Table of Contents

CHAPTER 1: INTRODUCTION

1.1 MOTIVATION

Today's world is all about the information mostly available online. The World Wide Web contains the huge amount of documents and still growing at an exponential pace. So we need some techniques or tools that provides; timely access to, and digest of, various resources are necessary in order to get rid of information overload the people are facing.

All these concerns have sparked the interest in the development of the Automatic Text Summarization (ATS) systems. And all these systems are designed to take article(s) as an input (any random or arbitrary text) and generate respective summarised text.

By winding information from document heaps, summarization can be thus called a boosting technology so as to organize the available heaps of information which users face regularly. For retrieving information, summarization is much helpful and interesting enough in fields where users process a larger number information related to various topics, for example database of search engine, news articles, social media content like comments, blogs, reviews.

This can be easily done by Multi Document Summarization (MDS) systems, which is the strongest fruit nowadays in the field of research in Automatic Text Summarization (ATS). The thematic documents and query oriented information used by users is provided as input to such systems. By providing brief and crisp summaries to MDS systems as input it then identifies the similar and dissimilar information, which then picks up the relevant content, by eliminating or clearing the redundancies, and the dissimilar information from the document [25].

1.2 SCOPE

The fashion that exist these days in this world of information is available online. The World Wide Web having trillions of obtainable online documents is growing speedily. The overburden that everyone faced nowadays procreates the avail of some necessary tools and sources, thus to deflate this massive downside to bit limit. These tools will therefore offer possible door to knowledge. These behaviour points to the development of automatic summarization systems.

These system design in such a way that it yields input as articles that ought to be strictly single, for instance it may be like newspaper articles, group discussions, or it might be any emails so the output is generated that is natural and crisp in nature as a summary. Huge amount of supply information is contained in these summaries that aim at reflecting main content of the source. Day to day life summaries may be a smart example to take for instance newspaper articles, cinema scenes and post book reviews. Summaries may be more practical moreover as useful enough to derive conclusion for compilation and organizing terribly great amount of knowledge. Information can even be collected ad combined in totally different source document, by stating their similarities moreover as variations, by providing topic based briefing for big quantity of information. Complementary advantage is that, consistent with the user information demand these summaries may be customized consequently, filtering the noise information from the source. All these blessings square measure useful to create up interest for making or generating summaries within the field of automatic summarization.

1.3 RESEARCH GOAL

There possibly following Goals

- To study the state of art in the area of automatic text summarization.
- To study the various approaches / techniques for generating summary.
- Automatic text summarization tools can help people to grasp main concepts of information sources in a short time.

1.4 ORGANISATION OF REPORT

In **Chapter 2**, I have given the Literature Survey which includes the review of extraction based approaches and what are the existing algorithms in these approaches.

In **Chapter 3,** I have discussed Research Methodology, proposed work and algorithms used in the implementation.

In **Chapter 4,** I have discussed Experimentation i.e. all about infrastructure required to be setup to do the further experimentation with minimum required resources.

In **Chapter 5,** the Results of the various implemented algorithms as well as the analysis of these Results

In **Chapter 6,** the conclusions drawn from the results as well as the scope for future work is discussed.

REFERENCESES

Appendix A: Code Snippets

Appendix A: Snapshots of the system

CHAPTER 2: LITERATURE SURVEY

2.1 History of Text Summarization

Enthusiasm for automatic text summarization, emerged as right on time as the fifties. An important public of nowadays is the one in 1958, proposed to weight the sentences of a document as a function of high recurrence words [1], ignoring the very high recurrence basic words. Automatic text summarization system [2] in 1969, which notwithstanding the standard catchphrase strategy (i.e., frequency depending weights), Also used these three methods for determining the sentence weights:

1. Cue Method: This depends on the theory that the importance of a sentence is figured by the nearness or nonattendance of certain cue words in the sign lexicon.
2. Title Method: Here, the sentence weight is registered as an aggregate of all the content words showing up in the title and (sub-) headings of a text.
3. Location Method: This method depends on the presumption that sentences occurring in introductory position of both text and individual paragraphs have a higher probability of being relevant. The outcomes appeared, that the best connection between the automatic and human-made extracts was accomplished utilizing a mix of these three latter methods.

The Trainable Document Summarizer [17] in 1995 performs sentence separating assignment, taking into account various weighting heuristics. Taking after features were utilized and assessed:

1. Sentence Length Cut-O Feature: sentences containing not exactly a pre-indicated number of words are excluded in the abstract.

2. Fixed-Phrase Feature: sentences containing certain sign words and phrases are incorporated.

3. Paragraph Feature: this is essentially proportional to Location Method feature in [2]

4. Thematic Word Feature: the most occurring words are characterized as thematic words. Sentence scores are elements of the thematic words' frequencies.

5. Capital Word Feature: capitalised words (with certain conspicuous exemptions) are dealt with as thematic words, too.

A Corpus was utilized as a part of method, which contained 188 document/summary sets from 21 publications in an exploratory/specialized domain. The summaries were created by expert specialists and the sentences happening in the summaries were adjusted to the original document texts, indicating also the level of closeness as specified before, by far most (about 80%) of the summary sentences could be named direct sentence matches

The ANES text extraction system [10] in 1995 is a framework that performs automatic, space free build-up of news information. The procedure of summary generation has four noteworthy constituents:

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

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

3. Sentence weighting: summing over all signature word weights, changing the weights by some different variables, for example, relative location

4. Sentence determination: Selecting high scored sentences.

2.2 Early Approaches for Extraction

Earlier approaches include exclusive dealing with identification of important content at the sentence level. The tradition for sentence extraction was set due to the work done by Luhn somewhere near 1950's, which was the first work in the field of automatic summarization. All this was implemented to work on magazine articles and technical papers. A straight forward idea was put forward by him [1], namely, some words illustrates some content also document's most important information contain many of these illustrative words.

He put forward that frequency of occurrence of the words can be used so that the words which are most illustrative or descriptive of the content. Luhn gave two warning conditions: some of most common words in an article of magazine or technical paper and in fact are not much informational about the content. All the common words for example it could be pronouns, prepositions are not completely capable of informing about the document. He called stop word table, by using predefined table, for removing the words. Some words that are not highlighted in the table are not indicative. For instance, the word "cell" in a scientific paper in a cell biology may not give much idea about the paper.

Also, the words appearing less in a document are not at all informative. Provisionally checked frequency either high or low to identify descriptive words [1]. High thresholds used to filter out words occurring at short intervals also elevated thresholds used to filter words occurring very less.

Descriptive words are the ones which are remaining, these indicates the important content. Sentences which are portrayed by eminent density of illustrative texts, are described as blob of Luhn's five one after the another words are important. Luhn's approach involve some problems. Luhn, give the rough idea as he knew the problem. He stated that merge similar words up to six letters so as to reduce the problem to some extent.

Several other trends in summarization research was the foundation of Edmundson which gave some approaches like machine learning. Luhn's expanded his formulas, he

proposed that sentence importance may be indicated by multiple features. To weigh sentences in a scientific article a linear combination of features was used by him [1][2].

These features were:

1. Number of words appearing in the article.
2. Sentence position in the article.
3. Title words or section headings.
4. Words that match a pre-compiled cue words list for example “In sum”.

An irresistible demeanour of his work tells that it was totally based on ongoing approaches and these are further categorised or compiled up under extractive summaries. Further this compilation was used by him [1] to calculate weights as well as evaluate them. On evaluating these above stated characteristics it was observed that word frequency is not used much. Whereas other features are of much more importance as well of domain interest. So these domain dependent approaches are discussed for summarization of medical information, email and scientific articles.

2.3 The Broad Classification in Text Summarization

The two pioneers Hahn and Mani showed the various techniques. The classification is as follows:

A. Query-relevant summarization

The user gives the input in the form of a query and thus the similar material related to the query is released. The material released is given a form known as a summary. Because that material basically contains that query.

The two brilliant pioneers Gong and Liu based on this query matching rank the sentences by giving some weights using latent semantic analysis (LSA). Similarly, whereas Park et al. used another different technique. Their technique was highly beneficial and performance wise it was outstanding.

K-means clustering was used to analyse the query sentences and it coupled with non-negative factorization basically used for matrix. Another pioneer Tang et al. used scoring methods with unified methods. The probability is determined by using the scores. The highest scores are counted and based on that important is identified. Then by using these summary is generated.

B. Generic Summarization

In this method nothing is fed in terms of query nor as topic. It is a highly robust method which focuses on having the minimum redundancy. It's a challenging property as nothing is fed but still highly robust in nature. Various methods are categorised under this:

a) Sentence extraction

As the name suggests sentences are extracted but this is done by using the chunks. From each and every sentence of the document the chunks are taken out. Then those chunks are used and is moulded to give a new form called as summary. These summaries are shorter in length briefly describing the whole content. These are also useful for large documents. Shen et al. used labels 0 and 1 for sentences to prvide them labelling [22]. For this labelling a method named Conditional Random Framework (CRF) is used. BAYESUM is a method which basically focuses on the query expansion and is proposed by Daume and Marcu .

b) Sentence Abstraction

Sentence abstraction basically deals with the important sentences. This portrait the best out of the document having much quality. Highly compressed parts are taken out. Knight and Marcu focused on making new sentences and thus overcome this intense problem. They used the two renown methods called using noisy-channel framework and decision-based model.

c) Supervised Approaches

The features are taken out by using the guide. The guide supervises the extracts or the portions to be taken out for further usage. Those sentences which are suggested by the guide are used and are thus moulded into a new form. Bravo-Marquez and Manriquez used this supervision technique and thus extracted the highly ranked sentences by giving or providing the training to ranking functions. ROUGE evaluation is one of the best measures that is used to compared the sentences with the different summary using these ranking techniques.

d) Unsupervised Approaches

As no such supervision is allowed here. So there is no requirement of giving input to generate a summary. But still the matched words are used. Hoffman used an unsupervised learning method called Probabilistic Latent Semantic Indexing (PLSI). This has solid statistical foundation. Blei et al gave a model named Latent Dirichlet allocation (LDA). This model has a hierarchical structure which is highly comprehensive in nature. Over an underlying set of topics every item is modeled into a finite mixture from a collection. Topic wise probabilities are taken out which provides the whole nature of the document.

2.4 Other Summarization Forms

Further there are more other forms as well as challenges in the wide field of automatic text summarization. So the overview can be taken from below stated aspects: -

1.Language: - Normally, original text is monolingual, having removal for the "Evaluation workshop" which worn both English and Arabic input. The output language is generally look alike as the information dialect. For "cross-lingual and multi-lingual summarization," however, output may be in a language peculiar from the input.

2.Genre: - Basically in today's era the input fed to the summarizer is almost real in nature. Very little research carried on "explicit text selection for output," commonly the input classifier has resolved output classifier.

3.Length: - The length also performs a vital performance, such as "summarizing a short story requires a much more effective condensation than summarizing a news article."

4.Structure of the document: - Document structure, like "section headings, tables, quotations, or domain-specific orderings of information," are much more helpful to acknowledge at the time of content analysis.

