111260

Abstract

The increasing availability of online information has necessitated intensive research in the area of automatic text summarization within the Natural Language Processing (NLP) community. ***Automatic text summarization*** is technique of compressing the original text into shorter form which will provide same meaning and information as provided by original text. The brief summary produced by summarization system allows readers to quickly and easily understand the content of original documents without having to read each individual document. The overall motive of text summarization is to convey the meaning of text by using less number of words and sentences. Summaries are of two types: *Abstractive summaries* and *Extractive summaries.* ***Extractive summaries*** involve extracting relevant sentences from the source text in proper order. The relevant sentences are extracted by applying statistical and language dependent features to the input text. On the other hand, ***abstractive text summaries*** are made by applying natural language understanding. Human beings usually make summaries in abstractive way. Moreover abstractive summaries can also involve the words or sentences which are not present in the input text. Automatic generation of abstractive summary is more difficult as compared to producing extractive text summary. This has some applications like summarizing the search-engine results, providing briefs of big documents that do not have an abstract etc..In this project, an auto-summarization tool is developed using statistical techniques. The designed algorithm works in three steps. In the first step the document which is required to be summarized is processed by eliminating the stop word and by applying the stemmers. In the second step term-frequent data is calculated from the document and frequent terms are selected, for these selected words the semantic equivalent terms are also generated. Finally in the third step all the sentences in the document, which are containing the frequent and semantic equivalent terms, are filtered for summarization. The designed algorithm is implemented using open source technology JAVA. It operates on a single document (but can be made to work on multiple documents by choosing proper algorithms for integration) and provides a summary of the document.

Table of Contents

S. No.		Topic	Page No.
1.		Introduction	1
	1.1	Types of Summaries	1
	1.2	How do summarization system work	3
	1.3	Where does summarization help	4
2.		A survey on text summarisation	7
	2.1	Features for Extractive Text Summarisation	9
	2.2	Extractive summarization methods	12
3.		A review of Text Summarisation	27
4.		Design and Implementation	32
	4.1	Document Preprocessing	33
	4.2	Features used	38
5.		Ranking Algorithm	40
	5.1	TextRank	40
	5.2	Why TextRank works	46
6.		Development	48
	6.1	User Interface	48
	6.2	Sentence Extraction	49
	6.3	Unique word Identification	50
	6.4	Weight of a particular word	50
	6.5	Ranking of sentences	51
7.		Experiments	52
	7.1	Syntactic Filtering	52
	7.2	Maximum occurring words	54
	7.3	Using 2 weight classes	55
	7.4	Using 3 weight classes	55
8.		Challenging issues of automatic summarization	56

9.	Application	64
10.	Conclusion and Future work	65
11.	References	66

List of Figures

S.No.	Title	Page No.
1.	Keyword extraction method	9
2.	Graph theoretic approach	14
3.	Neural network after prunning	19
4.	Neural network after feature fusion	20
5.	Multi Document summarisation	25
6.	Text Comparator	28
7.	Open Text Summariser	30
8.	Text Summarisation System Architecture	31
9.	User Interface	46
10.	Sentence Extraction	47
11.	Unique word identification	48
12.	Weight of a particular word	49
13.	Ranking	50

List of Tables

S.No.	Title	Page No.
1.	Baseline result	53
2.	Syntactic Filter	53
3.	Cosine similarity	54
4.	Maximum occurring word	54
5.	Using 2 weight classes	55
6.	Using 3 weight classes	55
7.	Consolidated result for text rank	55

CHAPTER 1

INTRODUCTION

Today's world is all about information, most of it online. The World Wide Web contains billions of documents and is growing at an exponential pace. Tools that provide timely access to, and digest of, various sources are necessary in order to alleviate the information overload people are facing. These concerns have sparked interest in the development of automatic summarization systems. Such systems are designed to take a single article, a cluster of news articles, a broadcast news show, or an email thread as input, and produce a concise and fluent summary of the most important information. Recent years have seen the development of numerous summarization applications for news, email threads, lay and professional medical information, scientific articles, spontaneous dialogues, voicemail, broadcast news and video, and meeting recordings. These systems, imperfect as they are, have already been shown to help users and to enhance other automatic applications and interfaces.

1.1 Types of Summaries

There are several distinctions typically made in summarization and here we define terminology that is often mentioned in the summarization literature.

Extractive summaries (extracts) are produced by concatenating several sentences taken exactly as they appear in the materials being summarized. Abstractive summaries (abstracts), are written to convey the main information in the input and may reuse phrases or clauses from it, but the summaries are overall expressed in the words of the summary author.

Early work in summarization dealt with single document summarization where systems produced a summary of one document, whether a news story, scientific article, broadcast show, or lecture. As research progressed, a new type of summarization task emerged: multi-document summarization. Multi-document summarization was motivated by use

cases on the web. Given the large amount of redundancy on the web, summarization was often more useful if it could provide a brief digest of many documents on the same topic or the same event. In the first deployed online systems, multi-document summarization was applied to clusters of news articles on the same event and used to produce online browsing pages of current events. A short one- paragraph summary is produced for each cluster of documents pertaining to a given news event, and links in the summary allow the user to directly inspect the original document where a given piece of information appeared. Other links provide access to all articles in the cluster, facilitating the browsing of news. User-driven clusters were also produced by collecting search engine results returned for a query or by finding articles similar to an example document the user has flagged as being of interest .

Summaries have also been distinguished by their content. A summary that enables the reader to determine has often been called an indicative summary, while one that can be read in place of the document has been called an informative summary . An indicative summary may provide characteristics such as length, writing style, etc., while an informative summary will include facts that are reported in the input document(s).

Much of the work to date has been in the context of generic summarization. Generic summarization makes few assumptions about the audience or the goal for generating the summary. Typically, it is assumed that the audience is a general one: anyone may end up reading the summary. Furthermore, no assumptions are made about the genre or domain of the materials that need to be summarized. In this setting, importance of information is determined only with respect to the content of the input alone. It is further assumed that the summary will help the reader quickly determine what the document is about, possibly avoiding reading the document itself.

In contrast, in query focused summarization, the goal is to summarize only the information in the input document(s) that is relevant to a specific user query. For example, in the context of information retrieval, given a query issued by the user and a set of relevant documents retrieved by the search engine, a summary of each document could make it easier for the user to determine which document is relevant. To generate a useful summary in this context, an automatic summarizer needs to take the query into account as

well as the document. The summarizer tries to find information within the document that is relevant to the query or in some cases, may indicate how much information in the document relates to the query. Producing snippets for search engines is a particularly useful query focused application. Researchers have also considered cases where the query is an open-ended question, with many different facts possibly being relevant as a response. A request for a biography is one example of an open-ended question as there are many different facts about a person that could be included, but are not necessarily required.

Update summarization addresses another goal that users may have. It is multi-document summarization that is sensitive to time; a summary must convey the important development of an event beyond what the user has already seen.
The contrast between generic, query-focused, and update summarization is suggestive of other issues raised by Sparck Jones in her 1998 call to arms . Sparck Jones argued that summarization should not be done in a vacuum, but rather should be viewed as part of a larger context where, at the least, considerations such as the purpose of summarization (or task which it is part of), the reader for which it is intended, and the genre which is being summarized, are taken into account. She argued that generic summarization was unnecessary and in fact, wrong-headed. Of course, if we look at both sides of the question, we see that those who write newspaper articles do so in much the same spirit in which generic summaries are produced: the audience is a general one and the task is always the same. Nonetheless, her arguments are good ones as they force the system developer to think about other constraints on the summarization process and they raise the possibility of a range of tasks other than to simply condense content.

1.2 How do Summarization Systems Work?

Summarization systems take one or more documents as input and attempt to produce a concise and fluent summary of the most important information in the input. Finding the most important information presupposes the ability to understand the semantics of written or spo- ken documents. Writing a concise and fluent summary requires the capability to

reorganize, modify and merge information expressed in different sentences in the input. Full interpretation of documents and generation of abstracts is often difficult for people,and is certainly beyond the state of the art for automatic summarization.

How then do current automatic summarizers get around this conundrum? Most current systems avoid full interpretation of the input and generation of fluent output. The current state of the art in the vast majority of the cases relies on sentence extraction. The extractive approach to summarization focuses research on one key question: how can a system determine which sentences are important? Over the years, the field has seen advances in the sophistication of language processing and machine learning techniques that determine importance.

At the same time, there have been recent advances in the field which move toward semantic interpretation and generation of summary language. Semantic interpretation tends to be done for specialized summarization. For example, systems that produce biographical summaries or summaries of medical documents tend to use extraction of information rather than extraction of sentences. Research on generation for summarization uses a new form of generation, text-to-text generation and focuses on editing input text to better fit the needs of the summary.

1.3 Where Does Summarization Help?

While evaluation forums such as DUC and TAC enable experimental setups through comparison to a gold standard, the ultimate goal in development of a summarization system is to help the end user perform a task better. Numerous task-based evaluations have been performed to establish that summarization systems are indeed effective in a variety of tasks. In the TIPSTER Text Summarization Evaluation (SUMMAC), single-document summarization systems were evaluated in a task-based scenario developed around the tasks of real intelligence analysts. This large-scale study compared the performance of a human in judging if a particular document is relevant to a topic of interest, by reading either the full document or a summary thereof. It established that automatic text summarization is very effective in relevance assessment tasks on news

articles. Summaries as short as 17% of the full text length sped up decision-making by almost a factor of two, with no statistically significant degradation in accuracy. Query-focused summaries are also very helpful in making relevance judgments about retrieved documents. They enable users to find more relevant documents more accurately, with less need to consult the full text of the document.

Multi-document summarization is key for organizing and presenting search results in order to reduce search time, especially when the goal of the user is to find as much information as possible about a given query .In McKeown et al. paper, users were given a task of writing reports on specified topics, with an interface containing news articles, some relevant to the topic and some not. When articles were clustered and summaries for the related articles were provided, people tended to write better reports, but moreover, they reported higher satisfaction when using the information access interface augmented with summaries; they felt they had more time to complete the task. Similarly, in the work of Mana-L´opez et al. paper, users had to find as many aspects as possible about a given topic. Clustering similar articles returned from a search engine together proved to be more advantageous than traditional ranked list presentation, and consider- ably improved user accuracy in finding relevant information. Providing a summary of the articles in each cluster that conveys the similarities between them, and single-document summaries highlighting the information specific to each document, also helped users in finding information, but in addition considerably reduced time as users read fewer full documents.

In summarization of scientific articles, the user goal is not only to find articles relevant to their interest, but also to understand in what respect a scientific paper relates to the previous work it describes and cites. In a study to test the utility of scientific paper summarization for determining which of the approaches mentioned in the paper are criticized and which approaches are supported and extended, automatic summaries were found to be almost as helpful as human-written ones, and significantly more useful than the original article abstract.

Voicemail summaries are helpful for recognizing the priority of the message, the call-back number, or the caller ; summaries of threads in help forums are useful in deciding if

the thread is relevant, and summaries of meetings are a necessary part of interfaces for meeting browsing and search.

Numerous studies have also been performed to investigate and confirm the usefulness of single document summaries for improvement of other automated tasks. For example, Sakai and Sparck Jones present the most recent and extensive study (others include and several studies conducted in Japan and published in Japanese) on the usefulness of generic summaries for indexing in information retrieval. They show that, indeed, indexing for retrieval based on automatic summaries rather than full document text helps in certain scenarios for precision-oriented search. Similarly, query expansion in information retrieval is much more effective when potential expansion terms are selected from a summary of relevant documents instead of the full document.

Another unexpectedly successful application of summarization for improvement of an automatic task has been reported by . They examined the impact of summarization on the automatic topic classification module that is part of a system for automatic scoring of student GMAT essays. Their results show that summarization of the student essay significantly improves the performance of the topical analysis component. The conjectured reason for the improvement is that the students write these essays under time constraints and do not have sufficient time for revision and thus their writing contains some digressions and repetitions, which are removed by the summarization module, allowing for better assessment of the overall topic of the essay.

The potential uses and applications of summarization are incredibly diverse as we have seen in this section.

CHAPTER 2

A SURVEY ON TEXT SUMMARIZATION

Interest in automatic text summarization, arose as early as the fifties. An important paper of these days is the one in 1958, suggested to weight the sentences of a document as a function of high frequency words, disregarding the very high frequency common words.

Automatic text summarization system in 1969, which, in addition to the standard keyword method (i.e., frequency depending weights), also used the following three methods for determining the sentence weights:

1. Cue Method: This is based on the hypothesis that the relevance of a sentence is computed by the presence or absence of certain cue words in the cue dictionary.

2. Title Method: Here, the sentence weight is computed as a sum of all the content words appearing in the title and (sub-) headings of a text.

3. Location Method: This method is based on the assumption that sentences occurring in initial position of both text and individual paragraphs have a higher probability of being relevant. the results showed, that the best correlation between the automatic and human-made extracts was achieved using a combination of these three latter methods.

The Trainable Document Summarizer in 1995 performs sentence extracting task, based on a number of weighting heuristics. Following features were used and evaluated:

1. Sentence Length Cut-O Feature: sentences containing less than a pre-specified number of words are not included in the abstract

2. Fixed-Phrase Feature: sentences containing certain cue words and phrases are included.

3. Paragraph Feature: this is basically equivalent to Location Method feature in [8].

4. Thematic Word Feature: the most frequent words are defined as thematic words. Sentence scores are functions of the thematic words' frequencies

5. Uppercase Word Feature: upper-case words (with certain obvious exceptions) are treated as thematic words, as well.

A Corpus was used in this method, which contained 188 document/summary pairs from 21 publications in a scientific/technical domain. The summaries were produced by professional experts and the sentences occurring in the summaries were aligned to the original document texts, indicating also the degree of similarity as mentioned earlier, the vast majority (about 80%) of the summary sentences could be classified as direct sentence matches.

The ANES text extraction system in 1995 is a system that performs automatic, domain-independent condensation of news data. The process of summary generation has four major constituents:

1. Corpus analysis: this is mainly a calculation of the tf*idf -weights for all terms

2. Statistical selection: of signature words: terms with a high tf*idf-weight plus headline-words

3. Sentence weighting: summing over all signature word weights, modifying the weights by some other factors, such as relative location

4. Sentence selection: Selecting high scored sentences. Hidden Markov Models (HMMs) : As prove to be a mathematically sound frame-work for document retrieval. If one approaches the task of text abstracting from such a probabilistic modeling perspective, it might well be possible that HMMs could be employed for this purpose, as well.

Clustering: Building links and/or clusters between index terms, phrases and/or other subparts of the documents has been employed by standard information retrieval. Although this is not an issue in any of the above mentioned abstracting systems, it seems to be worth of consideration when building such systems.