5.Format of the output: - the small content called summary commonly came across the "fluent text", but for some purposes fragmentary summaries, i.e. lists of words or phrases, may be adequate." The discarded text are the formats or patterns of a summary which is search engine uses so as to furnish a reference for the visible search results. Likewise, there are more structures like "features, key phrases or tag clouds."

2.5 Characteristics of Summarization

Frequency-based: recurrence of words or key terms, nearness, and location inside the content.

Knowledge-based: by and large rely on upon rich information sources to translate the calculated structure of the content. Learning based methodologies are typically extremely information serious and domain specific.

Discourse-based: hypotheses of content cohesion and coherence of sentence in the amount they push the points of confinement of content comprehension and the multifaceted nature and also automation of that processing.

CHAPTER 3: RESEARCH METHODOLOGY

3.1 AUTOMATIC TEXT SUMMARIZATION

Automated summarization system emerged as most important research area in the Natural Language Process (NLP) community for last half century. A variety of text automated summarization systems have been proposed. Radev et al. (2002) define summary in such a way "*a text that is produced from one or more texts, that conveys important information in the original text(s), and that is no longer than half of the original text(s) and usually significantly less than that*". Here are three important aspects that characterize research on automatic text summarization:

- ✓ Summaries may be produced from a single document or multiple documents,
- ✓ Summaries should preserve important information,
- ✓ Summaries should be short

3.1.1 Approaches

Text Summarization systems need to generate a concise and fluent text summary from input, provided that all the key information in the input preserved. These summarization systems need to identify the most important sentences in the input, which can be either a single document or a cluster of related documents as a input, and club all of them together to form a cognate summary.

Generally, there are two approaches to automatic text summarisation

1) **Extractive approach**
2) **Abstractive approach**

Extractive Approaches

Extractive methods [1][2][3] here main focus to pick out most relevant sentences [4][5] in the document and try to maintain low redundancy of information in final text summary. In this approach, most important and crucial document regions (phrases, sentences, paragraphs) are ranked high. All the highest ranked regions all across the documents can then be string and re-ranked using similarity measures to minimize redundancy within final summary.

There are various approaches to implement extractive summaries as follows:

Term Frequency-Inverse document frequency [1]: In this method there are two measures namely term frequency (tf) and documents frequency (df) are calculated corresponding to each non-stop-word (w) within the input document. Term frequency (tf), which gives the count for multiple appears of a word in the input text. Documents frequency (df) give the information in how many documents this particular word appeared. And then Thematic words are formed by comparing the ratio between two frequencies (i.e. td & df), and referred this (if*idf) as measure. Based on it the importance of sentence is calculated. If any sentences contain Thematic word, then its importance considered to be high.

Classical method [2]: In this approach, the scoring of the sentences based on these four factors namely: contains (i) high-frequency key words, (ii) pragmatic words (cue words), (iii) title and heading words and (iv) sentence location (what is current position within the document). The score is calculated based on all these words.

MEAD [7][8] is a centroid based, calculate the centroid of the words of a sentence within the document, which used for document classification and identification salient sentences within a cluster of the documents. Centroid calculated based on three factors (i) centroid value (Ci), (ii)positional value (Pi) and (iii) first- sentence overlap (Fi), then overall score (Si) for a sentence i^{TH} will weighted sum of these three parameters. If there are cluster of "d" document
with "n" sentences and compression rate (r) as input. So there will "n*r" sentence in summary.

Graph theoretic in this extractive summarization model, used to identify themes in the document. In following steps namely, stop word removal and stemming are done before, to generate graphical view of the input documents. All Sentences in the documents form nodes of an undirected graph. If some sentences share some words or have words in common, this can be shown on undirected graph by drowning an edge among the nodes, or whose (cosine, or such) similarity is above some threshold. Portioning of undirected graph gives us the distinct words, these words covered distinct topics. To generate summary, we have to use all these words.

Machine learning techniques [10] [11] [12] all these techniques used for classification of sentence as summary sentences or non-summary sentences. all done by application of statistical techniques.

Abstractive Approaches

An abstractive summary is a caprice text that describes the contexts of the input document. The process of abstractive summarization consists of "understanding" the original text and "re-telling" it in fewer its own words. Its incorporates linguistic methods to examine and interpret the input text and then to locate new concepts and expressions for best possible describe of it by generating a new shorter text that must conveys the most important information from the input document. This may seem the best way to construct a summary (and this is how human beings do it), in real-life scenario immaturity and less expertise in linguistic technology for text analysis and generation presently, by which all these methods practically infeasible.

3.1.2 Working of Summarization Systems

Summarization systems accept one or more documents as input and try to generate a concise and fluent summary which contain all important information of the input document. selecting the most important information requires the ability to understand the semantics of written or spoken input documents. Writing a clear, concise and fluent summary requires the potential to reorganize, modify and merge the information manifested in different sentences in the input documents.

Full elucidation of document and generation of abstracts is often troublesome for people [2], and is positively past the cutting edge for automatic summarization. How then do present automatic summarizers get around this quandary? Most current systems avoid full elucidation of the input documents and generation of eloquent output. The present cutting edge the majority of the cases relies on sentence extraction.

The extractive approach to summarization emphasises research on one key question: how can a system determine whether sentences are important or not? Over the years, the field has seen propels in the refinement of dialect handling and machine learning techniques that decide significance. In the meantime, there have been latest advances in the field which takes ahead semantic translation and generation of synopsis dialect. Semantic elucidation has a tendency to be accomplished for specific Summarization.

For instance, frameworks that produce biographical summaries or summaries of medical documents tend to utilize extraction of data than extraction of sentences. Research on generation for summarization utilizes another type of generation, content to-content generation and spotlights on altering information contents to better fit the requirements of the final summary

3.1.3 Features of Extractive Summarization

The features of extractive summarization can be divided in four types of features i.e. surface, content, event and relevance features, which will be analyzed systematically [24].

Surface Features: Surface features are based on structure of archives or sentences, incorporating sentence position in the record, the quantity of words in the sentence, and the quantity of cited words in the sentence.

Content Features: Content features incorporate three surely understood sentence highlights taking into account content-bearing words i.e., centroid words, signature terms, and high recurrence words. Both unigram and bigram representations have been examined.

Event Features: An event is included an event term and all related event elements. In this study, we pick verbs (for example, "choose and fuse") and action nouns (for example, "decision and joining") as event terms that can describe activities.

They identify with "did what". One or more related named entities are considered as event elements. Four sorts of named entities are at present under thought. [14] [15]

An event map is then built assembled on event instances or concepts. PageRank algorithm is utilized to allot weight to every node (an instance or concept) in the event map. The entirety of weights of a sentence is the sum of weights of event occurrences contained in the sentence

Relevance Features: Relevance features are consolidated to exploit inter-sentence connections. It is accepted that: (i) sentences identified with vital sentences are vital; (ii) sentences identified with numerous different sentences are essential. The primary sentence in a document or a paragraph is essential, and other sentences in a document are compared with the main ones. Another approach to exploit sentence importance is to build a sentence map. Each two sentences are regarded relevant if their similarity is above a threshold. Each two pertinent sentences are associated with a unidirectional link. Based on this map, PageRank algorithm is applied to evaluate the significance of a sentence

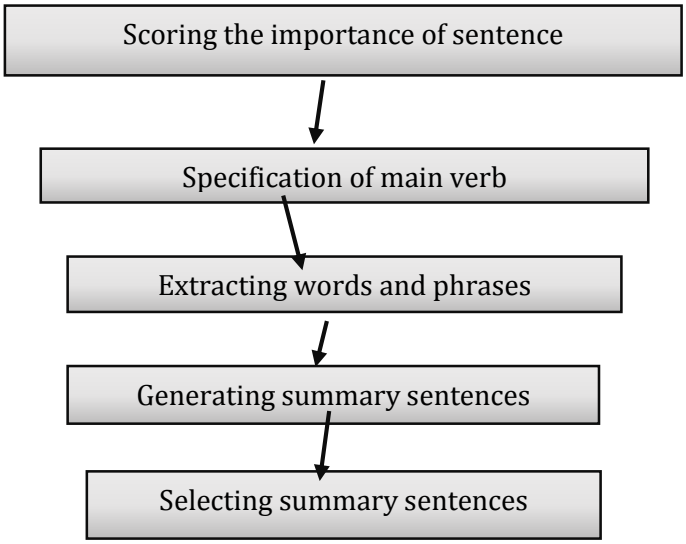


Fig 3.1 The process of auto summarization

3.1.4 Evaluating Summaries

Evaluating the quality of a summary are dreary assignment, costly and require a lot of cautious arranging, Accordingly, these are not that appropriate for framework correlations and assessment during the development. So need of the accessibility of the different measurements. Different measures which analyses against manual summary against the automated summary.

Compression Ratio:

(*how shortest the summary can be generated the actual text)*

Retention Ratio

(*amount of retained information in summary)*

$$\text{Compression Ratio} = \frac{\text{Defined length of the summary}}{\text{Defined actual length of the original doc}}$$

$$\text{Retention Ratio} = \frac{\text{Information Present in the Summary document}}{\text{Informantion present the full actual text}}$$

Intrinsic: - It enumerates the compactness of the system. This can be achieved by comparing to the defined gold standards, either by an illusion system or, it can be man-made by using notify.

Extrinsic: - Extrinsic evaluation enumerates the compactness as well as the effectualness of the summaries generated. It is easy to calculate the instructions which can be present in the summary to a possible extent. Various other measurable tasks which can be performed are information gathering, the time taken and effort required for summary by system also the system's impact say for example feedback for system

Text Quality Measures

There are a few parts of text (linguistic) quality:

Grammatical – the text ought not contain non-textual items (i.e., markers) or accentuation mistakes or erroneous words

Non-redundancy – the text ought not contain excess data

Reference clarity – the nouns and pronouns ought to be unmistakably alluded to in the summary. For example, the pronoun he needs to mean somebody with regards to the summary.

Coherence and structure – the summary ought to have great structure and the sentences sought to be coherent.

3.2 TEXT SUMMARIZATION TECHNIQUES

3.2.1 Pattern Matching

Pattern matching is follow conditional branching which allows you to concisely match on data structure patterns and bind variables at the same time This is also known as the act of checking a given **sequence** of tokens for the presence of part of some pattern. In contrast to pattern recognition, the match usually has to be exact same. Pattern matching provides a way to "dispatch control" based on structural properties of single value.

The patterns generally grouped up either as sequences or tree structures. Uses of pattern matching include depict the locations (if any) of a pattern within a sequence of token, to show some component of the matched pattern, and to replace the matching pattern with some different token sequence (i.e., search and replace). For calculating the importance of the sentence, first of all define and construct identifiers, identifiers should be based on grammatical aspects of English language pertaining or any language to which information to be extracted from the given text.

As per requirement or interest, there is need to extract the information from the corpus of English language regarding to its tense form, voice form, speech form, etc. So firstly have to construct the proper identifiers.

As all know a sentence of English language can be belonging to a particular tense, voice, speech etc. So classify all these sentences into particular classes for which they should construct some identifiers to these particular classes.

Given text is firstly divided into the sentences; here each sentence has some mean which lies in a single sentence it can be located with the help of sentence boundary detection. After that need to construct a matrix whose every *column* shows a particular *sentence* and the entries of such column are as per the possibility or not for the identifiers.

Thus these sentences are arranged in a matrix, have number of rows and number of columns are equal to the number of *identifiers (t)* and *sentences (d)* respectively. This matrix is thus said to be term by sentence matrix of order t×d. The presence and absence of a particular identifier in a document is defined as TiDj = 1; if **i**th identifier present in the **j**th document Otherwise the value of Ti Dj is 0 respectively. Thus in general the sentence may differ in grammatical aspects can be combine together and gives any other sentence of the text possess all aspects simultaneously, provided that they don't fall in the same class.

Pattern-Matching Algorithm

Get values for n and m, the size of the text and the pattern, respectively
Get values for both the text $T_1\ T_2 \ldots T_n$ and the pattern $P_1\ P_2 \ldots P_m$
Set k, the starting location for the attempted match, to 1
While ($k \leq (n - m + 1)$) do
    Set the value of i to 1
    Set the value of *Mismatch* to NO
    While both ($i \leq m$) and (*Mismatch* = NO) do
        If $P_i \neq T_{k+(i-1)}$ then
            Set *Mismatch* to YES
        Else
            Increment i by 1 (to move to the next character)
    End of the loop
    If *Mismatch* = NO then
        Print the message 'There is a match at position'
        Print the value of k
    Increment k by 1
End of the loop
Stop, we are finished

Fig. 3.2 Pseudo code of Pattern Matching Algorithm

3.2.2 K-Mean Clustering

K-means clustering is a method follow vector quantization approach, originally used for signal processing, that is also popular for cluster analysis in the domain of data mining. K-means clustering aims to partition of **n** observations into **k** clusters in which each new entry belongs to the cluster with the nearest mean (*distance from this point to mean or centre of the nearest cluster*), serving as a prototype of the cluster.

Clusters are nothing but combing the similar sentences together. Thus there are many clustering technique, one of them technique is K-Mean clustering. The main logic or idea to use K-Mean clustering is simple, there is to club all the similar set of sentences together by cumulative similarity, and divide the document into k-clusters to locate k centroids for each cluster. These centroids are placed in different location (not arranged properly) as per the centroid calculation result.

Now, proceed to next step to place them properly as per the given data and to group them with nearest centroid. And Just repeat this step until the complete grouping is done to the entire text file.

At this point need to re-calculate ‘k’ new centroids as centre of previous step clusters. All These ‘k’ new centroids calculated on the new data set points for nearest new centroid. As this repeatedly generate the k- centroids change their location step by step until no more changes can be done.

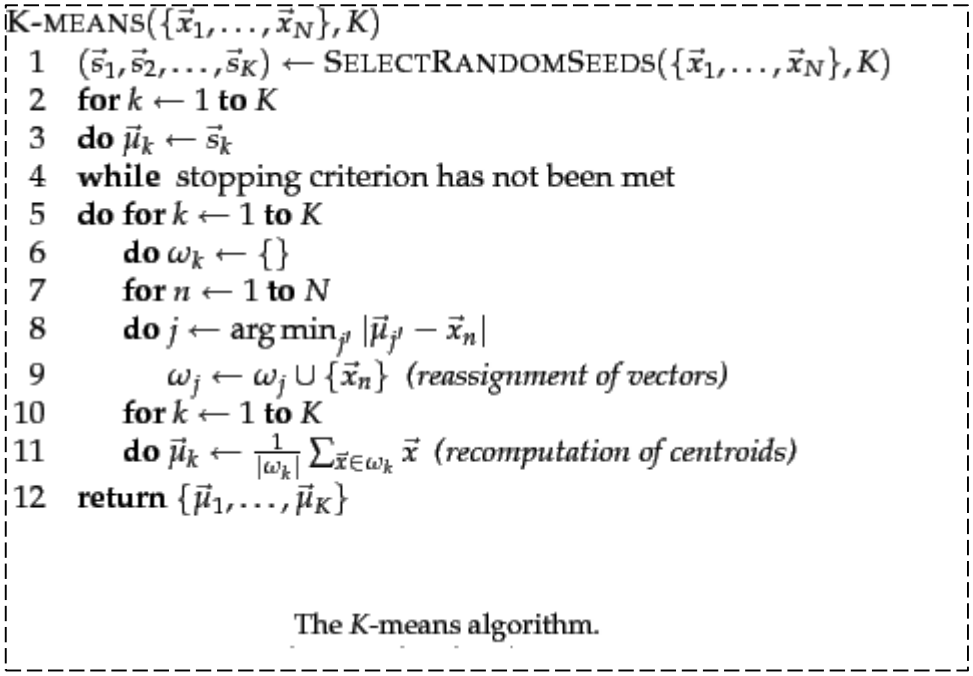


Fig. 3.3 Pseudo code of K-Mean Algorithm

3.2.3 Reduction Algorithm

Reduction is an algorithm which automatically summarizes a text by extracting the sentences which are deemed to be most important. Extraction of sentences based on the corpus of some commonly used dictionary words [21]

There are five steps in the reduction program:

1. **Syntactic Parsing:**
   First of all, parse the input sentence using the parser and produce the parse tree, add or annotates this parse tree with additional information.
2. **Grammar Checking**
   To check components of the sentence for grammatical correctness and insure how many sentences are correct. By traversing the parse tree in top-down order.
3. **Context information**
   To decides which content in the sentence are much related to the main phrase or heading on which whole document described. To measure the necessity of a phrase locally, the system worked on lexical links between words.

4. **Corpus evidence**

   Here is collection of the sentences reduced by human experts and professional and original sentences too for phrase checking.

5. **Final Decision**

   This step's decision for reduction based on all previous steps. In this program parse all annotated tree in top-down fashion and decide which phrases are to be removed, reduced or leave unchanged based on the information provided by earlier steps

Example 1:
Original sentence : *When it arrives sometime next year in new TV sets,* **the V-chip will give parents a new and potentially revolutionary device to block out programs they don't want their children to see.**
Reduction program: The V-chip will give parents a new and potentially revolutionary device to block out programs they don't want their children to see.
Professionals : The V-chip will give parents a device to block out programs they don't want their children to see.

3.2.4 Frequency Based Algorithm

First perform some pre-processing of input, means remove the stop words from the document. Then count the frequency of each word in processed text input file by comparing select word with each word. Then select the keyword, having highest frequency among all. After that select the sentences which have these keywords for final summary.

I. Term Frequency

The based Term frequency is very important feature. TF represents how many times the term appears in the input text or document (usually a function or algorithm is used for compression square root of it) to calculate the term frequency.

Some terms used to identify the boundary of the sentences in the documents based on punctuation such as " **. / , " , [, {"** etc. and split into sentences. All these are known as tokens.

II Keyword Frequency

All keywords oppose the highest frequency words in term sentence frequency. After computing the frequency of each world in entire documents. There some words have the highest frequency all these are called *keywords*.

The score of the words as keywords, based on this feature, in document if sentence in the document is have keywords then its score to be number of keywords it contains, where the sentence receives 0.1 score for each key word.

III Stop Word Filtering

In documents, there will be many words that appear regularly but provide little or no extra meaning to the document. Some words such as '*the*', '*and*', '*is*' and '*on*' are very frequently in the English language and most documents will contain many instances. These words are generally not very useful when will searching; they are not normally not part of user query what users are searching for when entering queries.

3.2.5 TextRank Algorithm

TextRank is a text summarization model based on graph-based ranking algorithm using for extraction of the sentences from natural language texts. The essential way using this algorithm of deciding the importance of the vertex within a graph, based on or available global information recursively drawn from the complete graph.

The basic thought of implementation by graph-based ranking model is that of "voting" or "recommendation". If there a vertex link (sentence) which is links to another one, it is basically casting a vote for that other vertex. Higher the votes for any vertex higher will be its importance over the other ones. The edges between the sentences generally based on the semantic similarities or content overlaps. All this makes some intuitive sense and allows the algorithms to be applied to any arbitrary new text/input [19].

This algorithm also uses Google's PageRank algorithm to get resulting graph same as in the LexRank algorithm.

A summary is formed from top ranking sentences during PageRank analysis. This implementation uses Levenshtein Distance as the relation between two or more text units[20]

TextRank Formula

$$S(V_i) = (1-d) + d \cdot \sum_{j \in In(V_j)} \frac{1}{Out(V_j)} S(V_j)$$

Where: S(Vi): TextRank, Vj: links: dumping factor 0.85, out (Vj): No. of out links on that page