2.1 FEATURES FOR EXTRACTIVE TEXT SUMMARIZATION

Some features [5][4] to be considered for including a sentence in final summary are:

2.1.1 Content word (Keyword) feature: Content words or Keywords are usually nouns and determined using tf × idf measure. Sentences having keywords are of greater chances to be included in summary. Another keyword extraction method is given below, having three modules:

1) Morphological Analysis
2) Noun Phrase (NP) Extraction and Scoring
3) Noun Phrase (NP) Clustering and Scoring

Figure1 shows a pictorial representation of the keyword extraction method.

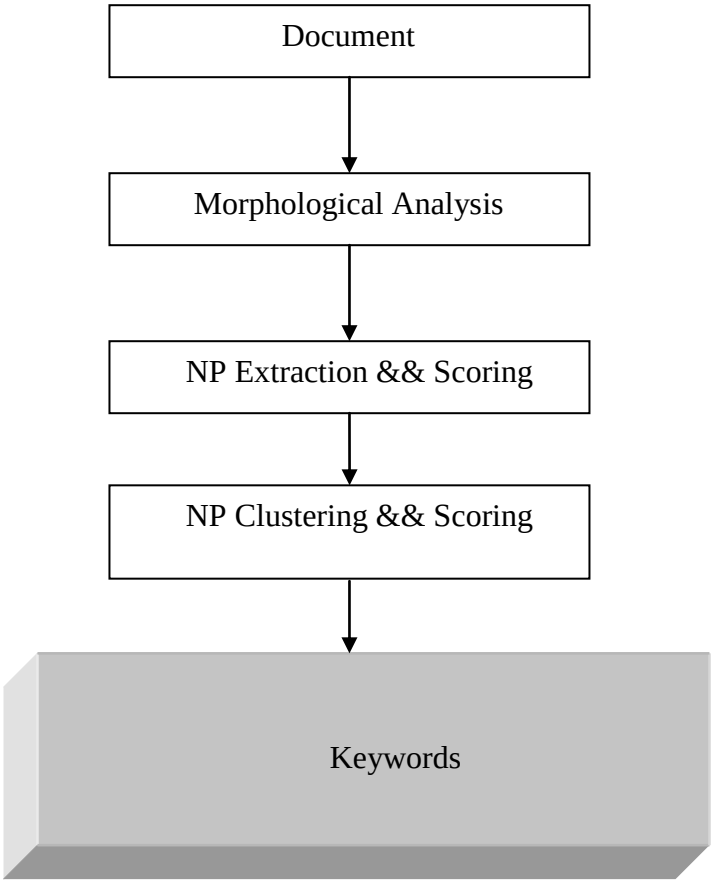


Figure 1. Keyword extraction method

2.1.2 Title word feature:

Sentences containing words that appear in the title are also indicative of the theme of the document. These sentences are having greater chances for including in summary.

2.1.3 Sentence location feature:

Usually first and last sentence of first and last paragraph of a text document are more important and are having greater chances to be included in summary.

2.1.4 Sentence Length feature:

Very large and very short sentences are usually not included in summary.

2.1.5 Proper Noun feature:

Proper noun is name of a person, place and concept etc. Sentences containing proper nouns are having greater chances for including in summary.

F. Upper-case word feature: Sentences containing acronyms or proper names are included.

2.1.6 Cue-Phrase Feature:

Sentences containing any cue phrase (e.g. "in conclusion", "this letter", "this report", "summary", "argue", "purpose", "develop", "attempt" etc.) are most likely to be in summaries.

2.1.7 Biased Word Feature:

If a word appearing in a sentence is from biased word list, then that sentence is important. Biased word list is previously defined and may contain domain specific words.

2.1.8 Font based feature:

Sentences containing words appearing in upper case, bold, italics or Underlined fonts are usually more important.

2.1.9 Pronouns:

Pronouns such as "she, they, it" cannot be included in summary unless they are expanded into corresponding nouns.

2.1.10 Sentence-to-Sentence Cohesion:

For each sentence compute the similarity between s and each other sentence s' of the document, then add up those similarity values, obtaining the raw value of this feature for s. The process is repeated for all sentences.

2.1.11 Sentence-to-Centroid Cohesion:

For each sentence as compute the vector representing the centroid of the document, which is the arithmetic average over the corresponding coordinate values of all the sentences of the document; then compute the similarity between the centroid and each sentence, obtaining the raw value of this feature for each sentence.

2.1.12 Occurrence of non-essential information:

Some words are indicators of non-essential information. These words are speech markers such as "because", "furthermore", and "additionally", and typically occur in the beginning of a sentence. This is also a binary feature, taking on the value "true" if the sentence contains at least one of these discourse markers, and "false" otherwise.

2.1.13 Discourse analysis:

Discourse level information , in a text is one of good feature for text summarization. In order to produce a coherent, fluent summary, and to determine the flow of the author's argument, it is necessary to determine the overall discourse structure of the text and then removing sentences peripheral to the main message of the text.

These features are important as, a number of methods of text summarization are using them. These features are covering statistical and linguistic characterize features are covering statistical and linguistic characteristics of a language.

2.2. EXTRACTIVE SUMMARIZATION METHODS

Extractive summarizers aims at picking out the most relevant sentences in the document while also maintaining a low redundancy in the summary.

2.2.1 Term Frequency-Inverse Document Frequency (TF- IDF) method:

Bag-of-words model is built at sentence level, with the usual weighted term-frequency and inverse sentence- frequency paradigm , where sentence-frequency is the number of sentences in the document that contain that term. These sentence vectors are then scored by similarity to the query and the highest scoring sentences are picked to be part of the summary. This is a direct adaptation of Information Retrieval paradigm to summarization. Summarization is query-specific, but can be adapted to be generic as described below.

To generate a generic summary, non stop-words that occur most frequently in the document(s) may be taken as the query words. Since these words represent the theme of the document, they generate generic summaries. Term- frequency is usually 0 or 1 for sentences since normally the same content-word does not appear many times in a given sentence. If users create query words the way they create for information retrieval, then the query based summary generation would become generic summarization.

2.2.2 Cluster based method:

Documents are usually written such that they address different topics one after the other in an organized manner. They are normally broken up explicitly or implicitly into sections. This organization applies even to summaries of documents. It is intuitive to think that summaries should address different “themes” appearing in the documents. Some summarizers incorporate this aspect through clustering. If the document collection for which summary is being produced is of totally different topics, document clustering becomes almost essential to generate a meaningful summary.

Documents are represented using term frequency- inverse document frequency (TF-IDF)of scores of words. Term frequency used in this context is the average number of occurrences (per document) over the cluster. IDF value is computed based on the entire corpus. The summarizer takes already clustered documents as input. Each cluster is considered a theme. The theme is represented by words with top ranking term frequency, inverse document frequency (TF-IDF) scores in that cluster. Sentence selection is based on similarity of the sentences to the theme of the cluster(Ci) .The next factor that is considered for sentence selection is the location of the sentence in the document (Li). In the context of newswire articles, the closer to the beginning a sentence appears, the higher its weight age for inclusion in summary. The last factor that increases the score of a sentence is its similarity to the first sentence in the document to which it belongs (Fi).

The overall score (Si) of a sentence i is a weighted sum of the above three factors:

Si =W1 *Ci + W2 *Fi+ W3 *Li ……..(2)

where Si is the score of sentence Ci,, Fi are the scores of the sentence i based on the similarity to theme of cluster and first sentence of the document it belongs to, respectively. Li is the score of the sentence based on its location in the document. w1, w2 andw3 are the weights for linear combination of the three scores. Note the similarity between the sentence score in equations (1) and (2). The role of F in (2) is similar to that of T in (1). The difference however, is that Si, in (2) is further re-scored using a redundancy factor. Once the documents are clustered, sentence selection from within the cluster to form its summary is local to the documents in the cluster. The IDF value based on the corpus statistics seems counter-intuitive. A better choice may be to take the Average-TF alone to determine the theme of the cluster, and then rely on the "anti redundancy" factor to cover the important 'themes' within the cluster.

2.2.3 Graph theoretic approach:

As seen in the previous methods, the first step involved in the process of summarizing one or more documents is identifying the issues or topics addressed in the

document. Graph theoretic representation of passages provides a method of identification of these themes. After the common preprocessing steps, namely, stop word removal and stemming, sentences in the documents are represented as nodes in an undirected graph. There is a node for every sentence. Two sentences are connected with an edge if the two sentences share some common words, or in other words, their (cosine, or such) similarity is above some threshold. This representation yields two results: The partitions contained in the graph

(that is those sub-graphs that are unconnected to the other sub graphs), form distinct topics covered in the documents. This allows a choice of coverage in the summary. For query-specific summaries, sentences may be selected only from the pertinent sub graph, while for generic summaries, representative sentences may be chosen from each of the sub-graphs.

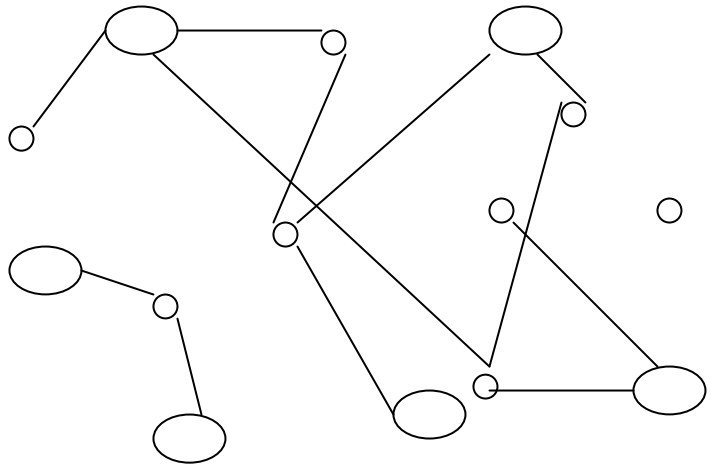


Figure 2: Graph Theoretic approach

The second result yielded by the graph-theoretic method is the identification of the important sentences in the document. The nodes with high cardinality (number of edges connected to that node), are the important sentences in the partition, and hence carry higher preference to be included in the summary. Figure2 shows an

example graph for a document. It can be seen that there are about 3-4 topics in the chapter; the nodes that are encircled can be seen to be informative sentences in the chapter, since they share information with many other sentences in the chapter. The graph theoretic method may also be adapted easily for visualization of inter- and intra-document similarity.

2.2.4 Machine Learning approach

In the 1990s, with the advent of machine learning techniques in NLP, a series of semi- nal publications appeared that employed statistical techniques to produce document extracts. While initially most systems assumed feature independence and relied on naive-Bayes methods, others have focused on the choice of appropriate features and on learning algorithms that make no independence assumptions. Other significant approaches involved hidden Markov models and log-linear models to improve extractive summarization. A very recent paper, in contrast, used neural networks and third party features (like common words in search engine queries) to improve purely extractive single document summarization. We next describe all these approaches in more detail.

2.2.4.1 Naive-Bayes Methods

Kupiec et al. (1995) describe a method derived from Edmundson (1969) that is able to learn from data. The classification function categorizes each sentence as worthy of extraction or not, using a naive-Bayes classifier. Let s be a particular sentence,
S the set of sentences that make up the summary, and F1, . . . , Fk the features. Assuming independence of the features:

$$P(s \in S \mid F1, F2, ..., FN) = P(F1, F2, ..., FN \mid s \in S) * P(s \in S) / P(F1, F2, ..., FN)$$

The features were compliant to (Edmundson, 1969), but additionally included the sentence length and the presence of uppercase words. Each sentence was given a score according to (1), and only the n top sentences were extracted. To evaluate the system, a corpus of technical documents with manual abstracts was used in the following way: for each sentence in the manual abstract, the authors manually analyzed its match with the actual document sentences and created a mapping (e.g. exact match with a sentence,

matching a join of two sentences, not matchable, etc.). The auto-extracts were then evaluated against this mapping. Feature analysis revealed that a system using only the position and the cue features, along with the sentence length sentence feature, performed best.

Aone et al. (1999) also incorporated a naive-Bayes classifier, but with richer features. They describe a system called DimSum that made use of features like term frequency (tf) and inverse document frequency (idf) to derive signature words.4

The idf was computed from a large corpus of the same domain as the concerned documents. Statistically derived two-noun word collocations were used as units for counting, along with single words. A named-entity tagger was used and each entity was considered as a single token. They also employed some shallow discourse analysis like reference to same entities in the text, maintaining cohesion. The references were resolved at a very shallow level by linking name aliases within a document like "U.S." to "United States", or "IBM" for "International Business Machines". Synonyms and morphological variants were also merged while considering lexical terms, the former being identified by using Wordnet (Miller, 1995). The corpora used in the experiments were from newswire, some of which belonged to the TREC evaluations.

2.2.4.2 Rich Features and Decision Trees

Lin and Hovy (1997) studied the importance of a single feature, sentence position. Just weighing a sentence by its position in text, which the authors term as the "position method", arises from the idea that texts generally follow a predictable discourse structure, and that the sentences of greater topic centrality tend to occur in certain specifiable locations (e.g. title, abstracts, etc). However, since the discourse structure significantly varies over domains, the position method cannot be defined as naively as in (Baxendale, 1958). The paper makes an important contribution by investigating techniques of tailoring the position method towards optimality over a genre and how it can be evaluated for effectiveness. A newswire corpus was used, the collection of Ziff-Davis texts produced from the TIPSTER5 program; it consists of text about computer and related hardware, accompanied by a set of key topic words and a small abstract of six

sentences. For each document in the corpus, the authors measured the yield of each sentence position against the topic keywords. They then ranked the sentence positions by their average yield to produce the Optimal Position Policy (OPP) for topic positions for the genre. Two kinds of evaluation were performed. Previously unseen text was used for testing whether the same procedure would work in a different domain. The first evaluation showed contours exactly like the training documents. In the second evaluation, word overlap of manual abstracts with the extracted sentences was measured. Windows in abstracts were compared with windows on the selected sentences and corresponding precision and recall values were measured. A high degree of coverage indicated the effectiveness of the position method.

In later work, Lin (1999) broke away from the assumption that features are independent of each other and tried to model the problem of sentence extraction using decision trees, instead of a naive-Bayes classifier. He examined a lot of features and their effect on sentence extraction. The data used in this work is a publicly available collection of texts.

2.2.5 LSA Method

Singular Value Decomposition (SVD) is a very powerful mathematical tool that can find principal orthogonal dimensions of multidimensional data. It has applications in many areas and is known by different names: Karhunen-Loeve Transform in image processing, Principal Component Analysis (PCA) in signal processes and Latent Semantic Analysis (LSA) in text processing. It gets this name LSA because SVD applied to document- word matrices, groups documents that are semantically related to each other, even when they do not share common words.

Words that usually occur in related contexts are also related in the same singular space. This method can be applied to extract the topic-words and content-sentences from documents. The advantage of using LSA vectors for summarization rather than the word vectors is that conceptual (or semantic) relations as represented in human brain are automatically captured in the LSA, while using word vectors without the LSA transformation requires design of explicit methods to derive conceptual relations. Since SVD finds principal and mutually orthogonal dimensions of the sentence vectors, picking out a representative sentence from each of the dimensions

ensures relevance to the document, and orthogonality ensures non-redundancy. It is to be noted that this property applies only to data that has principal dimensions inherently—however, LSA would probably work since most of the text data has such principal dimensions owing to the variety of topics it addresses.

2.2.6 An approach to concept-obtained text summarization

The idea of this approach is to obtain concepts of words based on HowNet and use concept as feature, instead of word. This approach uses conceptual vector space model to form a rough summarization, and then calculate degree of semantic similarity of sentence for reducing its redundancy. A good summary system should extract the diverse topics of the document while keeping redundancy to a minimum. This method consists of the following three main stages:

Stage 1: Using Hownet as tool to obtain concept of text, and establishing conceptual vector space model.

Stage 2: Calculate importance of concept based on conceptual vector space model.

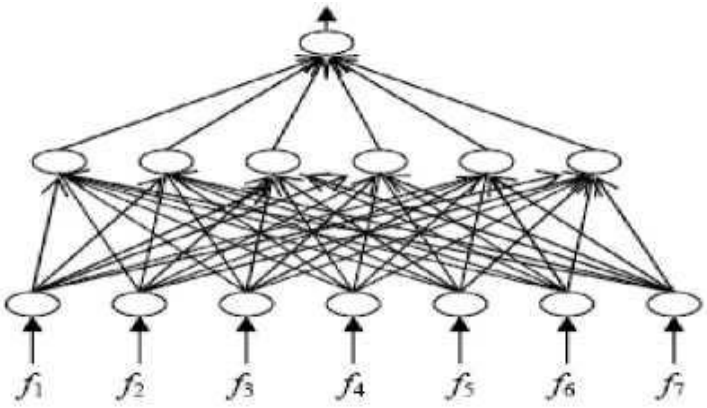
Stage 3: Generate the final summary by calculating importance of sentence and reducing the redundancy of summarization.

2.2.7 Text summarization with neural networks

This method involves training the neural networks to learn the types of sentences that should be included in the summary. This is accomplished by training the network with sentences in several test paragraphs where each sentence is identified as to whether it should be included in the summary or not. This is done by a human reader. The neural network learns the patterns inherent in sentences that should be included in the summary and those that should not be included. It uses three-layered Feed forward neural network, which has been proven to be a universal function approximator.

The first phase of the process involves training the neural networks to learn the types of sentences that should be included in the summary. This is accomplished by training the network with sentences in several test paragraphs where each sentence is identified as to whether it should be included in the summary or not. This is done by a human reader. The neural network learns the patterns inherent in

sentences that should be included in the summary and those that should not be included. The Neural Network after Training is shown in figure3.



Once the network has learned the features that must exist in summary sentences, we need to discover the trends and relationships among the features that are inherent in the

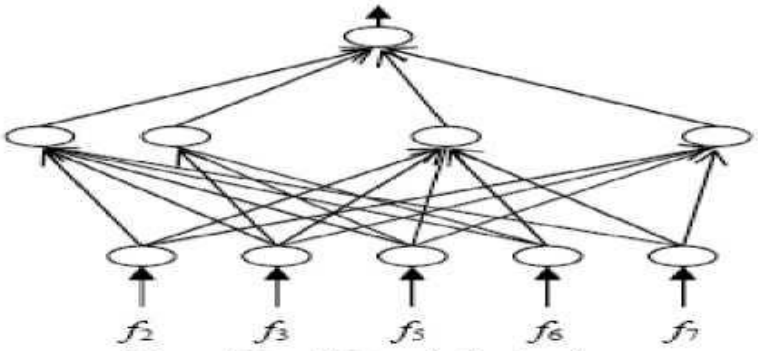


Figure 4. Neural Network after Pruning

majority of sentences. This is accomplished by the feature fusion phase, which consists of two steps: 1) eliminating uncommon features; and 2) collapsing the effects of common features. The connections having very small weights after training can be pruned without affecting the performance of the network. As a result, any input or hidden layer neuron having no emanating connections can be safely removed from the network. In addition, any hidden layer neuron having no abutting connections can be removed. This corresponds to eliminating uncommon features from the network [4] as shown in figure4.

The hidden layer activation values for each hidden layer neuron are clustered utilizing 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, which the activation value belongs to. This corresponds to collapsing the effects of common features. The combination of these two steps corresponds to generalizing the effects of features, as a whole, and providing control parameters for sentence ranking. The Neural Network [4] after feature fusion is shown in figure5.

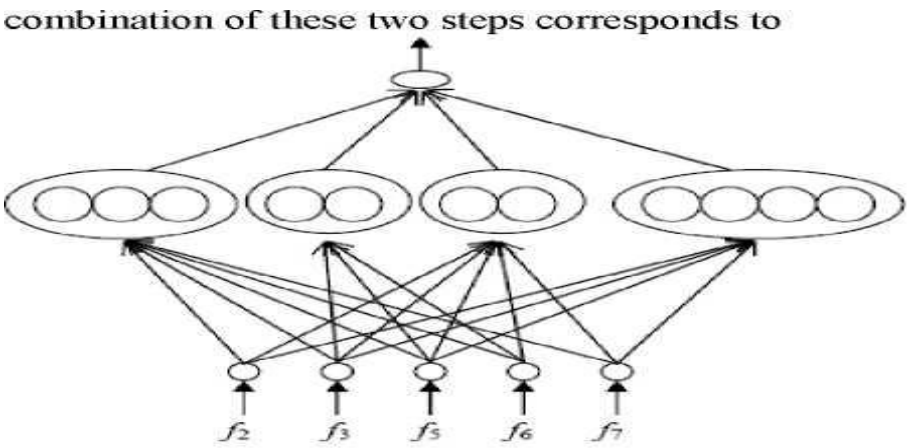


Figure 5. Neural Network after feature fusion

2.2.8 Query based extractive text summarization

In query based text summarization system, the sentences in a given document are scored based on the frequency counts of terms (words or phrases). The sentences containing the query phrases are given higher scores than the ones containing single query words. Then, the sentences with highest scores are incorporated into the output summary together with their structural context. Portions of text may be extracted from different sections or subsections. The resulting summary is the union of such extracts. The number of extracted sentences and the extent to which their context is displayed depends on the summary frame size which is fixed to the size of the screen that can be seen without scrolling. In the sentence extraction algorithm, whenever a sentence is selected for the inclusion in the summary, some of the headings in that context are also selected. The query based sentence extraction algorithm is as follows:

Algorithm:

1: Rank all the sentences according to their score.
2: Add the main title of the document to the summary.
3: Add the first level-1 heading to the summary.
4: While (summary size limit not exceeded)
5: Add the next highest scored sentence.

6: Add the structural context of the sentence:(if any and not already included in the summary)

7: Add the highest level heading above the extracted text (call this heading h).

8: Add the heading before h in the same level.

9: Add the heading after h in the same level.

10: Repeat steps 7, 8 and 9 for the next highest level headings.

11: End while

An another query-specific summarization [4] method views a document as a set of interconnected text fragments (passages) and focuses on keyword queries,since keyword search is the most popular information discovery method on documents, because of its power and ease of use. Firstly, at the preprocessing stage, it adds structure to every document, which can then be viewed as a labeled, weighted graph, called the document graph. Then, at query time, given a set of keywords, it performs keyword proximity search on the document graphs to discover how the keywords are associated in the document graphs. For each document its summary is the minimum spanning tree on the corresponding document graph that contains all the keywords. In query-specific opinion summarization system (QOS), When input an opinion question, the system returns a summary with relevance to the opinion and target described by the question. The system has several modules to be able to do 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. Bayesian summarization (BAYESUM) is a model for sentence extraction in query-focused summarization. BAYESUM leverages the common case in which multiple documents are relevant to a single query. Using these documents as reinforcement for query terms, BAYESUM is not afflicted by the paucity of information in short queries. For a collection of D documents and Q queries, assume a $D \times Q$ binary matrix r, where rdq = 1 if an only if document d is relevant to query q. In multi document summarization, rdq will be 1 exactly when d is in the document set corresponding to query q.

2.2.9 Multilingual Extractive Text summarization

Multilingual text summarization is to summarize the source text in different language to the target language final summary. SimFinderML identifies similar pieces of text by computing similarity over multiple features. There are two types of features, composite features, and unary features. All features are computed over primitives, syntactic, linguistic, or knowledge-based information units extracted from the sentences. Both composite and unary features are constructed over the primitives. The primitives used and features computed can be set at run-time, allowing for easy experimentation with different settings, and making it easy to add new features and primitives. Support for new languages is added to the system by developing modules conforming to interfaces for text pre-processing and primitive extraction for the language, and using existing dictionary-based translation methods, or adding other language-specific translation methods.

MINDS integrates multi-lingual summarization and multi document summarization capabilities using a multiengine, core summarization system and provides fast, interactive document access through hypertext summaries. Core summarization problem of MINDS is taking a single text and producing a shorter text in the same language that contains all the main points in the input text. It is using a robust, graded approach for building the core engine by incorporating statistical, syntactic and documents structure analyses among other techniques. This approach is less expensive and more robust than a summarization technique based entirely on a single method. The core engine is being designed in such a way that as additional resources, such as lexical and other knowledge bases or text processing and MT engines, become available from other ongoing research efforts they can be incorporated into the overall multiengine MINDS system. Ideally the core engine itself will remain language independent. A prototype core engine has been built for English, Spanish, Russian, and Japanese documents.

MEAD is the multi-lingual summarization and evaluation method. MEAD's architecture consists of four stages. First, documents in a cluster are converted to MEAD's internal (XML-based) format. Second, given a configuration file or command-line options, 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., repeated sentences,

chronological ordering, source preferences, etc.) In addition to a number of command-line utilities, MEAD provides a Perl API which lets external programs access its internal libraries.

2.2.10 Multi-document extractive summarization

Multi document extractive summarization deals with extraction of summarized information from multiple texts written about the same topic. Resulting summary report allows individual users, so as professional information consumers, to quickly familiarize themselves with information contained in a large cluster of documents. Multi-document summarization creates information reports that are both concise and comprehensive. With different opinions being put together & outlined, every topic is described from multiple perspectives within a single document.

NeATS is a multi-document summarization system that attempts to extract relevant or interesting portions from a set of documents about some topic and present them in coherent order. It is an extraction-based multi-document summarization system. Given an input of a collection of sets of newspaper articles, NeATS generates summaries in three stages: content selection, filtering, and presentation.

The goal of content selection is to identify 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 the off-topic document collection as the irrelevant set. 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 connection. Each sentence in the document set is then ranked, using the key concept structures.

NeATS uses three different filters: sentence position, stigma words, and maximum marginal relevancy. Sentence position is a good content filter, that only retains the leading 10 sentences. Some sentences start with stigma words like:

- Conjunctions (e.g., but, although, however)
- The verb *say* and its derivatives

- Quotation marks
- Pronouns such as he, she, and they

usually cause discontinuity in summaries. The scores of these sentences are reduced to avoid including them in short summaries. Redundancy issue is addressed in maximum marginal relevancy filter. 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 set empirically.

Hub/Authority framework is multi document summarization system which, firstly detect the sub-topics in multi-documents by sentence clustering and extract the feature words (or phrase) of different sub-topics. Secondly, all feature words and the cue phrases are used as the vertex of Hub and all sentences are regarded as the vertex of Authority. If the sentence contains the words in Hub, there is an edge between the Hub word and the Authority sentence. The initial weight of each vertex considers both the content and the cues such as cue phrase and first sentence. Through the mutual reinforcement mechanism of the Hub-Authority algorithm, we can rank the importance of the sentences within the multi-documents. The assumption behind this cue-based Hub/Authority approach is that a good Hub word (or phrase) is the content that points to many good authorities sentences and a good authority sentence is a vertex that is pointed to by many good hub words. Thirdly, It has used the Markov Model to order the subtopics that the final summarization should contain and output the text summarization according to the sentence ranking score of all sentences within one sub-topic as user' requirement.

Generic relation extraction (GRE) is a novel multi document text summarization approach, which aims to build systems for relation identification and characterization that can be transferred across domains and tasks without modification of model param.

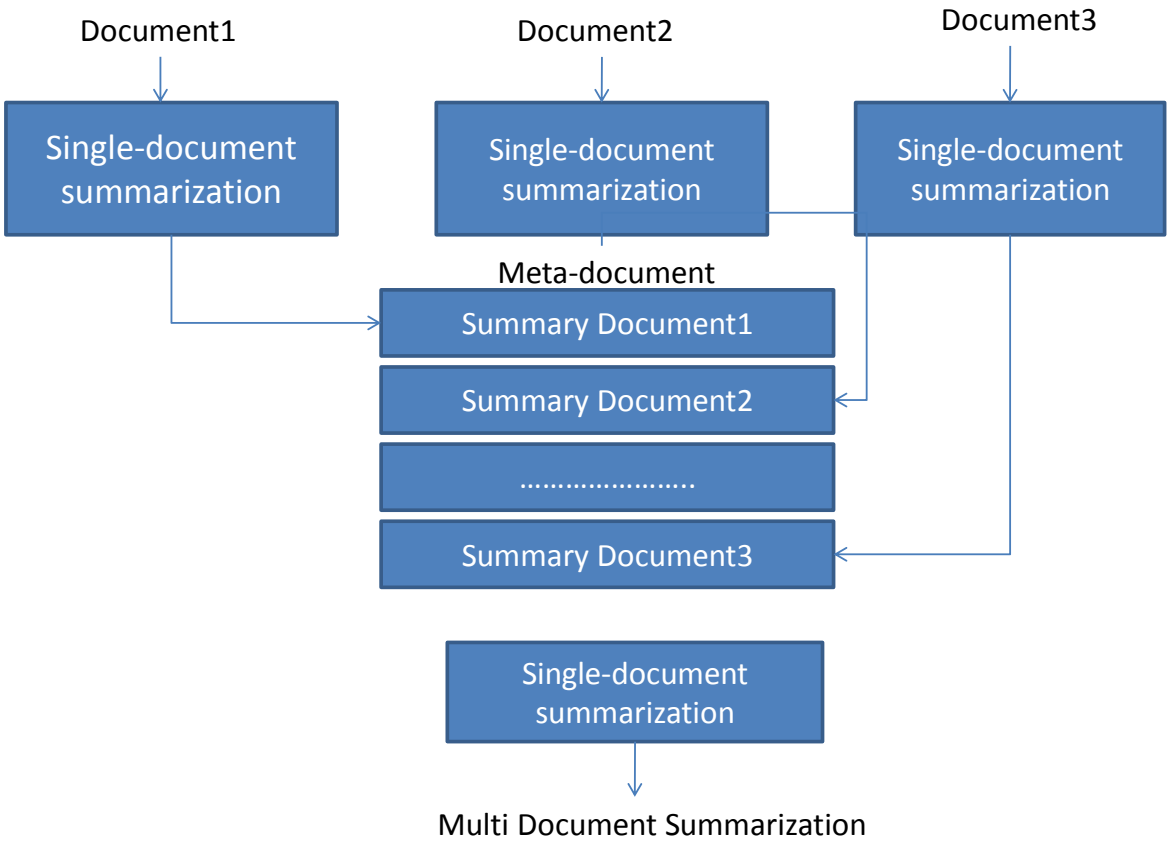


Figure 5. Multi Document Summarisation

CHAPTER 3

A REVIEW OF TEXT SUMMARIZATION

An automatic summarization process can be divided into three steps: (1) in the *preprocessing step* a structured representation of the original text is obtained; (2) in the *processing step* an algorithm must transform the text structure into a summary structure; and (3) in the *generation step* the final summary is obtained from the summary structure.