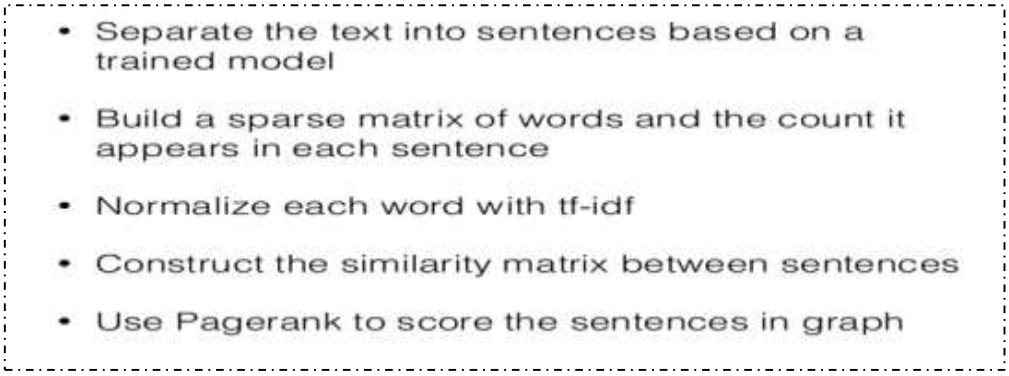


Fig. 3.4 Procedure for summarization using TextRank Algorithm

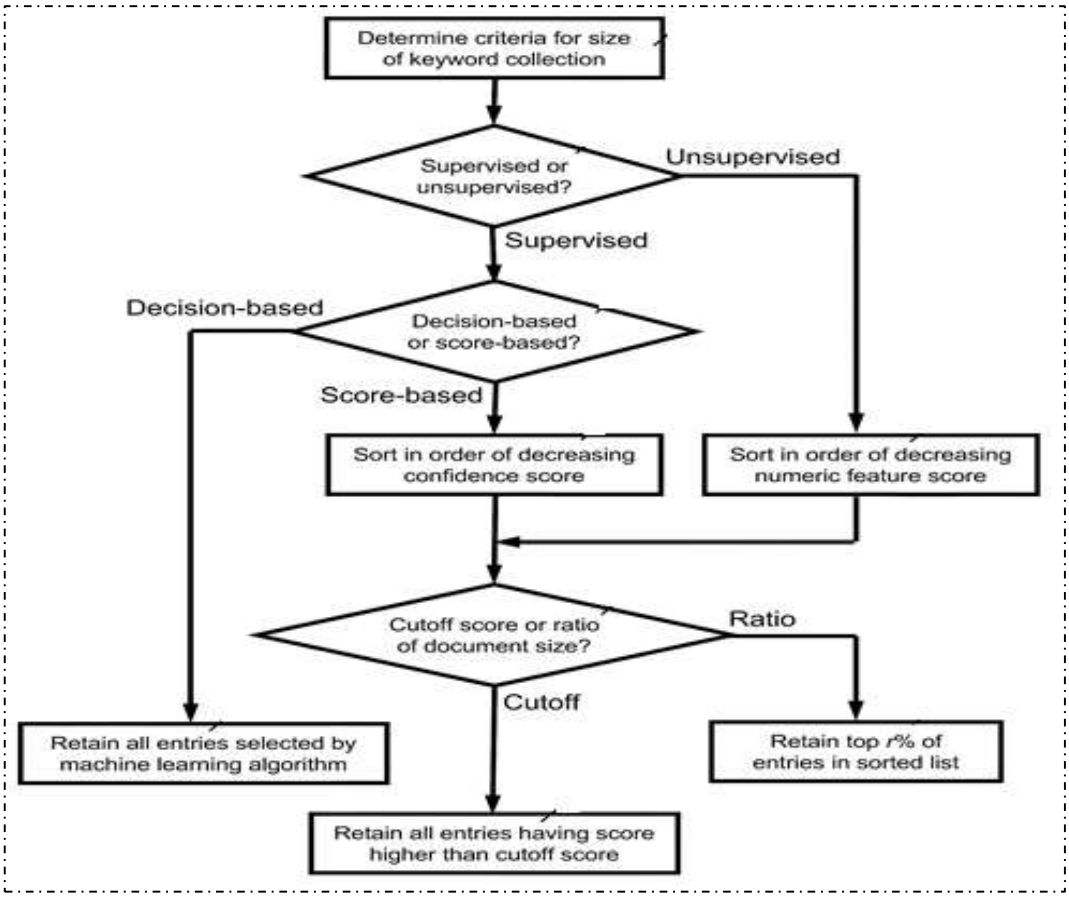


Fig. 3.5 Flow chart for summarization using TextRank Algorithm

3.2.6 LexRank Algorithm

The LexRank, this is the new approach for computing the importance of any sentence based on the concepts of the eigenvector centrality in a graphical representation of sentences. In this model, a connectivity matrix and intra-sentence cosine similarity is also used as the adjacency matrix of the graph representation of the sentences.

Similarity graph is use for the calculating the centrality. LexRank explored using the unweighted edges after application of the threshold to all cosine values, and can use edges with weights equal to the similarity score. This method is domain-independent and portable. One could think of the features pointing important sentences in the news domain might less possibility may differ from the biomedical domain [8].

LexRank applies a heuristic for post-processing steps that builds up a summary by appending sentences in rank order, but discards any sentences that are too similar (all duplicates) to ones already placed in the summary. And also used as Cross-Sentence Information Subsumption (CSIS). If one sentence in the input text, which is very similar to many others, it possibly be a sentence of great importance, will be in the summary text. [18]

```
input : An array S of n sentences, cosine threshold t
output: An array C of Centroid scores
Hash WordHash;
Array C;
/* compute tf×idf scores for each word */
for i ← 1 to n do
    foreach word w of S[i] do
        WordHash{w}{"tfidf"} = WordHash{w}{"tfidf"} + idf{w};
    end
end
/* construct the centroid of the cluster */
/* by taking the words that are above the threshold*/
foreach word w of WordHash do
    if WordHash{w}{"tfidf"} > t then
        WordHash{w}{"centroid"} = WordHash{w}{"tfidf"};
    end
    else
        WordHash{w}{"centroid"} = 0;
    end
end
/* compute the score for each sentence */
for i ← 1 to n do
    C[i] = 0;
    foreach word w of S[i] do
        C[i] = C[i] + WordHash{w}{"centroid"};
    end
end
return C;
```

Computing Centroid scores.

Fig. 3.6 steps for calculating Computing the centroid Scores of sentence and cluster too.

input : A stochastic, irreducible and aperiodic matrix $\mathbf{M}$
input : matrix size N, error tolerance ϵ
output: eigenvector $\mathbf{p}$
$\mathbf{p}_0 = \frac{1}{N}\mathbf{1}$;
t=0;
repeat
    t=t+1;
    $\mathbf{p}_t = \mathbf{M}^T\mathbf{p}_{t-1}$;
    $\delta = ||\mathbf{p}_t - \mathbf{p}_{t-1}||$;
until $\delta < \epsilon$;
return $\mathbf{p}_t$;

Power Method for computing the stationary distribution of a Markov chain.

Fig. 3.7 Steps for Calculating the values of Eigen vector

```
MInputAn array S of n sentences, cosine threshold t  output: An array L of LexRank
        scores
Array CosineMatrix[n][n];
Array Degree[n];
Array L[n];
for i ← 1 to n do
    for j ← 1 to n do
        CosineMatrix[i][j] = idf-modified-cosine(S[i],S[j]);
        if CosineMatrix[i][j] > t then
            CosineMatrix[i][j] = 1;
            Degree[i] + +;
        end
        else
            CosineMatrix[i][j] = 0;
        end
    end
end
for i ← 1 to n do
    for j ← 1 to n do
        CosineMatrix[i][j] = CosineMatrix[i][j]/Degree[i];
    end
end
L = PowerMethod(CosineMatrix,n,ϵ);
return L;
```

Computing LexRank scores.

Fig. 3.8 Steps for Calculating the LexRank using LexRank algorithm.

LexRank simply focused on summarization, but could just as easily be used for key phrase extraction or any other NLP ranking task.

3.3 EVALUATION MEASURES

To assess the worthiness of system that going to be used, then evaluation measures came to into the picture. By using, effectiveness of algorithm used in summarization is a difficult task and itself is a separate research field in Natural Language Processing (NLP). The quality of the summary we getting can be evaluated in different aspects: (1) selected contents importance, and the overall presentation quality.

Presentation quality itself in described into two aspects: grammatical correctness and coherence. Since there are extraction sentences from the original text based its importance, the grammatical correctness in sentences is assured, and to be as good as the source document's grammatical correctness. Coherence in our solution is a problem as our algorithm does not consider repetition or phrase and information ordering. Since deciding what is more important in an input is a subjective task, an evaluation method is evaluation of summaries based on human judges. However, just comparing the contents of automatically generated summaries with human extracted summaries it looks a fairer methodology [22].

All this evaluation done automatically using distributed similarity techniques. The similarity between the referenced summary and that of system output reflects the summary quality. Recall between the system generated and the referenced summaries is used for this reason. In the evaluation method, it is better to use multiple referenced summaries by different summarizer, since summarization is a subjective task.

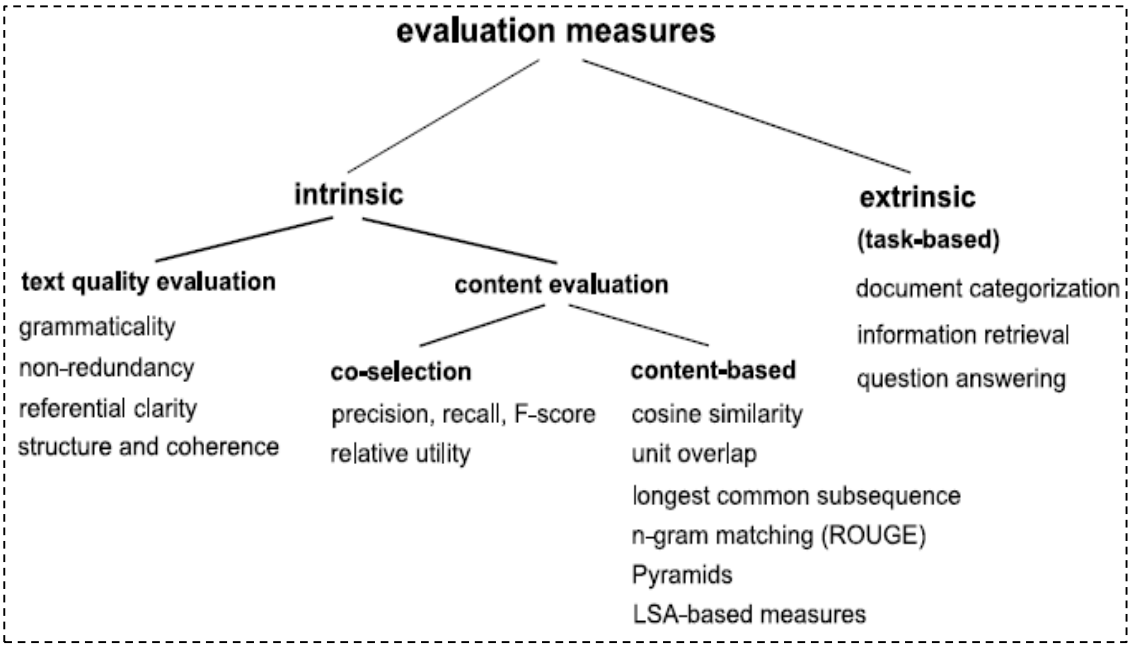


Fig. 3.9 The taxonomy of summary evaluation measures

Core evaluation metrics used in co-selection are precision, recall and F-score. Precision (P) is the number of sentences appearing in both generated (by ATS system) and reference summaries divided by the number of sentences in the summary generated by system. Recall (R) is the number of sentences appearing in both system and reference summaries divided by the number of sentences in the referenced summary [23].

F-score is a composite measure of precision and recall. F-score (*iii*) is nothing but harmonic average of precision (*ii*) and recall (*i*):
These Metrics are respectively given in the Equations (i), (ii) and (iii).

$$Recall = \frac{RelevantSentences \cap RetrievedSentences}{Relevant\ Sentences} \quad \text{(i)}$$

$$Precision = \frac{RelevantSentences \cap RetrievedSentences}{Retrieved\ Sentences} \quad \text{(ii)}$$

$$F - score = \frac{2 * Precision * Recall}{Precision + Recall} \quad \text{(iii)}$$

Below is a more complex formula for measuring the F-score:

$$F - score = \frac{(\beta 2 + 1) \cdot P \cdot R}{\beta 2 \cdot P + R}$$

where β is a weighting factor that favours precision when $\beta > 1$ and favours recall when $\beta < 1$.

In pattern recognition and all techniques binary classification, precision (also called positive predictive value) is the fraction between retrieved instances to instances that are relevant, while recall (also known as sensitivity) is the fraction between relevant instances to instances that are retrieved. Both precision and recall are based on understanding and measure of relevance. In other words, high precision means that an algorithm generated substantially more relevant summaries than irrelevant, while high recall means that an algorithm generated most of the relevant summaries.

In classification tasks, the precision for a category is the number of true positives (i.e. the number of sentences correctly labeled as belonging to the positive class (*summaries*)) divided by the total number of sentences labeled as belonging to the positive class (*summaries*) (i.e. the sum of true positives and false positives, which are items incorrectly labeled as belonging to the class). Recall in this context is defined as the number of true positives divided by the total number of sentences that actually belong to the positive class (*summaries*) (i.e. the sum of true positives and false negatives, which are sentences which were not labeled as belonging to the positive class (*summaries*) but should have been) as shown in figure below.

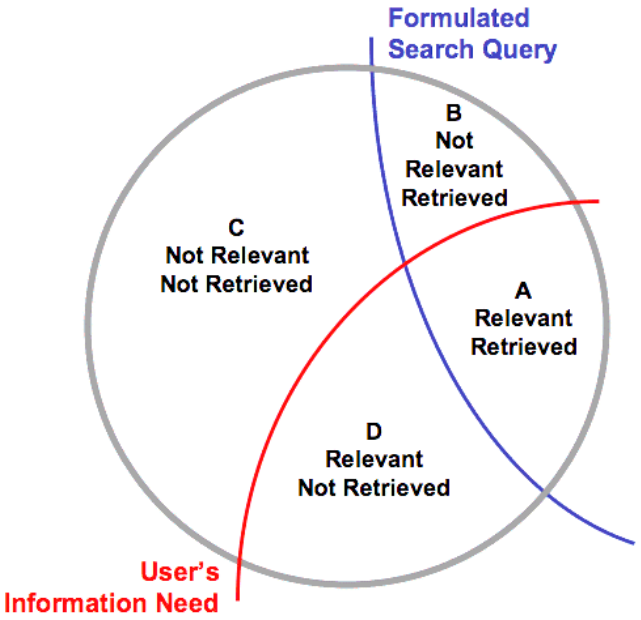


Fig. 3.10 Co-relation between the system retrieved and input information visually

Since the calculation of these measures is difficult and time-consuming also the work is totally based on machine leaning algorithms, therefore automated software procedure or algorithm is used for this evaluation. This development system is a tools designed to aid software developers-whether they are novices or seasoned professionals-face complex challenges and create innovative solutions. Different platform or environment used to improve the process of development to make the work of achieving those breakthroughs easier and more satisfying.

CHAPTER 4: EXPERIMENTATION

4.1 DATASET COLLECTION

This is the Timeline17 dataset used for experiments on summarization. Briefly, the dataset consists of 17 manual-created timelines each associated with article, news articles. They belong to 9 news topics namely: BP Oil Spill, Michael Jackson Death (~Dr. Murray Trial), Haiti Earthquake, H1N1 (Influenza), Financial Crisis, Syrian Crisis, Libyan War, Iraq War, Egyptian Protest.

Each timeline and news articles belongs to one news agency, such as BBC, Guardian, CNN, Foxnews, NBCNews, etc. The contents of these news are in plain text file format and noise filtered.

e.g.

The statistics about our dataset

<Topics > , < Source> , <#Docs> <#Ground Truth>, <#Dates> <Average Sentence per date of manually created timeline>, <#Since>

BP Oil , Washington Post , 296 , 1 , 12 , 1.6 , 2010
BP Oil , Reuters , 298 , 1 , 16, 1.9 , 2010
BP Oil , BBC , 293 , 1 , 48 , 2.0 , 2010
BP Oil , Foxnews , 286 , 1 , 13 , 4.0 , 2010
BP Oil , Guardian , 288 , 1, 102 , 3.0 , 2010
Michael Jackson death ,BBC , 142 , 1 , 38 , 2.1 , 2009
Haiti Earthquake , BBC , 296 , 1 , 11 , 7.8 , 2010
H1N1 , Reuters , 207 , 1 , 15 , 1.5 , 2009
H1N1 , BBC , 122 , 1 , 7 , 4.6 , 2009
H1N1 , Guardian , 76 , 1 , 12 , 2.8 , 2009
Financial Crisis , WP , 298 , 1, 65 , 6.9 , 2008
Syrian Crisis , Reuters , 346 , 3 , 76 , 1.6 , 2011
Syrian Crisis , BBC , 308 , 1 , 13 , 2.4 , 2011
Libyan War , Reuters , 379 , 1 , 28 , 1.3 , 2011
Libyan War , CNN , 398 , 1 , 38 , 2.1 , 2011
Iraq War , Guardian , 344 , 1 , 155 , 2.6 , 2005
Egyptian Protest , CNN , 273 , 1 , 20 , 2.8 , 2011

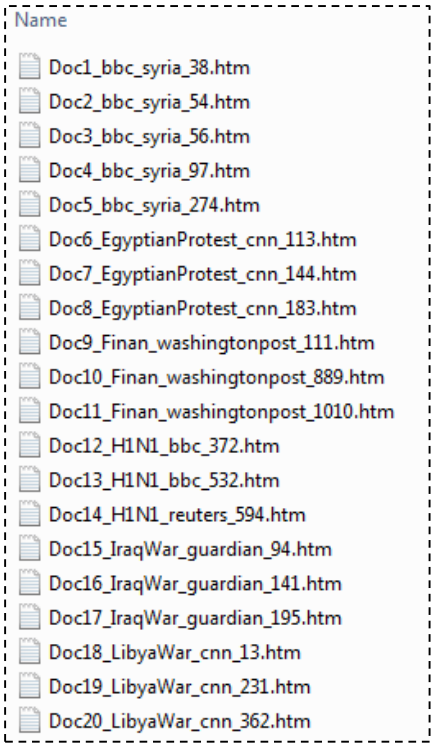


Fig. 4.1 List of input file selected for the experimentation from the new corpus "timeline17"

An example original document in this data set is shown in paragraph below. "*bbc_syria_97.htm.txt"*

> *"Syrian unrest: Aleppo clashes spark rise in refugees Rebel fighters have vowed to retake territory in Aleppo after retreating on Thursday Syrian civilians are fleeing the city of Aleppo in increasing numbers, the UN says , amid a lull in fighting between rebels and government forces .The UN refugee agency says it has now registered almost 150,000 refugees in four neighbouring countries . It said 6,000 had crossed into Turkey in the past week , many of them from the Aleppo area . Earlier , the UK announced it was giving an extra # 5m -LRB- $ 7.8 m -RRB- in non-lethal equipment to the Free Syrian Army .*
>
> *Foreign Secretary William Hague said the support would include more radio and satellite equipment , as well as portable power generators .*
>
> *In another development on Friday , the US accused the Lebanese militant group Hezbollah of providing `` training , advice and extensive logistical support '' to the Syrian regime and added further sanctions against the group . Hezbollah is designated as a terrorist organization by the US and is already under heavy sanctions .*
>
> *The US state department also imposed sanctions on Syria 's state-run oil company , Sytrol , for conducting business with the Iranian energy sector . Adrian Edwards , spokesman for the UN refugee agency -LRB- UNHCR -RRB- , told reporters : `` There certainly in the past week has been a sharp increase in the numbers arriving in Turkey , and many of the people are coming from Aleppo and surrounding villages . `` If you look at other areas , I think that the situation is*

more of a steady and continued increase , but where fighting happens we tend to see the consequences , '' he said .The agency says there are now 50,227 refugees registered in Turkey ; 45,869 in Jordan ; 36,841 in Lebanon and 13,730 in Iraq . `` In several countries we know there to be substantial refugee numbers who have not yet registered , '' Mr Edwards said .

In Aleppo , rebels vowed to fight back after being largely ousted from the strategic south-west district of Salah al-Din on Thursday . The Free Syrian Army withdrew after intense shelling by tanks and jet fighters . William Hague : `` Aid would include providing trauma and medical supplies , communications equipment and body armor '' `` We will not let Salah al-Din go , '' FSA commander Abu Mohammed told AFP by telephone . UK-based activist group , the Syrian Observatory for Human Rights , said government forces had also bombed Sakhur and Hananu in the east of the city .

Journalists from Reuters news agency reported seeing residents streaming out of Aleppo with cars packed with possessions , taking advantage of the calm spell .AFP news agency reported that a bakery in the eastern Tariq al-Bab district had been hit by a shell , killing about 12 people and injuring at least 20 . State news agency Sana also reported that government forces had repelled a rebel attack on Aleppo 's international airport . The opposition Syrian National Council said part of Aleppo 's 13th-century citadel had been damaged by shelling . Activists also reported fighting in suburbs of the capital , Damascus . Syrian President Bashar al-Assad is resisting international pressure to step down despite months of anti-government protests that have given way to civil war . He has faced a string of high-status defections , including his former Prime Minister Riad Hijab , who fled to Jordan earlier this week . US Secretary of State Hillary Clinton is due to arrive in Turkey later on Friday for talks over the weekend that are expected to be dominated by the Syrian crisis . Hopes for a negotiated settlement in Syria remain alive , despite regional tensions stoked by recent air strikes attributed to Israel , says Jim Muir .”

4.2 STUDY SYSTEM REQUIREMENTS

SOFTWARE REQUIREMENTS

Programming Language : ASP.NET, C#, Python 2.7 .

Database : DUC 2002

Tool : Visual Studio 2013,MS Excel, Python IDE, Virtual env, MSsql.

HARDWARE REQUIREMENTS

Processor	1GHz or over
RAM	1GB or more
HDD	80GB
OS	Windows

4.3 ENVIRONMENT & EXPERIMENT SETTING

Experimented with the news article corpus used in "*timeline17*". To properly evaluate our algorithm, and compare with existing algorithms we have attempted experiments on 20 documents of this dataset. In this task, all summarization systems provide a short summary for each of the 20 articles. Each summary is automatically evaluated against 6 model summaries extracted by "*timeline17*". While calculating scores, words in both the model and the system output are stemmed using Porter Stemmer. These are the values used in "*timeline17*" and we have used the same values to be compatible with their evaluation. We used a newer version of Visual Studio and python 2.7 for our evaluations, thus official scores of "*timeline17*" and our scores may differ in small quantities.

In the following experiments for independent event-based summarization, we investigate the effectiveness of the approach. In addition, we attempt to test the importance of contextual information in scoring event terms.

The number and type of neighbouring named entities are considered to set the weights of event terms. We also consider a variety of stop words from internet and remove them in next steps within original document after comparison. Weight and other terms within TF-IDF method are optimized using genetic algorithm and further this extracted optimized output is given to proposed multi-layer feed-forward neural network as an output of network and pre-processed data as a input to neural network.

To further progress in summarization and enable researchers to participate in large-scale experiments, the National Institute of Standards and Technology (NIST) continued an evaluation in the area of text summarization called the Document Understanding Conference (DUC).

Summarization evaluation is a very hard task; performance of an algorithm can change dramatically in different summarization settings. It is not possible to say that single method is best for every corpus. There are systems that incorporate different features and techniques into a single algorithm and decide which to use depending on the corpus with machine learning algorithms.

We have seen that our algorithm, partitioned the document into topic segments with acceptable quality. It is possible to say that our algorithm is successful in summarization at least for the domain of news articles. To evaluate the event based summarization approaches proposed, we conduct a set of experiments on "*timeline17*" dataset. Document Understanding Conferences (DUC) is a series of evaluation tasks on text summarization.