The methods of summarization can be classified, in terms of the level in the linguistic space, in two broad groups: (a) shallow approaches, which are restricted to the syntactic level of representation and try to extract salient parts of the text in a convenient way; and (b) deeper approaches, which assume a semantics level of representation of the original text and involve linguistic processing at some level.

In the first approach the aim of the preprocessing step is to reduce the dimensionality of the representation space, and it normally includes: (i) *stop-word* elimination - common words with no semantics and which do not aggregate relevant information to the task (e.g., “the”, “a”) are eliminated; (ii) *case folding:* consists of converting all the characters to the same kind of letter case - either upper case or lower case; (iii) *stemming:* syntactically-similar words, such as plurals, verbal variations, etc. are considered similar; the purpose of this procedure is to obtain the stem or radix of each word, which emphasize its semantics.

A frequently employed text model is the vectorial model . After the preprocessing step each text element - a sentence in the case of text summarization - is considered as a A-dimensional vector. So it is possible to use some metric in this space to measure similarity between text elements. The most employed metric is the cosine measure, defined as $\cos d = (\langle x.y \rangle) / (|x| . |y|)$ for vectors x and y, where $(\langle,\rangle)$ indicates the scalar product, and $|x|$ indicates the module of x. Therefore maximum similarity corresponds to $\cos d = 1$, whereas $\cos d = 0$ indicates total discrepancy between the text elements.

The evaluation of the quality of a generated summary is a key point in summarization research. A detailed evaluation of summarizers was made at the TIPSTER Text

Summarization Evaluation Conference (SUMMAC), as part of an effort to standardize summarization test procedures. In this case a reference summary collection was provided by human judges, allowing a direct comparison of the performance of the systems that participated in the conference. The human effort to elaborate such summaries, however, is huge. Another reported problem is that even in the case of human judges, there is low concordance: only 46 % according to Mitra; and more importantly: the summaries produced by the same human judge in different dates have an agreement of only 55 % .

The idea of a "reference summary" is important, because if we consider its existence we can objectively evaluate the performance of automatic summary generation procedures using the classical Information Retrieval (IR) *precision* and *recall* measures. In this case a sentence will be called *correct* if it belongs to the reference summary. As usual, *precision* is the ratio of the number of selected correct sentences over the total number of selected sentences, and *recall* is the ratio of the number of selected correct sentences over the total number of correct sentences. In the case of fixed-length summaries the two measures are identical, since the sizes of the reference and the automatically obtained extractive summaries are identical.

Mani and Bloedorn proposed an automatic procedure to generate reference summaries: if each original text contains an author-provided summary, the corresponding size-K reference extractive summary consists of the *K* most similar sentences to the author-provided summary, according to the cosine measure. Using this approach it is easy to obtain reference summaries, even for big document collections.

A Machine Learning (ML) approach can be envisaged if we have a collection of documents and their corresponding reference extractive summaries. A trainable summarizer can be obtained by the application of a classical (trainable) machine learning algorithm in the collection of documents and its summaries. In this case the sentences of each document are modeled as vectors of features extracted from the text. The summarization task can be seen as a two-class classification problem, where a sentence is labeled as "correct" if it belongs to the extractive reference summary, or as "incorrect" otherwise. The trainable summarizer is expected to "learn" the patterns which lead to the summaries, by identifying relevant feature values which are most correlated with the classes "correct" or "incorrect". When a new document is given to the system, the

"learned" patterns are used to classify each sentence of that document into either a "correct" or "incorrect" sentence, producing an extractive summary. A crucial issue in this framework is how to obtain the relevant set of features; the next section treats this point in more detail.

To understand how a summarizer works various tools were studied and experimented with. The tools studied were as follows:

1. **Text compactor**: Text compactor is an Online summarizer for English language. It involves three steps. In step1 the user Types or pastes text into the box. In step2 the user Drags the slider, or enters a number in the box, to set the percentage of text to keep in the summary. In step3 the user is presented with the summarized text. The tool does not support any other language besides English.

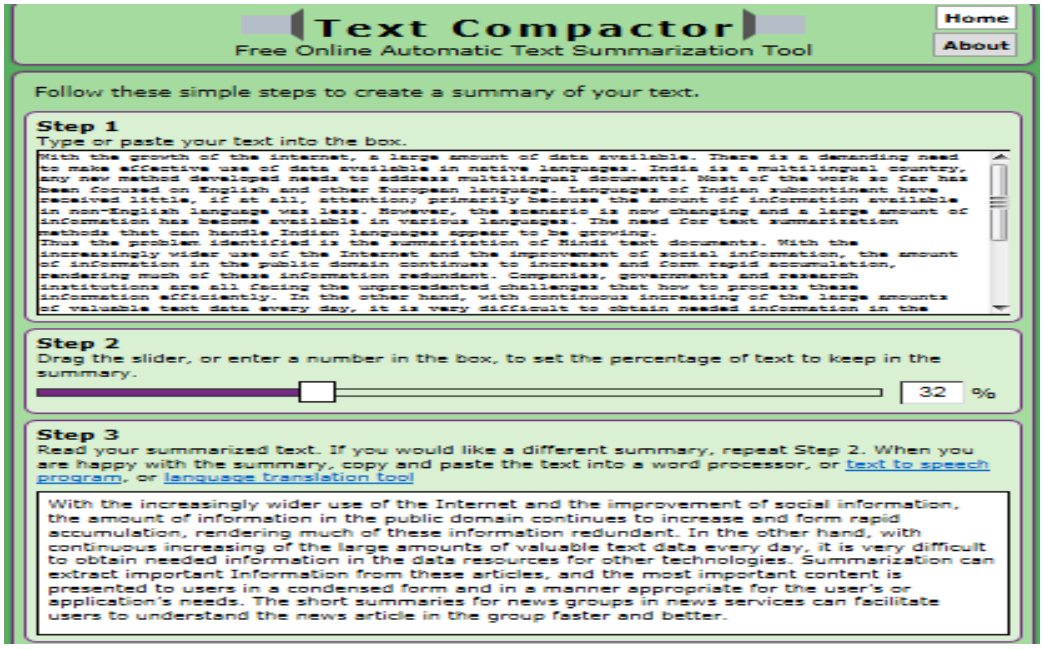


Fig.6 Text Comparator

2. **Online Tool Summarizer:** The tool generates the summary based on the threshold set by the user. The user can also set the Minimum sentence length. In the output along with the summary the user is also presented with the best words found in the text entered along with the frequency of occurrence of the best word. Also the system does not support Hindi.

3. **Open Text Summarizer:** Open Text Summarizer is an open source tool for summarizing texts. The program reads a text and decides which sentences are important and which are not. In this approach, keywords are identified by means of word occurrence, and sentences are given a score based on the keywords they contain It ships with Ubuntu, Fedora and other linux distros. OTS supports many (25+) languages which are configured .Several academic publications have benchmarked it and praised it. OTS is both a library and a command line tool. Word processors such as AbiWord and KWord can link to the library and summarize documents while the command line tool lets you summarize text on the console. The program can either print the summarized text as text or HTML. If HTML, the important sentences are highlighted. The program is multi lingual. The Open Text Summarizer summarizes texts in English, German, Spanish, Russian, Hebrew, Esperanto and other languages.

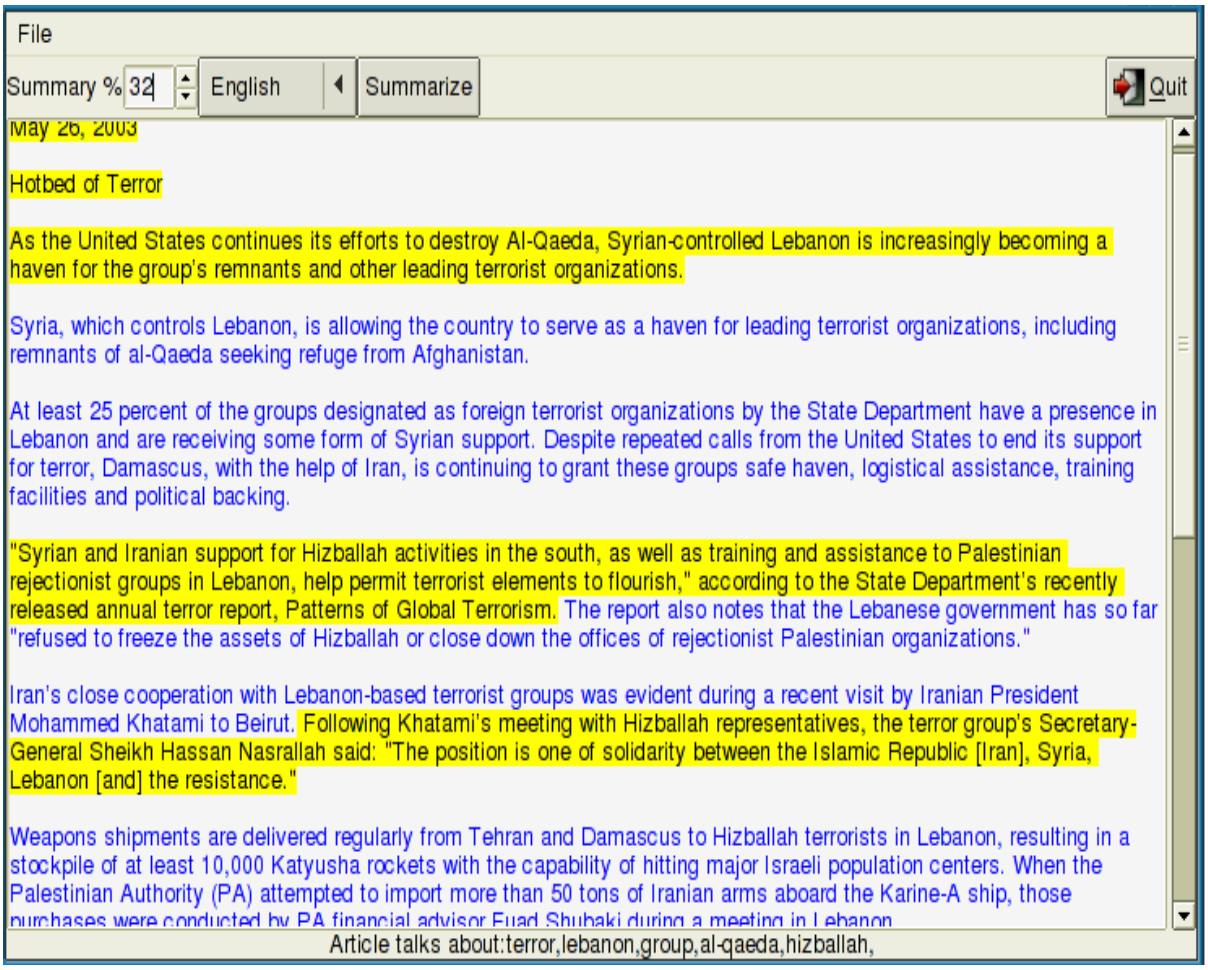


Fig.7 Open text summariser

CHAPTER 4

DESIGN AND IMPLEMENTATION

I have made automatic summarization with the machine learning approach with the following way:

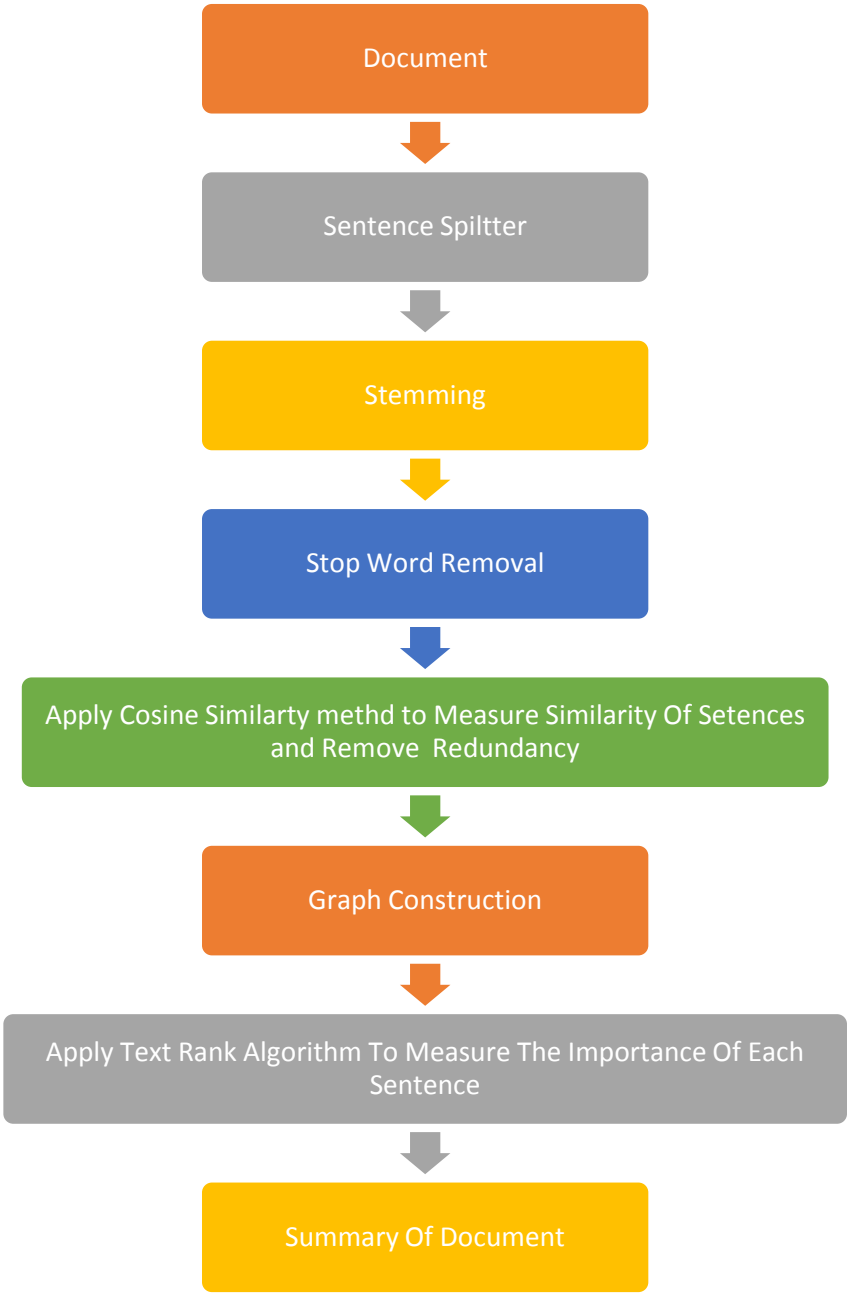


Figure8. Text Summarization System Architecture

4.1 Document Preprocessing

The original document will contain many words, which will not be important in the sentence. It would be obvious to remove the stop words such as “A”, “the” etc. from the sentence.

There are many more sophisticated methods to process the documents, such as retrieving only nouns or nouns and adverbs etc.

This report has two main points: (1) the set of employed features; and (2) the framework defined for the trainable summarizer, including the employed classifiers.

A large variety of features can be found in the text-summarization literature. In my proposal we employ the following set of features:

4.1.1 Mean-TF-IDF: Since the seminal work of Luhn, text processing tasks frequently use features based on IR measures . In the context of IR, some very important measures are term frequency (TF) and term frequency x inverse document frequency (TF-IDF) . In text summarization we can employ the same idea: in this case we have a single document d, and we have to select a set of relevant sentences to be included in the extractive summary out of all sentences in d. Hence, the notion of a collection of documents in IR can be replaced by the notion of a single document in text summarization. Analogously the notion of document - an element of a collection of documents - in IR, corresponds to the notion of sentence - an element of a document - in summarization. This new measure will be called term frequency x inverse sentence frequency, and denoted TF-ISF(w,s).The final used feature is calculated as the mean value of the TF-ISF measure for all the words of each sentence.

(a) Sentence Length. This feature is employed to penalize sentences that are too short, since these sentences are not expected to belong to the summary. We use the normalized length of the sentence, which is the ratio of the number of words occurring in the sentence over the number of words occurring in the longest sentence of the document.

(b) Sentence Position. This feature can involve several items, such as the position of a sentence in the document as a whole, its the position in a section, in a paragraph, etc., and has presented good results in several research projects We use here the percentile of the sentence position in the document, the final value is normalized to take on values between 0 and 1.

(c) Similarity to Title. According to the vectorial model, this feature is obtained by using the title of the document as a "query" against all the sentences of the document; then the similarity of the document's title and each sentence is computed by the cosine similarity measure .

(d) Similarity to Keywords. This feature is obtained analogously to the previous one, considering the similarity between the set of keywords of the document and each sentence which compose the document, according to the cosine similarity.

For the next two features we employ the concept of text cohesion. Its basic principle is that sentences with higher degree of cohesion are more relevant and should be selected to be included in the summary.

(e) Sentence-to-Sentence Cohesion. This feature is obtained as follows: for each sentence s we first compute the similarity between s and each other sentence s' of the document; then we add up those similarity values, obtaining the raw value of this feature for *s*; the process is repeated for all sentences. The normalized value (in the range [0, 1]) of this feature for a sentence *s* is obtained by computing the ratio of the raw feature value for *s* over the largest raw feature value among all sentences in the document. Values closer to 1.0 indicate sentences with larger cohesion.

(f) Sentence-to-Centroid Cohesion. This feature is obtained for a sentence *s* as follows: first, we compute the vector representing the centroid of the document, which is the arithmetic average over the corresponding coordinate values of all the sentences of the

document; then we compute the similarity between the centroid and each sentence, obtaining the raw value of this feature for each sentence. The normalized value in the range [0, 1] for *s* is obtained by computing the ratio of the raw feature value over the largest raw feature value among all sentences in the document. Sentences with feature values closer to 1.0 have a larger degree of cohesion with respect to the centroid of the document, and so are supposed to better represent the basic ideas of the document.

For the next features an approximate argumentative structure of the text is employed. It is a consensus that the generation and analysis of the complete rethorical structure of a text would be impossible at the current state of the art in text processing. In spite of this, some methods based on a surface structure of the text have been used to obtain good-quality summaries. To obtain this approximate structure we first apply to the text an agglomerative clustering algorithm. The basic idea of this procedure is that similar sentences must be grouped together, in a bottom-up fashion, based on their lexical similarity. As result a hierarchical tree is produced, whose root represents the entire document. This tree is binary, since at each step two clusters are grouped. Five features are extracted from this tree, as follows:

(g) Depth in the tree. This feature for a sentence *s* is the depth of *s* in the tree.

(h) Referring position in a given level of the tree (positions 1, 2, 3, and 4). We first identify the path form the root of the tree to the node containing *s*, for the first four depth levels. For each depth level, a feature is assigned, according to the direction to be taken in order to follow the path from the root to *s*; since the argumentative tree is binary, the possible values for each position are: left, right and none, the latter indicates that *s* is in a tree node having a depth lower than four.

(i) Indicator of main concepts. This is a binary feature, indicating whether or not a sentence captures the main concepts of the document. These main concepts are obtained by assuming that most of relevant words are nouns. Hence, for each sentence, we identify its nouns using a part-of-speech software . For each noun we then compute the number of sentences in which it occurs. The fifteen nouns with largest occurrence are selected as

being the main concepts of the text. Finally, for each sentence the value of this feature is considered “true” if the sentence contains at least one of those nouns, and “false” otherwise.

(j) Occurrence of proper names. The motivation for this feature is that the occurrence of proper names, referring to people and places, are clues that a sentence is relevant for the summary. This is considered here as a binary feature, indicating whether a sentence *s* contains (value “true”) at least one proper name or not (value “false”). Proper names were detected by a part-of-speech tagger.

(k) Occurrence of anaphors. We consider that anaphors indicate the presence of non-essential information in a text: if a sentence contains an anaphor, its information content is covered by the related sentence. The detection of anaphors was performed in a way similar to the one proposed by Strzalkowski we determine whether or not certain words, which characterize an anaphor, occur in the first six words of a sentence. This is also a binary feature, taking on the value “true” if the sentence contains at least one anaphor, and “false” otherwise.

(l) Occurrence of non-essential information. We consider that some words are indicators of non-essential information. These words are speech markers such as “because”, “furthermore”, and “additionally”, and typically occur in the beginning of a sentence. This is also a binary feature, taking on the value “true” if the sentence contains at least one of these discourse markers, and “false” otherwise.

The ML-based trainable summarization framework consists of the following steps:

1. We apply some standard preprocessing information retrieval methods to each document, namely stop-word removal, case folding and stemming. We have employed the stemming algorithm proposed by Porter .
2. All the sentences are converted to its vectorial representation .
3. We compute the set of features described in the previous subsection. Continuous features are discretized: we adopt a simple “class-blind” method, which consists of separating the original values into equal-width intervals. We did some experiments with

different discretization methods, but surprisingly the selected method, although simple, has produced better results in our experiments.

4. A ML trainable algorithm is employed; we employ two classical algorithms, namely Text Rank Algorithm and Hits Algorithm. As usual in the ML literature, we employ these algorithms trained on a training set and evaluated on a separate test set.

The framework assumes, of course, that each document in the collection has a reference extractive summary. The "correct" sentences belonging to the automatically produced extractive summary are labeled as "positive" in classification/data mining terminology, whereas the remaining sentences are labeled as "negative".

4.1.2 Graph Construction

- We then construct a graph from the words that we have filtered from each sentence.
- We assign each sentence a node and calculate the edge weight between each sentence by various similarity functions.
- There are two methods to calculate similarity between sentences.
    - o One is to calculate sentence similarity with the words in sentences.
    - o The other is to calculate sentence similarity with stemmed words in sentences.
- We have chosen the original words in the sentences, because the stemmed words would miss the tense and voice information of the sentence.
- Moreover, there are many different Similarity functions, which can be used, with each having its own benefits and drawbacks.

4.1.3 Ranking Algorithms

Once we have obtained a graph with edges, we can apply various techniques, such as HITS or TextRank, to obtain weight for each node. Using these ranking algorithms, we can obtain a weight or importance for each node.

4.1.4 Summarization

Once we have each sentence and a measure of its importance, we can sort the node in the order of their weights and display the sentences with most similarity.

4.2 Features used

TO evaluate how to improve the summaries , I will be using the following features,

- Nouns and verbs
- Nouns, adjectives and Verbs
- Nouns and adjectives

I will be using the following similarity function [6] to generate different summaries:

4.2.1Jaccard Similarity

a.)It is a statistic used for comparing the similarity and diversity of sample sets.It uses overlap of words between sentences to calculate similarity.

b.)It uses overlap of words between sentences to calculate similarity.

4.2.2 Cosine Similarity

4.2.2.1 Vector Space Model:

1.Sentences are also treated as a "bag" of words or terms.

2.Each sentence is represented as a vector.

3.However, the term weights are no longer 0 or 1. Each term weight is computed based on some variations of TF or TF-IDF scheme.

4.Term Frequency (TF) Scheme: The weight of a term ti in document dj is the number of times that ti appears in dj, denoted by fij. Normalization may also be applied.

$$\text{similarity} = \cos(\theta) = \frac{A \,.\, B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i * B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} * \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

4.2.2.2 TF-IDF term weighting scheme:

TF: still term frequency,IDF: inverse document frequency.

The most well known weighting scheme

$$tf_{ij} = \frac{f_{ij}}{\max\{f_{1j}, f_{2j}, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots, f_{|V|j}\}}$$

$$\boldsymbol{idf_i = log \frac{N}{df_i}}$$

N: total number of docs

dfi: the number of docs that ti appears.

- The final TF-IDF term weight is:

$$w_{ij} = tf_{ij} * idf_i$$

Retrieval in vector space model:
Query q is represented in the same way or slightly differently.

Relevance of di to q: Compare the similarity of query q and document di.

Cosine similarity (the cosine of the angle between the two vectors)

CHAPTER 5

RANKING ALGORITHMS

We will be using the 3 features along with the 3 syntactic filtering mentioned above to obtain different graphs, on which we various ranking algorithms can be applied to obtain the sentence weights (importance).

5.1 TextRank

Graph-based ranking algorithms are essentially a way of deciding the importance of a vertex within a graph, based on information drawn from the graph structure. The basic idea implemented by a graph-based ranking model is that of "voting" or "recommendation". When one vertex links to another one, it is basically casting a vote for that other vertex. The higher the number of votes that are cast for a vertex, the higher the importance of the vertex. The score associated with a vertex is determined based on the votes that are cast for it, and the score of the vertices casting these votes.To enable the application of graph-based ranking algorithms to natural language texts, we have to build a graph that represents the text, and interconnects words or other text entities with meaningful relations. Depending on the application at hand, text units of various sizes and characteristics can be added as vertices in the graph, e.g. words, collocations, entire sentences, or others. Similarly, it is the application that dictates the type of relations that are used to draw connections between any two such vertices, e.g. lexical or semantic relations, contextual overlap, etc.

Regardless of the type and characteristics of the elements added to the graph, the application of graph-based ranking algorithms to natural language texts consists of the following main steps:

1.Identify text units that best define the task at hand, and add them as vertices in the graph.

2. Identify relations that connect such text units, and use these relations to draw edges

between vertices in the graph. Edges can be directed or undirected, weighted or unweighted.

3.Iterate the graph-based ranking algorithm until convergence.

4. Sort vertices based on their final score. Use the values attached to each vertex for ranking/selection decisions

TextRank does not require deep linguistic knowledge, nor domain or language specific annotated corpora, which makes it highly portable to other domains, genres, or languages.

Graph-based ranking algorithms like Kleinberg's HITS algorithm (Kleinberg, 1999) or Google's PageRank (Brin and Page, 1998) have been success- fully used in citation analysis, social networks, and the analysis of the link-structure of the World Wide Web. Arguably, these algorithms can be singled out as key elements of the paradigm-shift triggered in the field of Web search technology, by providing a Web page ranking mechanism that relies on the collective knowledge of Web architects rather than individual content analysis of Web pages. In short, a graph-based ranking algorithm is a way of deciding on the importance of a vertex within a graph, by taking into account global information recursively computed from the entire graph, rather than relying only on local vertex-specific information.

Applying a similar line of thinking to lexical or semantic graphs extracted from natural language documents, results in a graph-based ranking model that can be applied to a variety of natural language processing applications, where knowledge drawn from an entire text is used in making local ranking/selection decisions. Such text oriented ranking methods can be applied to tasks ranging from auto- mated extraction of keyphrases, to extractive summarization and word sense disambiguation (Mihalcea et al., 2004).

In this paper, we introduce the TextRank graph- based ranking model for graphs extracted from natural language texts. We investigate and evaluate the application of TextRank to two language processing tasks consisting of unsupervised keyword and sentence

extraction, and show that the results obtained with TextRank are competitive with state-of-the-art systems developed in these areas.

5.1.1 The TextRank Model

Graph-based ranking algorithms are essentially a way of deciding the importance of a vertex within a graph, based on global information recursively drawn from the entire graph. The basic idea implemented by a graph-based ranking model is that of "voting" or "recommendation". When one vertex links to another one, it is basically casting a vote for that other vertex. The higher the number of votes that are cast for a vertex, the higher the importance of the vertex. Moreover, the importance of the vertex casting the vote determines how important the vote itself is, and this information is also taken into account by the ranking model. Hence, the score associated with a vertex is determined based on the votes that are cast for it, and the score of the vertices casting these votes.

Formally, let *G* = (V, *E)* be a directed graph with the set of vertices U and set of edges !?, where I? is a subset o f U x U . For a given vertex U, let In(U) be the set of vertices that point to it (predecessors), and let *Out (Vi)* be the set of vertices that vertex V* points to (successors). The score of a vertex V) is defined as follows (Brin and Page, 1998):

$$WS(V_i) = (1-d) + d * \sum_{V_j \varepsilon \ln(v_i)} \frac{w_{ji}}{\sum_{v_k \varepsilon out(v_j)} w_{jk}} * WS(V_j)$$

where d is a damping factor that can be set between 0 and 1, which has the role of integrating into the model the probability of jumping from a given vertex to another random vertex in the graph. In the context of Web surfing, this graph-based ranking algorithm implements the "random surfer model", where a user clicks on links at random with a probability *d,* and jumps to a completely new page with probability 1 — *d.* The factor d is usually set to 0.85 (Brin and Page, 1998), and this is the value we are also using in our implementation.

Starting from arbitrary values assigned to each node in the graph, the computation iterates until convergence below a given threshold is achieved [1]. After running the algorithm, a

score is associated with each vertex, which represents the "importance" of the vertex within the graph. Notice that the final values obtained after TextRank runs to completion are not affected by the choice of the initial value, only the number of iterations to convergence may be different.

It is important to notice that although the TextRank applications described in this paper rely on an algorithm derived from Google's PageRank (Brin and Page, 1998), other graph-based ranking algorithms such as e.g. HITS (Kleinberg, 1999) or Positional Function (Herings et al., 2001) can be easily integrated into the TextRank model (Mihalcea, 2004).