They provide a public test benchmark for summarization systems. It contains 30 clusters of documents and a total of 308 English news reports. The number of documents in each cluster is between 3 and 20. The contents of each cluster are about certain news topic, such as

BP Oil , Washington Post ,	296 , 1 , 12 , 1.6 , 2010
BP Oil , Reuters ,	298 , 1 , 16, 1.9 , 2010
BP Oil , BBC ,	293 , 1 , 48 , 2.0 , 2010
BP Oil , Fox news ,	286 , 1 , 13 , 4.0 , 2010
BP Oil , Guardian ,	288 , 1, 102 , 3.0 , 2010
Michael Jackson death ,BBC ,	142 , 1 , 38 , 2.1 , 2009
Haiti Earthquake , BBC ,	296 , 1 , 11 , 7.8 , 2010
H1N1 , Reuters ,	207 , 1 , 15 , 1.5 , 2009
H1N1 , BBC ,	122 , 1 , 7 , 4.6 , 2009
H1N1 , Guardian ,	76 , 1 , 12 , 2.8 , 2009
Financial Crisis , WP ,	298 , 1, 65 , 6.9 , 2008
Syrian Crisis , Reuters ,	346 , 3 , 76 , 1.6 , 2011
Syrian Crisis , BBC ,	308 , 1 , 13 , 2.4 , 2011
Libyan War , Reuters ,	379 , 1 , 28 , 1.3 , 2011
Libyan War , CNN ,	398 , 1 , 38 , 2.1 , 2011
Iraq War , Guardian ,	344 , 1 , 155 , 2.6 , 2005
Egyptian Protest , CNN ,	273 , 1 , 20 , 2.8 , 2011

Table. 4.1 List of the all datasets available in “timeline17”

CHAPTER 5: RESULT & ANALYSIS

5.1 Results

Pattern Matching

	Pattern Matching		
*Length in # words			
Text*	**Precision**	**Recall**	**F-Score**
1250	0.069934	0.143585	0.094057
1624	0.082578	0.157753	0.108408
629	0.089448	0.142432	0.109886
4787	0.084596	0.156419	0.109806
658	0.065629	0.152653	0.091794
10326	0.037328	0.145995	0.059455
4365	0.062849	0.153994	0.089266
8693	0.036744	0.147278	0.058815
550	0.072336	0.157683	0.099176
799	0.056808	0.137538	0.080406
3527	0.081215	0.145072	0.104133
4068	0.05159	0.15212	0.07705
6149	0.050518	0.156358	0.076364
2449	0.068049	0.154476	0.094478
2468	0.050661	0.139613	0.074344
1539	0.066911	0.150634	0.092662
7153	0.00794	0.145698	0.01506
4436	0.085732	0.160652	0.111801
5381	0.0546	0.152959	0.080474
2284	0.05649	0.14783	0.081744

Table 5.1 Tabular representation of results (Precision, Recall and F-Score) of Pattern Matching corresponding the Text Input Set.

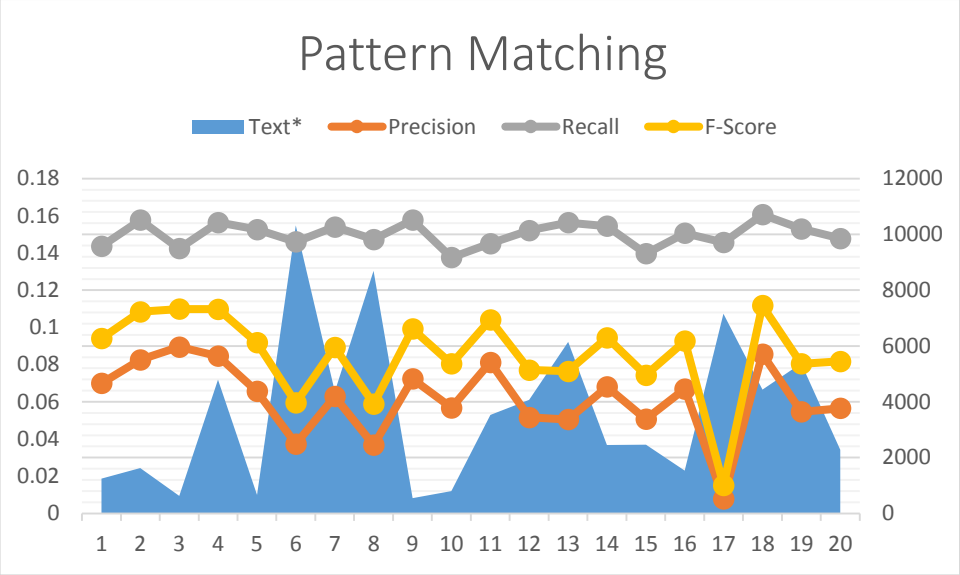


Fig. 5.1 Graph between the Precision, Recall and F-Score corresponding to given input using pattern matching.

K-Mean Algorithm

	K-Mean		
*Length in # words			
Text*	**Precision**	**Recall**	**F-Score**
1250	0.373764	0.125396	0.18779
1624	0.453623	0.133504	0.206294
629	0.574767	0.167469	0.259367
4787	0.422344	0.137767	0.207763
658	0.532623	0.146179	0.229399
10326	0.78679	0.208686	0.329876
4365	0.855167	0.152394	0.258689
8693	0.445113	0.227089	0.300744
550	0.43139	0.132575	0.202819
799	0.479743	0.242981	0.322581
3527	0.288848	0.142465	0.190816
4068	0.597641	0.136873	0.222735
6149	0.441275	0.159264	0.234054
2449	0.385491	0.182004	0.247265
2468	0.565993	0.212941	0.309457
1539	0.686792	0.142053	0.235414
7153	0.422344	0.152654	0.224253
4436	0.618907	0.163004	0.258045
5381	0.538407	0.183003	0.273159
2284	0.3773	0.188859	0.251719

Table 5.2 Tabular representation of results (Precision, Recall and F-Score) of K-Mean corresponding the Text Input set.

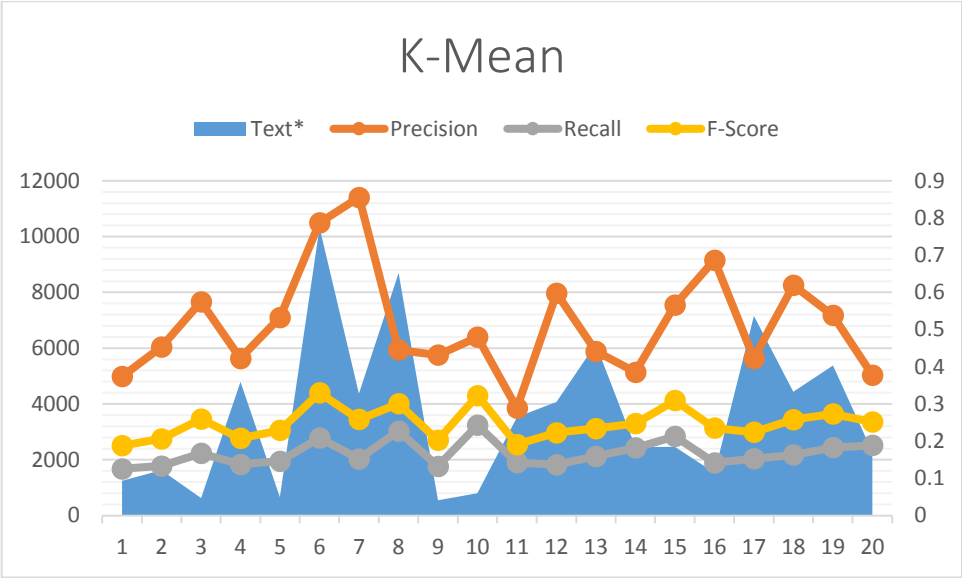


Fig. 5.2 Graph between the Precision, Recall and F-Score corresponding to given input set using K-Mean Algorithm.

Reduction Algorithm

	Reduction		
*Length in # words			
Text*	**Precision**	**Recall**	**F-Score**
1250	0.531165	0.780909	0.632269
1624	0.530781	0.690769	0.600298
629	0.547068	0.869231	0.671508
4787	0.548616	0.825309	0.659101
658	0.518969	0.865672	0.648915
10326	0.548022	0.772973	0.641344
4365	0.537468	0.832432	0.653195
8693	0.544115	0.757746	0.633403
550	0.534132	0.86392	0.660129
799	0.530386	0.790625	0.634872
3527	0.52698	0.893939	0.663075
4068	0.548114	0.763333	0.638064
6149	0.551479	0.786667	0.648405
2449	0.551434	0.760486	0.639304
2468	0.532911	0.815493	0.644592
1539	0.544415	0.758904	0.63401
7153	0.544415	0.758904	0.63401
4436	0.551566	0.727013	0.627252
5381	0.535728	0.725926	0.616491
2284	0.544115	0.807746	0.650225

Table 5.3 Tabular representation of results (Precision, Recall and F-Score) of Reduction corresponding the Text Input set

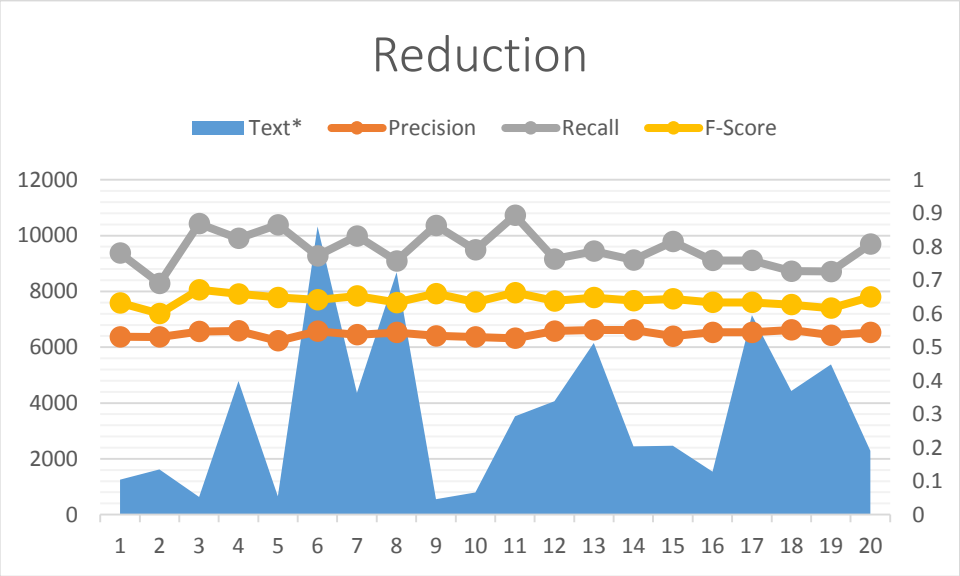


Fig. 5.3 Graph between the Precision, Recall and F-Score corresponding to given input set using Reduction.