5.1.1.1 Undirected Graphs

Although traditionally applied on directed graphs, a recursive graph-based ranking algorithm can be also applied to undirected graphs, in which case the out- degree of a vertex is equal to the in-degree of the vertex. For loosely connected graphs, with the number of edges proportional with the number of vertices, undirected graphs tend to have more gradual convergence curves.

5.1.1.2 Weighted Graphs

In the context of Web surfing, it is unusual for a page to include multiple or partial links to another page, and hence the original PageRank definition for graph-based ranking is assuming unweighted graphs.

However, in our model the graphs are build from natural language texts, and may include multiple or partial links between the units (vertices) that are extracted from text. It may be therefore useful to indicate and incorporate into the model the "strength" of the connection between two vertices V) and V) as a *weight* $\boldsymbol{Wij}$ added to the corresponding edge that connects the two vertices.

5.1.1.3 Text as a Graph

To enable the application of graph-based ranking algorithms to natural language texts, we have to build a graph that represents the text, and interconnects words

or other text entities with meaningful relations. Depending on the application at hand, text units of various sizes and characteristics can be added as vertices in the graph, e.g. words, collocations, entire sentences, or others. Similarly, it is the application that dictates the type of relations that are used to draw connections between any two such vertices, e.g. lexical or semantic relations, contextual overlap, etc.

Regardless of the type and characteristics of the elements added to the graph, the application of graph- based ranking algorithms to natural language texts consists of the following main steps:

1. Identify text units that best define the task at hand, and add them as vertices in the graph.
2. Identify relations that connect such text units, and use these relations to draw edges between vertices in the graph. Edges can be directed or undirected, weighted or unweighted.
3. Iterate the graph-based ranking algorithm until convergence.
4. Sort vertices based on their final score. Use the values attached to each vertexfor ranking/selection decisions.

5.1.1.4 TEXT RANK FOR KEYWORD EXTRACTION

The expected end result for this application is a set of words or phrases that are representative for a give natural language text. The units to be ranked are therefore sequences of one or more lexical units extracted from text, and these represent the vertices that are added to the text graph. Any relation that can be defined between two lexical units is a potentially useful connection (edge) that can be added between two such vertices. We are using a *co-occurrence* relation, controlled by the distance between word occurrences: two vertices are connected if their corresponding lexical units co-occur within a window of maximum words, where can be set anywhere from 2 to 10 words. Co-occurrence links express relations between syntactic elements, and similar to the semantic links found useful for the task of word sense disambiguation, they represent cohesion indicators for a given text. The vertices added to the graph can be restricted with syntactic

filters, which select only lexical units of a certain part of speech. One can for instance consider only nouns and verbs for addition to the graph, and consequently draw potential edges based only on relations that can be established between nouns and verbs. We experimented with various syntactic filters, including: all open class words, nouns and verbs only, etc., with best results observed for nouns and adjectives only.

The TextRank keyword extraction algorithm is fully unsupervised, and proceeds as follows. First,the number of keywords based on the size of the text. For the data used in our experiments, which consists of relatively short abstracts, is set to a third of the number of vertices in the graph. During post-processing, all lexical units selected as potential keywords by the TextRank algorithm are marked in the text, and sequences of adjacent keywords are collapsed into a multi-word keyword.

5.1.1.5 TEXT RANK FOR SENTENCE EXTRACTION

The other TextRank application that we investigate consists of sentence extraction for automatic summarization. In a way, the problem of sentence extraction can be regarded as similar to keyword extraction, since both applications aim at identifying sequences that are more "representative" for the given text. In keyword extraction, the candidate text units consist of words or phrases, whereas in sentence extraction, we deal with entire sentences. TextRank turns out to be well suited for this type of applications, since it allows for a ranking over text units that is recursively computed based on information drawn from the entire text.

To apply TextRank, we first need to build a graph associated with the text, where the graph vertices are representative for the units to be ranked. For the task of sentence extraction, the goal is to rank entire sentences, and therefore a vertex is added to the graph for each sentence in the text. The co-occurrence relation used for keyword extraction cannot be applied here, since the text units in consideration are significantly larger than one or few words, and "co-occurrence" is not a meaningful relation for such large contexts. Instead, we are defining a different relation, which determines a connection between two sentences if there is a "similarity" relation between them, where "similarity" is measured as a function of their content overlap. Such a relation between two sentences can be seen as a process of "recommendation": a sentence that addresses certain concepts

in a text, gives the reader a “recommendation” to refer to other sentences in the text that address the same concepts, and therefore a link can be drawn between any two such sentences that share common content. The overlap of two sentences can be determined simply as the number of common tokens between the lexical representations of the two sentences, or it can be run through syntactic filters, which only count words of a certain syntactic category, e.g. all open class words, nouns and verbs, etc. Moreover, to avoid promoting long sentences, we are using a normalization factor, and divide the content overlap of two sentences with the length of each sentence.

5.2 WHY TEXTRANK WORKS?

Intuitively, TextRank works well because it does not only rely on the local context of a text unit (vertex), but rather it takes into account information recursively drawn from the entire text (graph).Through the graphs it builds on texts, TextRank identifies connections between various entities in a text, and implements the concept of *recommendation*. A text unit recommends other related text units, and the strength of the recommendation is recursively computed based on the importance of the units making the recommendation. For instance, in the keyphrase extraction application, co-occurring words recommend each other as important, and it is the common context that enables the identification of connections between words in text. In the process of identifying important sentences in a text, a sentence recommends another sentence that addresses similar concepts as being useful for the overall understanding of the text. The sentences that are highly recommended by other sentences in the text are likely to be more informative for the given text, and will be therefore given a higher score. An important aspect of TextRank is that it does not require deep linguistic knowledge, nor domain or language specific annotated corpora, which makes it highly portable to other domains, genres, or languages.

CHAPTER 6

DEVELOPMENT

To make User Interface of My Tool, I have used swings in advanced java and then perform some methods and algorithm as I mentioned above to produce a summarization of the text document. Below are the snapshots of my project after applying methods:

6.1 User Interface:

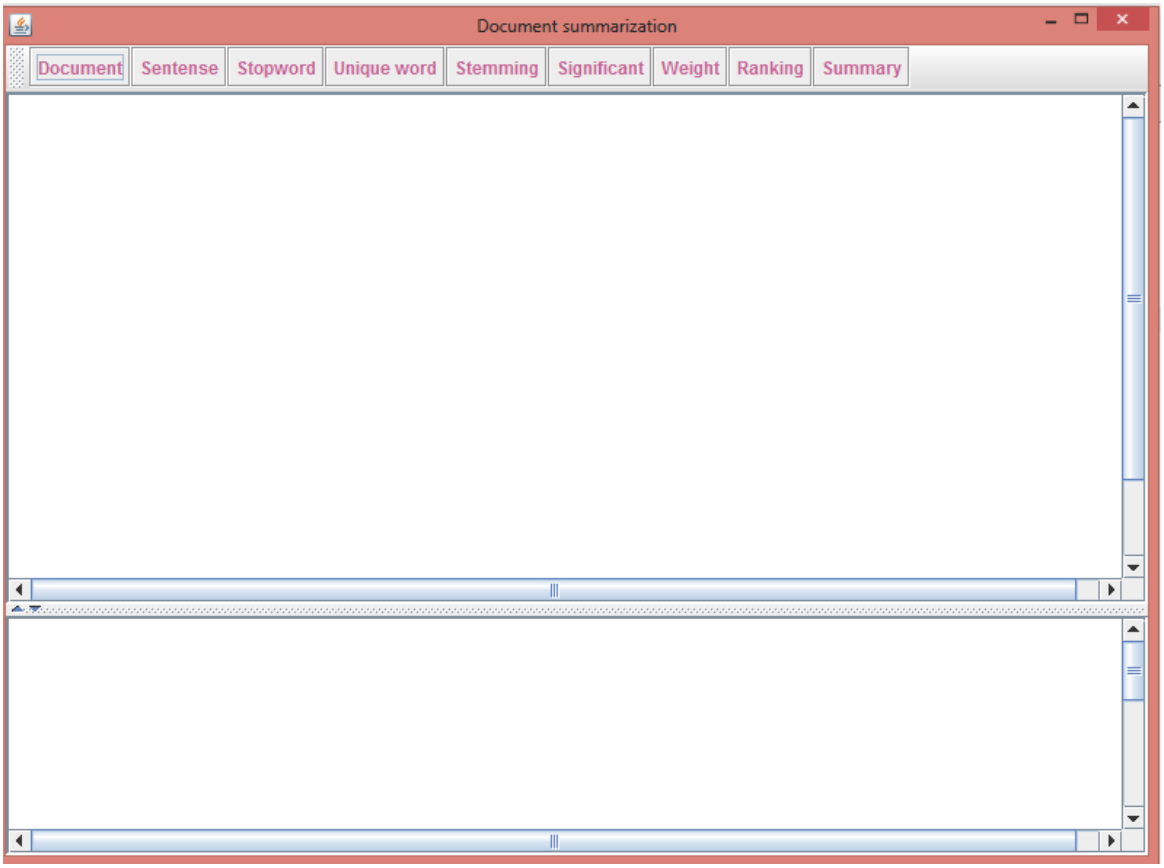


Figure.9 User Interface

6.2 Sentence Extraction:

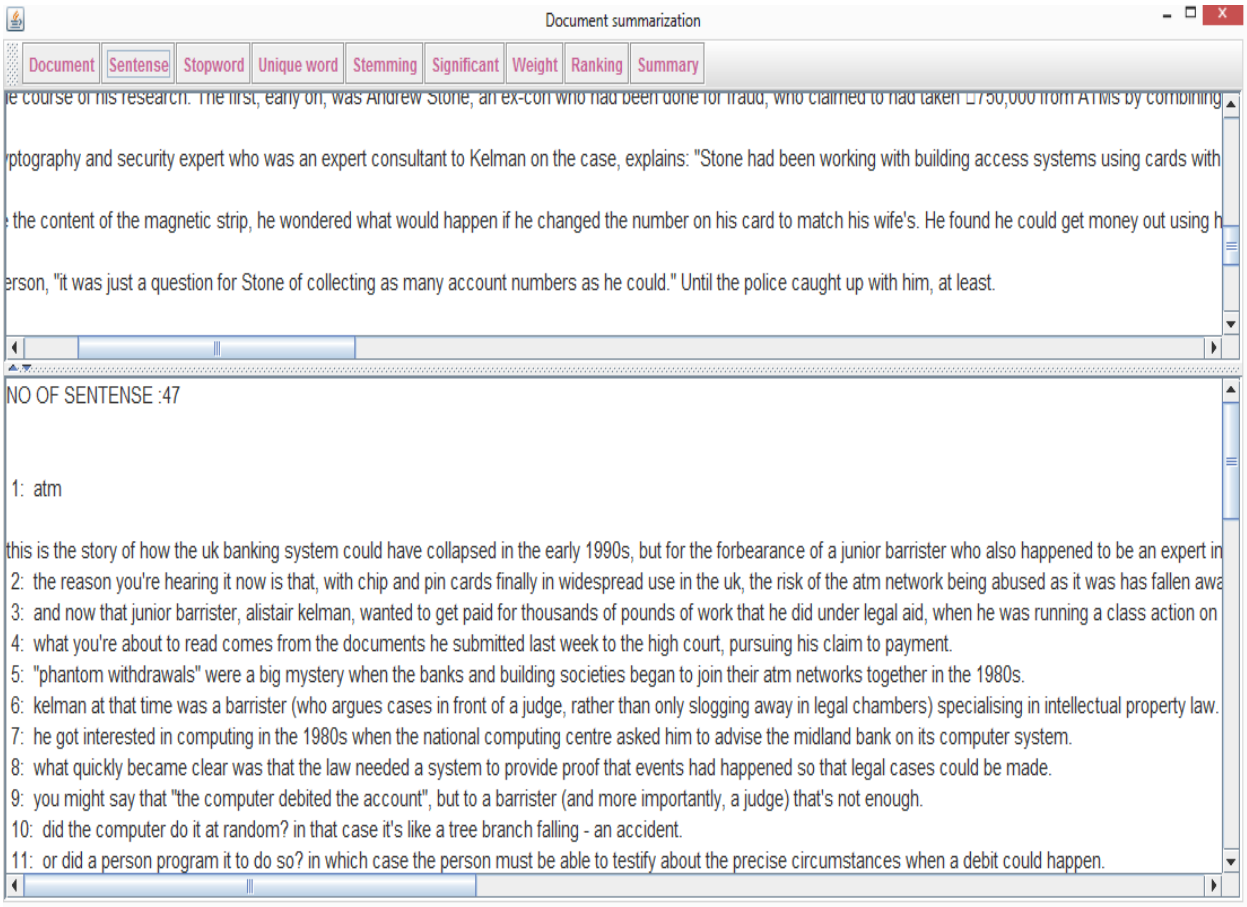


Figure. 10 Sentence Extraction

6.3 Unique Words Indentification:

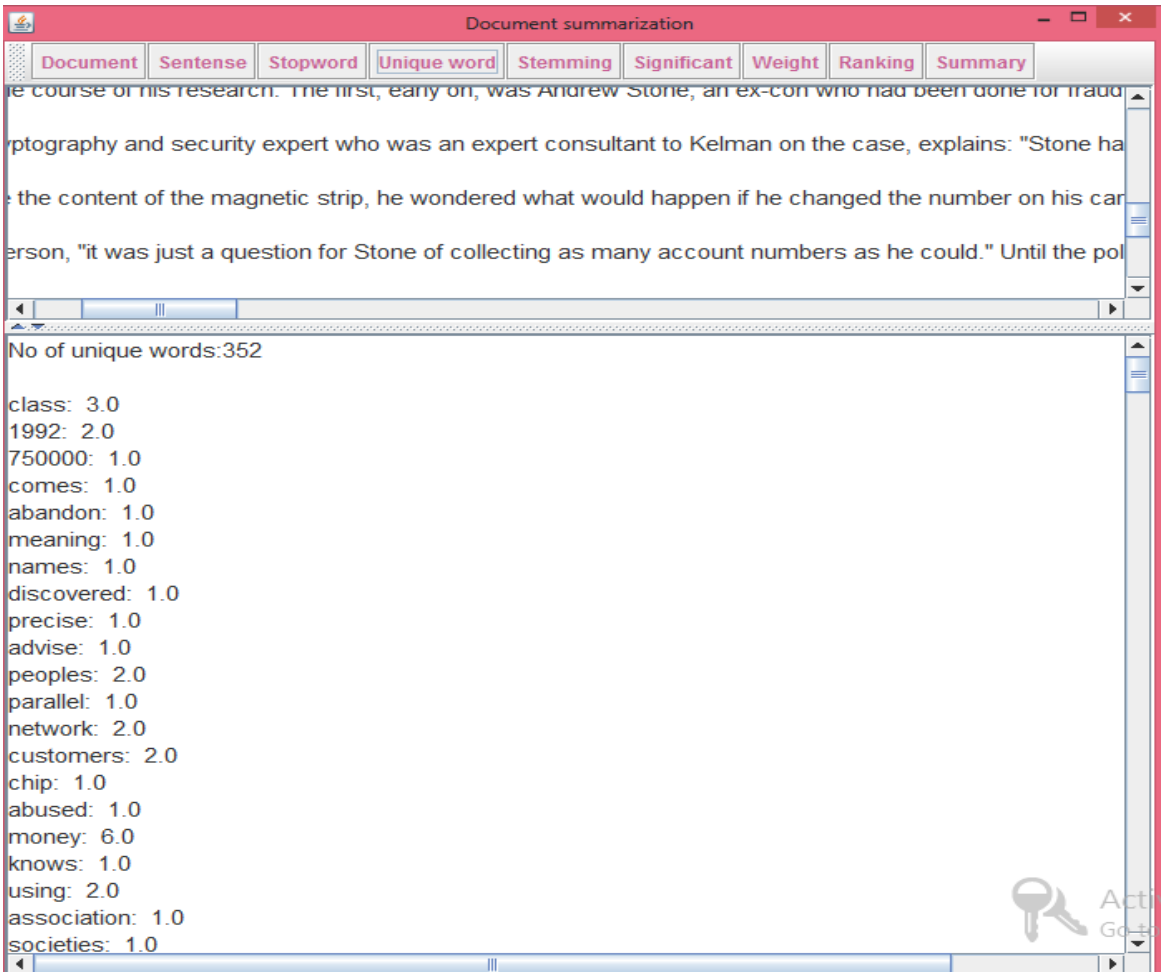


Figure.11 Unique word Identification

6.4 Weight of Particular Word By tf-idf Method:

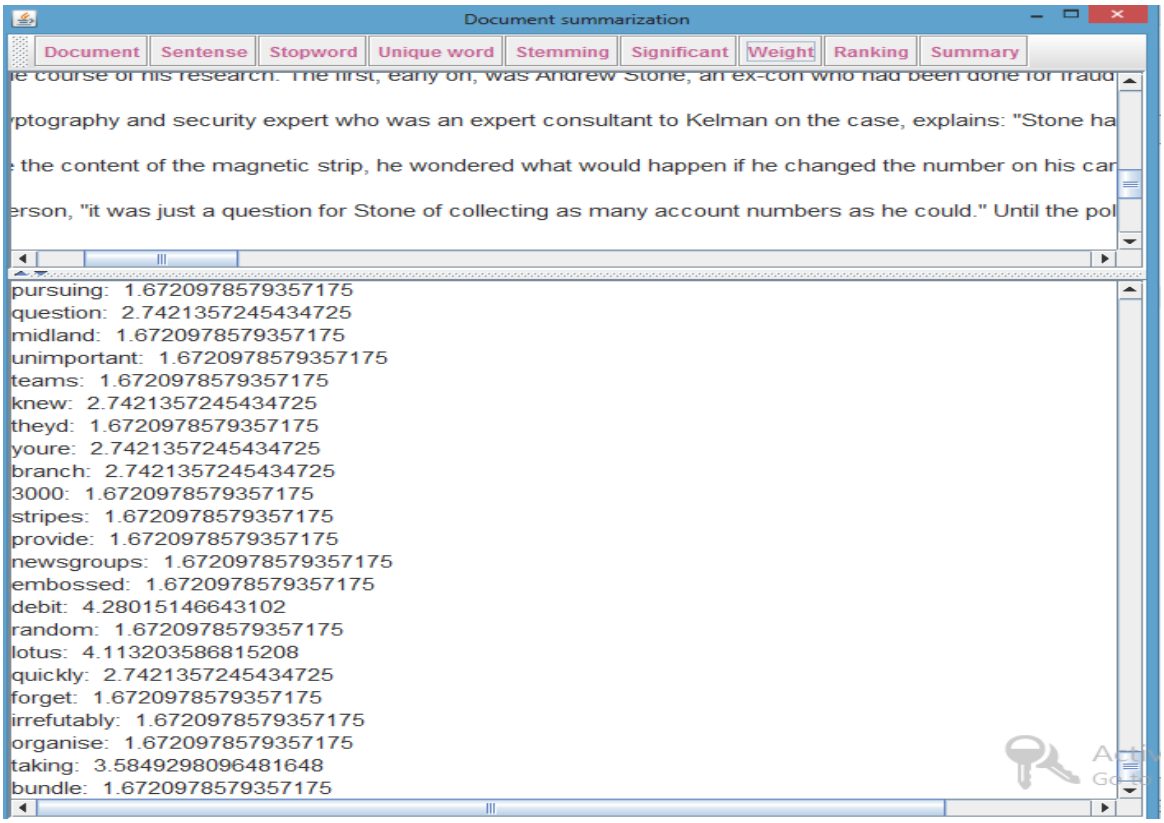


Figure 12. Weight of a particular word

6.5 Ranking of sentences:

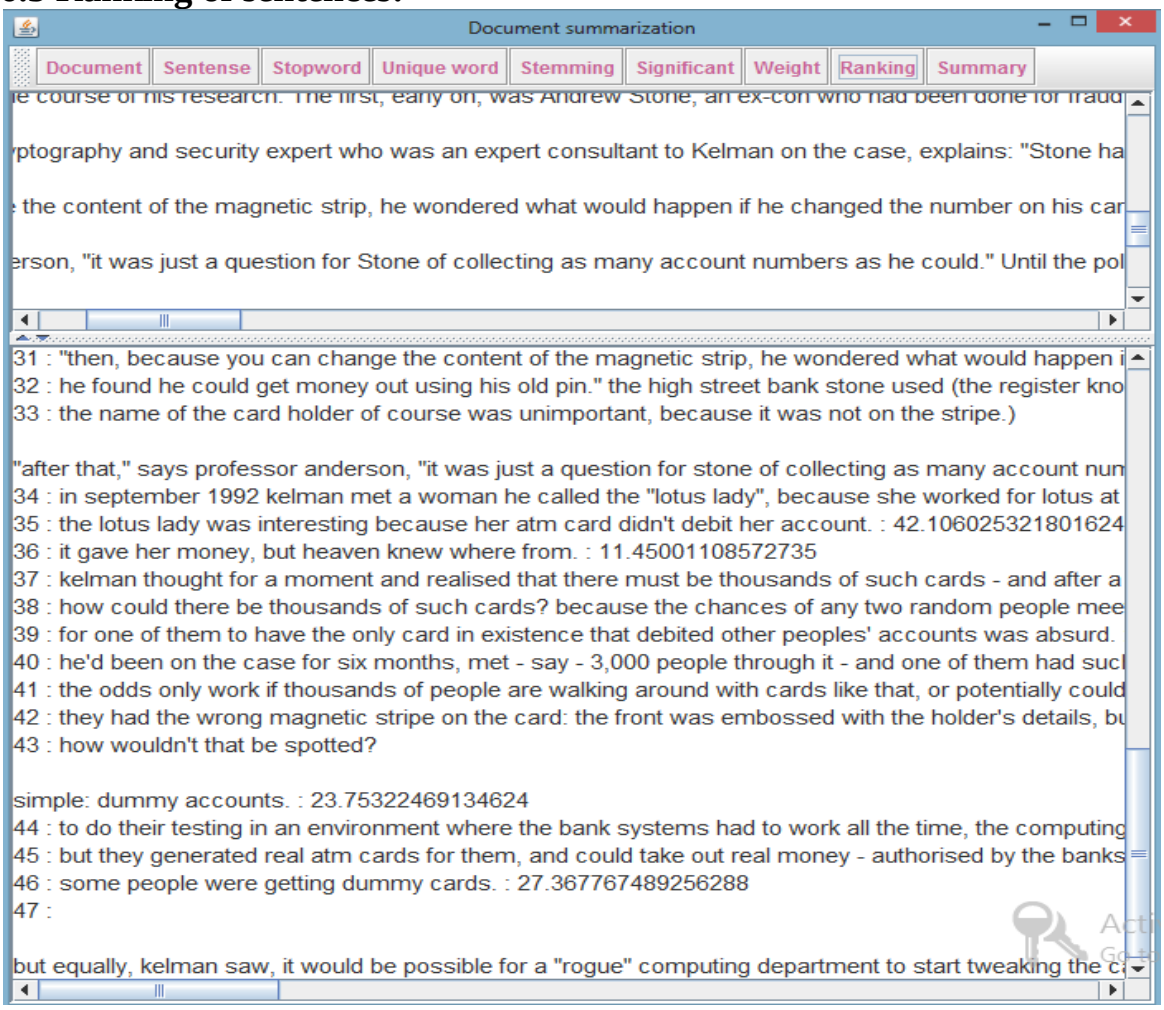


Figure 13. Ranking

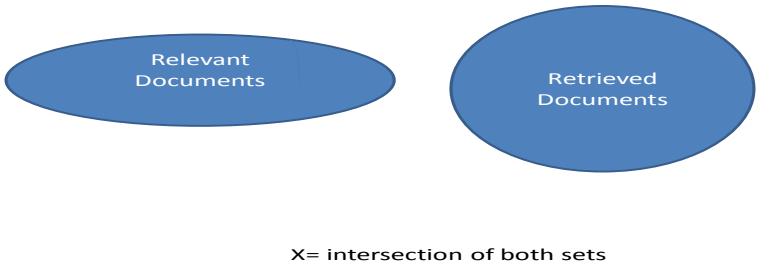
CHAPTER 7

EXPERIMENTS

We have done the following experiments to summarize the documents

Baseline

1. Selecting all words
2. Cosine Similarity
3. TextRank



Precision evaluates the ability of the IR system to retrieve top-ranked documents that are most relevant and is defined as the percentage of the retrieved documents that are truly relevant to user's query.

$$Precision = \frac{X}{retrieved\ documents}$$

Precision evaluates the ability of the IR system to find all the relevant items in the database and is defined as the percentage of the retrieved documents that are truly relevant to user's query.

$$Recall = \frac{X}{relevant\ documents}$$

The results for this baseline were the following

	Machine Generated Summaries			Human Summaries		
N	**Precision**	**Recall**	**FMeasure**	**Precision**	**Recall**	**FMeasure**
1	70.58	44.87	50.67	24.98	50.47	30.87
2	59.21	34.73	40.49	9.79	18.21	11.65
3	56.32	32.26	37.96	6.14	10.67	7.10

Table 1. Baseline result

EXPERIMENT 7.1

For this experiment, we compared the results of syntactic filtering and compared which one gives the best results. We have used the average of various similarity measures to compute the value for each syntactic filter.

	Nouns + Adjective			Nouns + Adjective + Verbs			Nouns + Verbs		
N	1	2	3	1	2	3	1	2	3
Precision	74.40%	57.50%	51.85%	74.10%	56.60%	51.00%	72.00%	56.00%	52.70%
Recall	54.40%	36.80%	31.90%	54.30%	35.80%	30.90%	48.50%	37.40%	34.10%
FMeasure	57.20%	41.20%	36.50%	57.00%	40.30%	35.70%	52.70%	41.30%	38.25%

Table 2. Syntactic filter

Inferences

- The Nouns and adjectives outperform the other two features for N = 1.
- However, for N = 3, Nouns and verbs perform much better.

Table 3. Cosine Similarity

	Machine Generated Summaries			Human Summaries		
N	**Precision**	**Recall**	**FMeasure**	**Precision**	**Recall**	**FMeasure**

1	71.53	56.35	57.29	23.31	58.16	30.63
2	62.98	47.09	48.85	10.50	24.98	13.49
3	60.45	44.51	46.43	6.69	15.11	8.41

Inference

- The FMeasure is significantly lower than the FMeasure obtained from the comparison with Manual Summaries.
- However a decrease in FMeasure does not mean that our summaries are not good. The evaluation method used by us compares the ngrams in each document. However, if the user generates summary using other words and changing the order of sentences, then any computer-generated summary will score low on FMeasure. Hence, we should not compare the FMeasure values obtained with human summaries and computer summaries.

EXPERIMENT 7.2

Using maximum occurring words without stop words as the keywords

	Intellexer			DUC		
N	**Precision**	**Recall**	**FMeasure**	**Precision**	**Recall**	**FMeasure**
1	69.39	68.70	67.42	27.06	53.79	35.57
2	59.04	58.28	57.24	11.76	23.07	15.36
3	56.35	55.55	54.58	7.55	14.69	9.82

Table 4. Maximum occurring words

EXPERIMENT 7.3

Using 2 weight classes for sentences with Top occurring words as Keywords with stop words

	Intellexer			DUC		
N	**Precision**	**Recall**	**FMeasure**	**Precision**	**Recall**	**FMeasure**

1	67.96	71.63	68.19	25.24	54.68	34.16
2	57.92	60.58	57.84	10.70	22.65	14.34
3	55.30	57.75	55.17	6.76	14.09	9.00

Table 5. Using 2 weight classes

EXPERIMENT7.4

Using 3 weight classes for sentences with Top occurring words as Keywords with stop words

	Intellexer			DUC		
N	Precision	Recall	FMeasure	Precision	Recall	FMeasure
1	68.00	71.65	68.23	25.23	54.63	34.13
2	57.98	60.63	57.90	10.68	22.61	14.32
3	55.37	57.79	55.23	6.74	14.04	8.98

Table 6. Using 3 weight classes

Inferences

- We see that Bipartite HITS seems to give better results than the other conventional methods
- Moreover, both TextRank and maximum occurring words as keywords perform well, with the latter performing slightly better.

Table 7. CONSOLIDATED RESULT FOR TEXT RANK

Experiment	**Algorithm**	**Intellexer**			**DUC**		
		FMEASURE			**FMEASURE**		
		N = 1	N = 2	N = 3	N = 1	N = 2	N = 3
Baseline	All Words + Cosine Similarity + TextRank	50.67	40.49	37.96	30.87	11.65	7.10
Experiment 1	TextRank with Jaccard Similarity	54.93	45.65	43.12	31.26	13.72	8.59
Experiment 2	TextRank with Cosine Similarity	57.29	48.85	46.43	30.63	13.49	8.41

CHAPTER 8

Challenging Issues of Automatic Summarization

Relevance Detection and Quality-based Evaluation

This chapter is about the Automatic Summarization task within two different points of view, focusing on two main goals. On the one hand, a study of the suitability for "The Code Quantity Principle" in the Text Summarization task is described. This linguistic principle is implemented to select those sentences from a text, which carry the most important information. Moreover, this method has been run over the DUC 2002 data, obtaining encouraging results in the automatic evaluation with the ROUGE tool. On the other hand, the second topic discussed in this chapter deals with the evaluation of summaries, suggesting new challenges for this task. The main methods to perform the evaluation of summaries automatically have been described, as well as the current problems existing with regard to this difficult task. With the aim of solving some of these problems, a novel type of evaluation is outlined to be developed in the future, taking into account a number of quality criteria in order to evaluate the summary in a qualitative way.