Frequency Based

	Frequency Based		
*Length in # words			
Text*	**Precision**	**Recall**	**F-Score**
1250	0.34594	0.424242	0.381111
1624	0.35741	0.446154	0.396882
629	0.45193	0.661538	0.537005
4787	0.33684	0.407407	0.368778
658	0.35049	0.432836	0.387335
10326	0.35731	0.445946	0.396738
4365	0.32822	0.391892	0.357241
8693	0.30507	0.352113	0.326908
550	0.4111	0.56	0.474134
799	0.3362	0.40625	0.36792
3527	0.33776	0.409091	0.370019
4068	0.28519	0.32	0.301594
6149	0.32527	0.386667	0.353321
2449	0.32822	0.391892	0.357241
2468	0.38063	0.492958	0.429572
1539	0.25462	0.273973	0.263942
7153	0.33123	0.39726	0.361253
4436	0.39305	0.519481	0.447507
5381	0.27788	0.308642	0.292454
2284	0.44135	0.633803	0.520352

Table 5.4 Tabular representation of results (Precision, Recall and F-Score) of Frequency Based corresponding the Text Input set

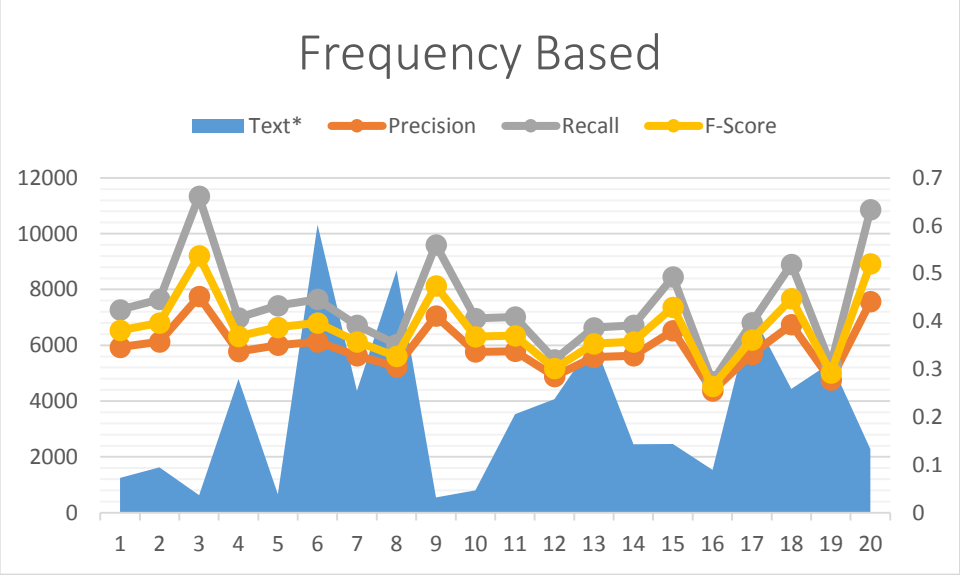


Fig. 5.4 Graph between the Precision, Recall and F-Score corresponding to given input set using Frequency Based

TextRank Algorithm

	TextRank		
*Length in # words			
Text*	**Precision**	**Recall**	**F-Score**
1250	0.490603	0.772727	0.600163
1624	0.52219	0.876923	0.654587
629	0.530781	0.907692	0.669857
4787	0.507609	0.82716	0.629133
658	0.466486	0.701493	0.560347
10326	0.53384	0.918919	0.675344
4365	0.518737	0.864865	0.648506
8693	0.529052	0.901408	0.666767
550	0.486455	0.76	0.593212
799	0.512568	0.84375	0.637725
3527	0.464878	0.69697	0.557742
4068	0.523063	0.88	0.656129
6149	0.537708	0.933333	0.68232
2449	0.522604	0.878378	0.655317
2468	0.529052	0.901408	0.666767
1539	0.518202	0.863014	0.647568
7153	0.521521	0.883714	0.655941
4436	0.508816	0.831169	0.631219
5381	0.518544	0.864198	0.648168
2284	0.500146	0.802817	0.616327

Table 5.5 Tabular representation of results (Precision, Recall and F-Score) of TextRank Algorithm corresponding the Text Input set

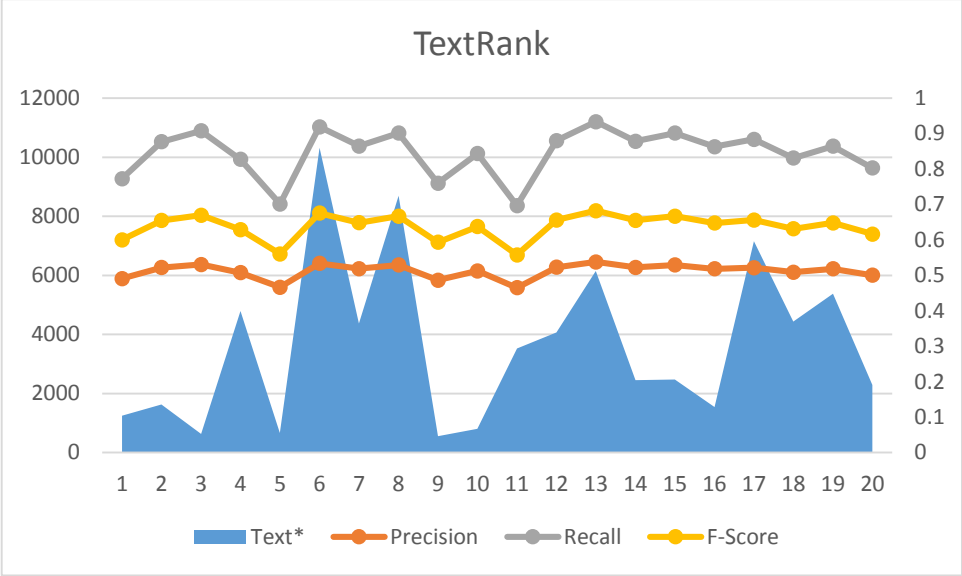


Fig. 5.5 Graph between the Precision, Recall and F-Score corresponding to given input set using TextRank Algorithm.

LexRank Algorithm

	LexRank		
*Length in # words			
Text*	**Precision**	**Recall**	**F-Score**
1250	0.47023	0.712121	0.566432
1624	0.46868	0.707692	0.563905
629	0.42155	0.584615	0.489869
4787	0.39261	0.518519	0.446865
658	0.38043	0.492537	0.429285
10326	0.43097	0.607595	0.504263
4365	0.41435	0.567568	0.479005
8693	0.42683	0.597403	0.497913
550	0.45136	0.66	0.536096
799	0.42533	0.59375	0.495623
3527	0.3617	0.454545	0.402842
4068	0.34723	0.426667	0.382871
6149	0.39329	0.52	0.447855
2449	0.32822	0.391892	0.357241
2468	0.34503	0.422535	0.37987
1539	0.45554	0.671233	0.542742
7153	0.3872	0.506849	0.439018
4436	0.42146	0.584416	0.489738
5381	0.36284	0.45679	0.404431
2284	0.43582	0.619718	0.51175

Table 5.6 Tabular representation of results (Precision, Recall and F-Score) of LexRank Algorithm corresponding the Text Input set

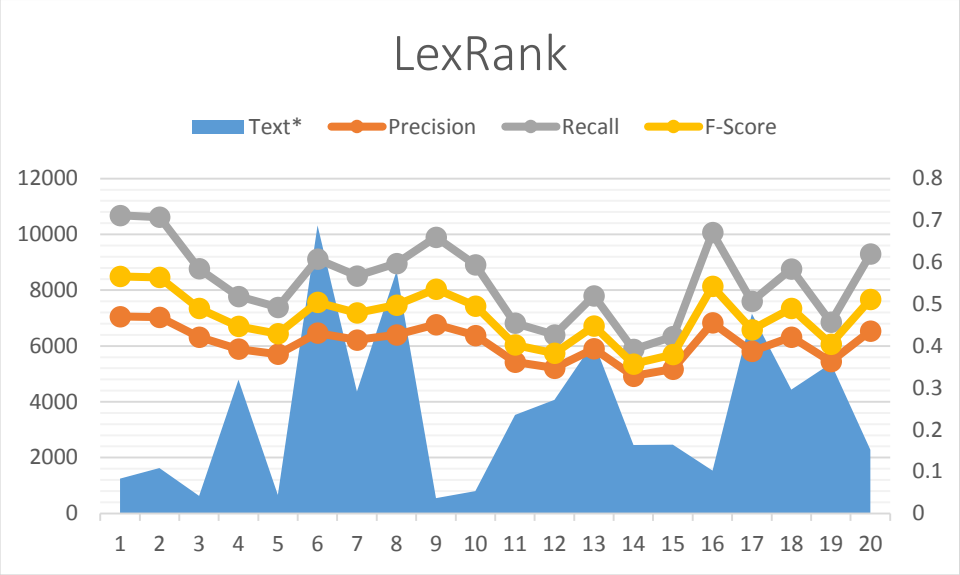


Fig. 5.6 Graph between the Precision, Recall and F-Score corresponding to given input set using LexRank Algorithm

5.2 Analysis

		Comparison of Recalls of different Algorithm					
Sno.	**Text**	**PM_Recall**	**KM_ Recall**	**R_Recall**	**FB_Recall**	**TR_Recall**	**LR_Recall**
Doc_1	1250	0.143585	0.125396	0.9090909	0.424242	0.772727	0.712121
Doc_2	1624	0.157753	0.133504	0.9076923	0.446154	0.876923	0.707692
Doc_3	629	0.142432	0.167469	0.9692307	0.661538	0.907692	0.584615
Doc_4	4787	0.156419	0.137767	0.9753086	0.407407	0.82716	0.518519
Doc_5	658	0.152653	0.146179	0.8656716	0.432836	0.701493	0.492537
Doc_6	10326	0.145995	0.208686	0.9729729	0.445946	0.918919	0.607595
Doc_7	4365	0.153994	0.152394	0.9324324	0.391892	0.864865	0.567568
Doc_8	8693	0.147278	0.227089	0.9577464	0.352113	0.901408	0.597403
Doc_9	550	0.157683	0.132575	0.92	0.56	0.76	0.66
Doc_10	799	0.137538	0.242981	0.90625	0.40625	0.84375	0.59375
Doc_11	3527	0.145072	0.142465	0.8939393	0.409091	0.69697	0.454545
Doc_12	4068	0.15212	0.136873	0.9733333	0.32	0.88	0.426667
Doc_13	6149	0.156358	0.159264	0.9866666	0.386667	0.933333	0.52
Doc_14	2449	0.154476	0.182004	0.9864864	0.391892	0.878378	0.391892
Doc_15	2468	0.139613	0.212941	0.9154929	0.492958	0.901408	0.422535
Doc_16	1539	0.150634	0.142053	0.9589041	0.273973	0.863014	0.671233
Doc_17	7153	0.145698	0.152654	0.9589041	0.39726	0.883714	0.506849
Doc_18	4436	0.160652	0.163004	0.9870129	0.519481	0.831169	0.584416
Doc_19	5381	0.152959	0.183003	0.9259259	0.308642	0.864198	0.45679
Doc_20	2284	0.14783	0.188859	0.9577464	0.633803	0.802817	0.619718

Table 5.7 Tabular comparison of recalls of all algorithms.