8.1 Introduction

The high amount of electronic information available on the Internet increases the difficulty of dealing with it in recent years. Automatic Summarization (AS) task helps users condense all this information and present it in a brief way, in order to make it easier to process the vast amount of documents related to the same topic that exist these days. Moreover, AS can be very useful for neighbouring Natural Language Processing (NLP) tasks, such as Information Retrieval, Question Answering or Text Comprehension, because these tasks can take advantadge of the summaries to save time and resources.

A summary can be defined as a reductive transformation of source text through content condensation by selection and/or generalisation of what is important in the source . This

process involves three stages: topic identification, interpretation and summary generation. To identify the topic in a document what systems usually do is to assign a score to each unit of input (word, sentence, passage) by means of statistical or machine learning methods. The stage of interpretation is what distinguishes extract- type summarization systems from abstract-type systems. During interpretation, the topics identified as important are fused, represented in new terms, and expressed using a new formulation, using concepts or words not found in the original text. Finally, when the summary content has been created through abstracting and/or information extraction, it requires techniques of Natural Language Generation to build the summary sentences. When an extractive approach is taken, there is no generation stage involved.

Another essential part of the Text Summarization (TS) task is how to perform the evaluation of a summary. Methods for evaluating TS can be classified into two categories. The first, intrinsic evaluations, test the summary on itself. The second, extrinsic evaluations, test how the summary is good enough to accomplish some other task, for example, an Information Retrieval task. However, to determine whether an automatic, or even a human-made summary, is appropriate or not, is a subjective task which depends greatly on a lot of factors, for instance, what the summary is intended for, or to whom the summary is addressed. We focus on single-document Text Summarization from an extractive point of view, and we set out two goals for this research. On the one hand, the first goal is to present a method to detect relevant sentences within a document, and therefore, select them to make up the final summary. On the other hand, the second aim of this piece of work is to discuss the current problems the automatic evaluation of summaries in a quantitative way have, so that we can outline a novel approach to measure the quality of a summary to be developed in further research.

8.2 Determining sentence's relevance in text summarization

Although there has been increased attention to different criteria such as well-formedness, cohesion or coherence when dealing with summarization most work in this NLP task is still concerned with detecting relevant elements of text and presenting them together to

produce a final summary. As it has been previously mentioned, the first step in the process of summarization consists of identifying the topic of a document. To achieve this, the most common things systems do is to split the text into input units, usually sentences, and give them a relevance score to decide on which ones are the most important. Criteria such as *sentence position* within texts and *cue phrase indicators , word and phrase frequency*, *query and title overlap ,cohesive or lexical connectedness* or *discourse structure* are examples of how to account for the relevance of a sentence. Furthermore, the use of a graph to obtain a representation of the text has proven effective, especially in multi-document summarization .

In contrast to all this work, this paper suggests a novel approach for determining the relevance of a sentence based on *"The Code Quantity Principle"* . This principle tries to explain the relationship between syntax and information within a text. The first goal of this chapter is to study whether this principle can be suitable or not as a criterion to select relevant sentences to produce a summary. This idea will be explained in detail in the next Section.

8.3 The code quantity principle within the text summarization task

"The Code Quantity Principle" is a linguistic theory which states that: (1) a larger chunk of information will be given a larger chunk of code; (2) less predictable information will be given more coding material; and (3) more important information will be given more coding material. In other words, the most important information within a text will contain more lexical elements, and therefore it will be expressed by a high number of units (for instance, syllables, words or phrases). Moreover, *"The Code Quantity, Attention and Memory Principle"* states that the more salient and different coding information used within a text, the more reader's attention will be caught. As a result, readers will retain, keep and retrieve this kind of information more efficiently. There exists, then, a proportional relation between the relevance of information and the amount of quantity through it is coded. On the basis of this, a coding element can range from characters to phrases. A noun-phrase is the syntactic structure which allows more flexibility in the

number of elements it can contain (pronouns, adjectives, or even relative clauses), and is able to carry more or less information (words) according to the user's needs. Furthermore, the longer a noun-phrase is, the more information it carries for its nucleus. For example, if a text contained two distinct noun-phrases referring to the same entity ("the Academy of Motion Pictures Arts and Sciences" and "the Academy"), the second one would lead to ambiguities. Therefore, if a summary selected this noun-phrase without having previously given more specific information about the concept, the real meaning could be misunderstood. Starting from these principles, the approach we suggest here is to study how "The Code Quantity Principle" can be applied in the summarization task, to decide on which sentences of a document may contain more relevant in- formation through its coding, and select these sentences to make up a summary. In this particular case, the lexical units considered as encoding elements are words in- side a noun-phrase, without taking into account stopwords. The hypothesis is that sentences containing longer noun-phrases will be given a higher score so, at the end, the highest ranked sentences will be chosen to appear in the final summary. To identify noun-phrases within a sentence the BaseNP Chunker2, which was developed at the University of Pennsylvania, was used. One important thing to take into consideration is that the use of a chunker (as well as any other NLP tool) can introduce some error rate. This tool achieves recall and precision rates of roughly 93% for base noun-phrase chunks, and 88% for more complex for base noun-phrase chunks, and 88% for more complex chunks . For the experiments performed, the score for a sentence was increased by one unit, each time a word belonged to a sentence's noun-phrase.

8.4 Evaluating automatic summarization

Evaluating summaries, either manually or automatically, is a hard task. The main difficulty in evaluation comes from the impossibility of building a fair gold-standard against which the results of the systems can be compared . Furthermore, it is also very hard to determine what a correct summary is, because there is always the possibility of a system to generate a good summary that is quite different from any human summary used as an approximation to the correct output. In Section 1, we mentioned the two approaches that can be adopted to evaluate an automatic summary: instrinsic or extrinsic evaluation.

Instrinsic evaluation assesses mainly coherence and summary's information content, whereas extrinsic methods focus on determining the effect of summarization on some other task, for instance Question Answering.

8.5 The code quantity principle evaluation environment

For the approach we have suggested taking into consideration *"The Code Quantity Principle",* we have chosen an intrinsic evaluation because we are interested in measuring the performance of the automatic summary by itself. To do this, we used the state-of-the-art measure to evaluate summarization systems automatically, ROUGE . This metric measures content overlap between two summaries (normally between a gold-standard and an automatic summary), which means that the distance between two summaries can be established as a function of their vocabulary (unigrams) and how this vocabulary is used (n-grams).

In order to assess the performance of our novel approach based on *"The Code Quantity Principle"* and show that it is suitable for Text Summarization, we evaluated the summaries generated from the DUC 2002 data, consisting of 567 newswire documents. As a preprocessing step, we converted the HTML files into plain text, and we kept only the body of the news. In the DUC 2002 workshop[2], there was a task whose aim was to generate 100-word length sum maries. A set of human-made summaries written by experts was also provided. We evaluated our summaries against the reference ones, and we compared our results with the ones obtained by the systems in the real competition. Moreover, the organisation developed a simple baseline which consisted of taking the first 100 words of a document. In , the participating systems in DUC 2002 were evaluated automatically with the ROUGE tool, and we set up the same settings[5] for it, so that we could make a proper comparison among all the systems.

8.6 Current difficulties in evaluating summaries automatically

The most common way to evaluate the informativeness of automatic summaries is to compare them with human- made model summaries. However, as content selection is not a deterministic problem, different people would chose different sentences, and even, the same person may chose different sentences at different times, showing evidence of low agreement among humans as to which sentences are good summary sentences. Besides the human variability, the semantic equivalence is another problem, because two distinct sentences can express the same meaning but not using the same words. This phenomenon is known as paraphrase. we can find an approach to automatically evaluating summaries using paraphrases (ParaEval). Moreover, most summarization systems perform an extractive approach, selecting and copying important sentences from the source documents. Although humans can also cut and paste relevant information of a text, most of the times they rephrase sentences when necessary, or they join dif- ferent related information into one sentence .

For years, the summarization community research has been actively seeking an automatic evaluation methodology. Several methods have been proposed, and thanks to the conferences carried out annually until 2007 within the DUC context6, some of these methodologies, for instance, ROUGE or the Pyramid Method have been well adopted by the researchers to evaluate summaries automatically. Although ROUGE is a recall-oriented metric, the lat- est version (ROUGE-1.5.5) can compute precision and F-measure, too. It is based on content overlap and the idea be- hind it is to assess the number of common n-grams between two texts, with respect to different kinds of n-grams, like unigrams, bigrams or the longest common subsequence. In order to address some of the shortcomings of the comparison of fixed words n-grams, an evaluation framework in which very small units of content were used, called Basic Elements (BE) was developed .

The idea underlying the Pryamid method is to identify information with the same meaning across different human-authored summaries, which are tagged as Summary Content Units (SCU) in order to derive a gold-standard for the evaluation. Each SCU will have a weight depending on the number of summarizers who expressed the same information, and these weights will follow a specific distribution, allowing important content to be differentiated from less important one. The main disadvantages of this method are (1) the need to have several human-made summaries, and the labourious task to annotate all the SCU. An at- tempt to automate the annotation of the SCUs in the pyramids can be found in. In the former, Relative Utility (RU) is proposed as a metric to evaluate summaries, where multiple judges rank each sentence in the input with a score, giving them a value which ranged from 0 to 10, with respect to its suitability for inclusion in a summary. Highly ranked sentences would be very suitable for a summary, whereas low ranked ones should not be incuded. Like the commonly used information retrieval metric of precision and recall, it compares sentence selection between automatic and reference summaries. The latter have developed an evaluation framework, called QARLA, which provides three types of measures for the evaluation under the assumption that the best similarity metric should be the one that best discriminates between manual and automatically generated summaries. These measures are: (1) a measure to evaluate the quality of any set of similarity metrics, (2) a measure to evaluate the quality of a summary using an optimal set of similarity metrics, and (3) a measure to evaluate whether the set of baseline summaries is reliable or may produce biased results.

Despite the fact that many approaches have been developed, some important aspects of summaries, such as legibility, grammaticality, responsiveness or well-formedness are still evaluated manually by experts. For instance, DUC assessors had a list of linguistic qualitity questions7, and they manually assigned a mark to automatic summaries depending on what extent they accomplished each of these criteria.

8.7 Evaluating summaries qualitatively

The main drawback of the evaluation systems existing so far is that we need at least one reference summary, and for some methods more than one, to be able to compare automatic summaries with models. This is a hard and expensive task. Much effort has to be done in order to have corpus of texts and their corresponding summaries. Furthermore, for some methods presented in the previous Secion, not only do we need to have human-made summaries available for comparison, but also manual annotation has to be performed in some of them (e.g. SCU in the Pyramid Method). In any case, what the evaluation methods need as an input, is a set of summaries to serve as gold-standards and a set of automatic summaries. Moreover, they all perform a quantitative evaluation with regard to different similarity metrics. To overcome these problems, we think that the quantitative evaluation might not be the only way to evaluate summaries, and a qualitative automatic evaluation would be also important. Therefore, the second aim of this paper is to suggest a novel proposal for evaluating automatically the quality of a summary in a qualitative manner rather than in a quantitative one. Our evaluation approach is a preliminary approach which has to be studied more deeply, and developed in the future. Its main underlying idea is to define several quality criteria and check how a generated summary tackles each of these, in such a way that a reference model would not be necessary anymore, taking only into consideration the automatic summary and the original source. Once performed, it could be used together with any other automatic methodology to measure summary's informativeness.

APPLICATIONS

Other than presenting information in summarized form there can be various applications in which this system can be used. The applications of a multi document summarizer are:

a) It can be used as a news portal and can help to present articles from different sources.
b) Corporate emails or emails in general can be organized by subjects with relevant and most important information.
c) It can help to obtain precise information which is represented as charts or graphs along with related text.
d) It can be used to generate medical reports for patients.
e) Aggregating social media data.

CONCLUSION AND FUTURE WORK

This report is concentrating on extractive summarization methods. An extractive summary is selection of important sentences from the original text. The importance of sentences is decided based on statistical and linguistic features of sentences.

Many variations of the extractive approach have been tried in the last ten years. However, it is hard to say how much greater interpretive sophistication, at sentence or text level, contributes to performance. Without the use of NLP, the generated summary may suffer from lack of cohesion and semantics. If texts containing multiple topics, the generated summary might not be balanced. Deciding proper weights of individual features is very important as quality of final summary is depending on it. We should devote more time in deciding feature weights.

The biggest challenge for text summarization is to summarize content from a number of textual and semi structured sources, including databases and web pages, in the right way (language, format, size, time) for a specific user. The text summarization software should produce the effective summary in less time and with least redundancy. Summaries can be evaluated using intrinsic or extrinsic measures. While intrinsic methods attempt to measure summary quality using human evaluation and extrinsic methods measure the same through a task based performance measure such the information retrieval- oriented task.

REFERENCES

1. A. Turpin, Y. Tsegay, D. Hawking, and H. E. Williams, “Fast generation of Advances in Automatic Text Summarization, pp. 1–12, MIT Press, 1998.

2. Ani Nenkova and Kathleen McKeown, “Automatic Summarization”,in Information ,Vol. 5, Nos. 2,103–233,2011

3. Fang Chen, Kesong Han and Guilin Chen, “An Approach to sentence selection based text summarization" Proceedings of IEEE TENCON02, 489-493, 2002.

4. H. P. Luhn, “The Automatic Creation of Literature Abstracts”, Presented at IRE National Convention, New York, 159-165, 1958.

5. http://en.wikipedia.org/wiki/Automatic_summarization

6. J. Kupiec, J. Pedersen, and F. Chen, “A trainable document summarizer”, In Proceedings of the 18th ACM- SIGIR Conference, pages 68-73, 1995

7. Joel Larocca Neto, Alex A. Freitas and Celso A. A. Kaestner,” Automatic Text Summarization using a Machine Learning Approach”

8. K. Sparck Jones, “Automatic summarizing: factors and directions,” in Knowledge Management, 2006.

9. Mani, I.; Bloedorn, E. Machine Learning of Generic and User-Focused Summarization. In Proceedings of the Fifteenth National Conference on AI (AAAI-98) (1998) 821-826

10. Nevill-Manning, C. G. ; Witten, I. H. Paynter, G. W. et al. KEA: Practical Automatic Keyphrase Extraction. ACM DL 1999 (1999) 254-255

11. R. Varadarajan and V. Hristidis, “A system for query-specific document result snippets in web search,” in Proceedings of the Annual International

12. Ronald Brandow, Karl Mitze, and Lisa F. Rau. ,“Automatic condensation of electronic publications by sentence selection. Information Processing and Management”, 31(5):675-685,1995.summarization,” in Proceedings of the ACM Conference on Information

13. Rada Mihalcea and Paul Tarau ,“ TextRank : Bringing Order into Texts ”,Presented at Conference on Empirical Methods in Natural Language Processing,2011

14. Vishal Gupta and Gurpreet Singh Lehal ,“A Survey of Text Summarization Extractive Techniques”,Presented at Journal Of Emerging Technologies In Web Intelligence,Vol. 2,No. 3,August 2010