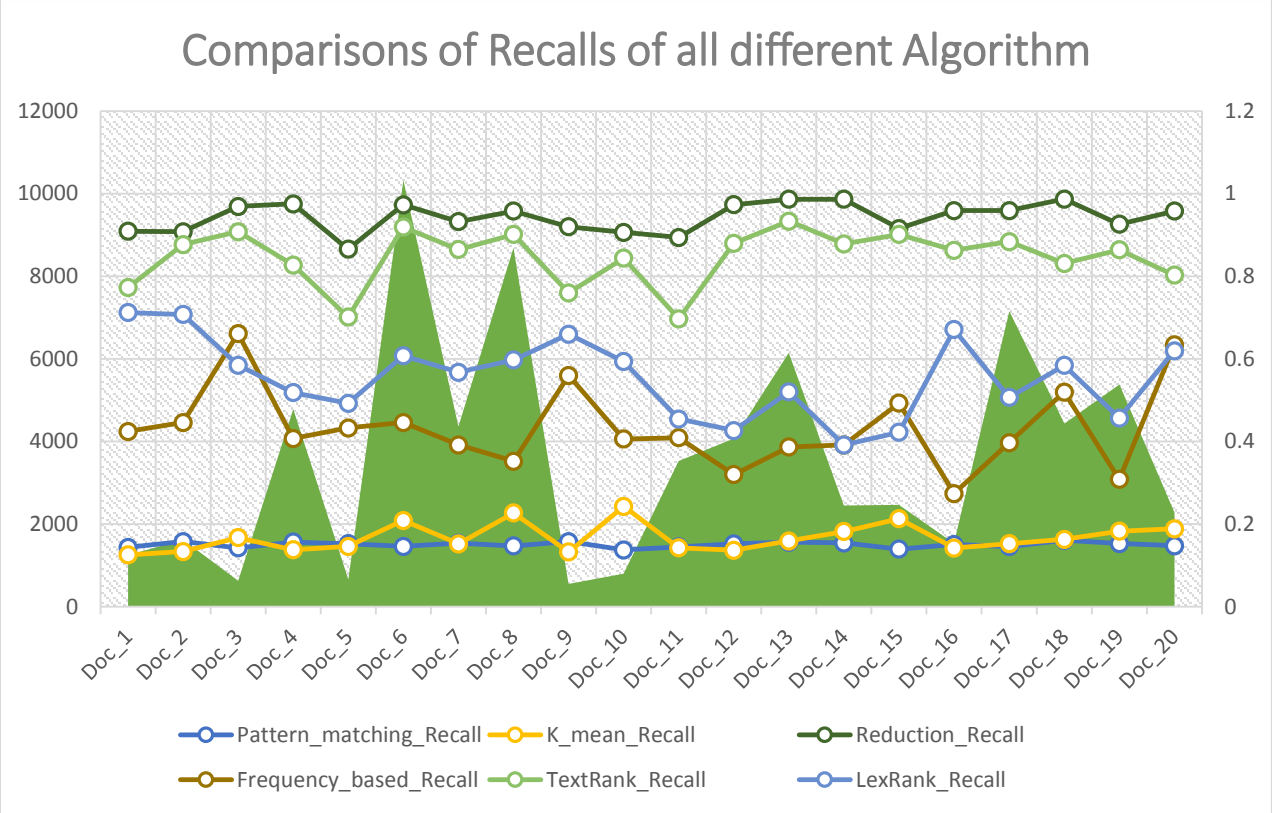


Fig. 5.7 Comparison Graph of 'Recall' of all different Algorithm for all Input sets

"As we can see in above plot that graph-based automatic summarization techniques have better 'recalls' as compare to tradition approaches for summarization, which means content generated by these techniques are more sensitive or opposes much exactness"

	Comparison of Precisions of all different Algorithms					
Input Text	**PM _Precision**	**KM _Precision**	**R_Precision**	**FB_Precision**	**TR_Precision**	**LR_Precision**
Doc_1	0.069934	0.373764	0.531164623	0.345936759	0.490602584	0.470228833
Doc_2	0.0825779	0.453623	0.530781455	0.357413956	0.52219019	0.468675817
Doc_3	0.0894476	0.574767	0.547068188	0.45193218	0.530781455	0.421547961
Doc_4	0.084596	0.422344	0.548615533	0.336835912	0.507608602	0.392611111
Doc_5	0.065629	0.532623	0.518969432	0.350486546	0.466486472	0.380427228
Doc_6	0.037328	0.78679	0.548022152	0.357306951	0.533839576	0.430972312
Doc_7	0.0628491	0.855167	0.537468129	0.328220998	0.518736818	0.414351975
Doc_8	0.036744	0.445113	0.544115245	0.305074943	0.529052132	0.426830474
Doc_9	0.072336	0.43139	0.534131959	0.411099868	0.48645529	0.45135584
Doc_10	0.056808	0.479743	0.530385656	0.336200906	0.512567729	0.425331436
Doc_11	0.081215	0.288848	0.526979974	0.337757393	0.464877833	0.361703421
Doc_12	0.0515904	0.597641	0.548113803	0.285191821	0.523063472	0.347226733
Doc_13	0.050518	0.441275	0.551478942	0.325269058	0.537708025	0.393291412
Doc_14	0.0680486	0.385491	0.551433798	0.328220998	0.522603632	0.328220998
Doc_15	0.0506607	0.565993	0.532910592	0.380628282	0.529052132	0.345025259
Doc_16	0.066911	0.686792	0.544414645	0.254622298	0.518202225	0.455536184
Doc_17	0.0079404	0.422344	0.544414645	0.331226972	0.521520795	0.387198708
Doc_18	0.085732	0.6189066	0.551565679	0.393053043	0.508816044	0.421464694
Doc_19	0.0546	0.5384066	0.535728139	0.277883908	0.518544232	0.362841109
Doc_20	0.05649	0.3773	0.544115208	0.441352228	0.500145832	0.435821496

Table 5.8 Tabular comparison of Precisions of all algorithms

Comparison of Precisions of all different Algorithms

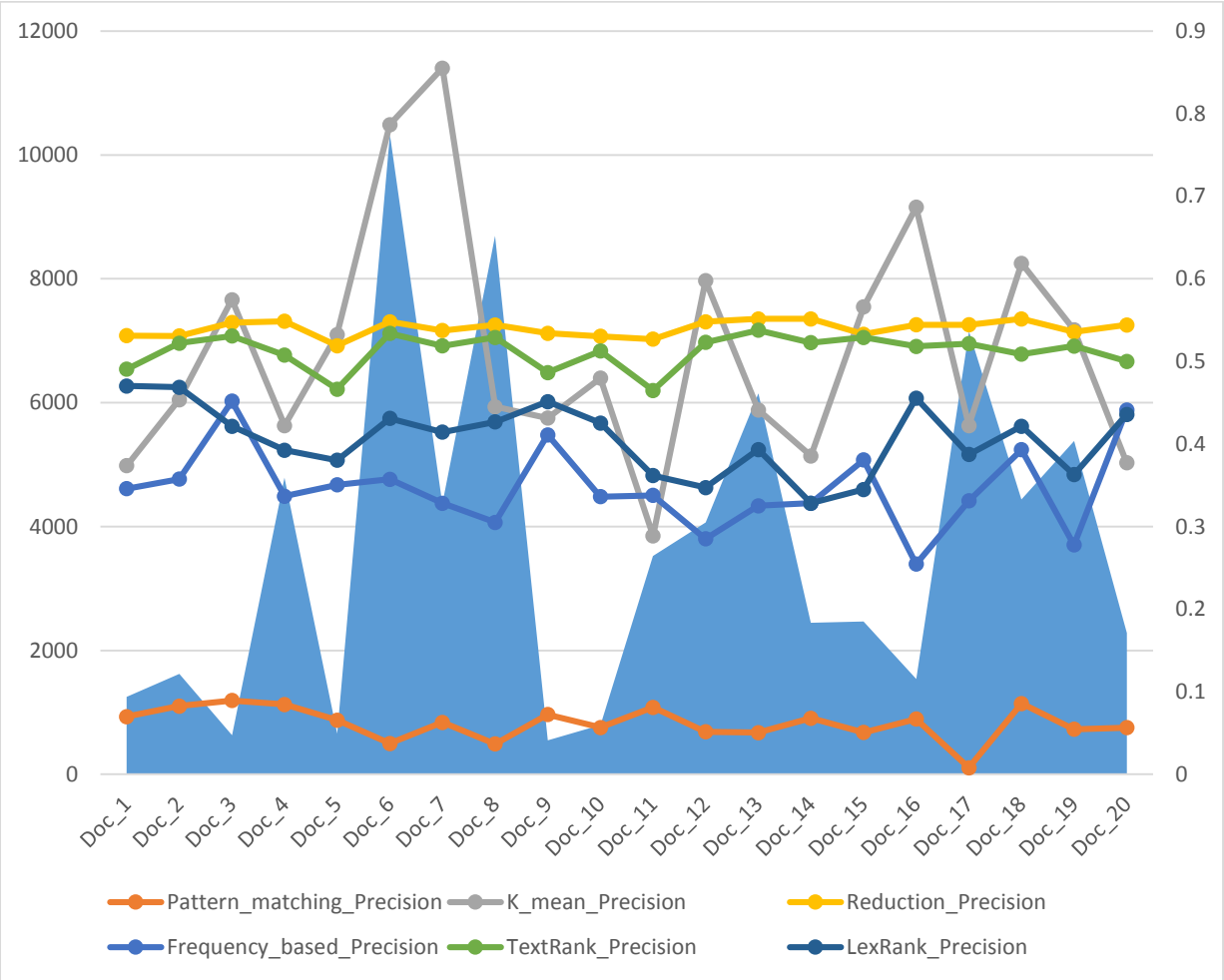


Fig. 5.8 Comparison Graph of 'Precisions' of all different Algorithm for all Input sets

"As we can see in above plot that graph-based automatic summarization techniques have better 'precisions' as compare to tradition approaches for summarization, which means content generated by these techniques are more relevant"

	Comparison of F-Scores of all different Algorithms					
Input Text	**PM_F-Score**	**KM_F-Score**	**R_F-Score**	**FB_F-Score**	**TR_F-Score**	**LR_F-Score**
Doc_1	0.09405695	0.18778953	0.632268528	0.381110801	0.600163413	0.566431957
Doc_2	0.10840802	0.20629433	0.600298168	0.396881566	0.654586786	0.563905198
Doc_3	0.10988637	0.25936671	0.67150849	0.537005	0.66985717	0.48986897
Doc_4	0.10980579	0.20776263	0.659100948	0.368778201	0.629133487	0.446864626
Doc_5	0.09179382	0.22939914	0.648914562	0.387334669	0.560346668	0.429285186
Doc_6	0.05945463	0.32987648	0.641344118	0.396737651	0.675343535	0.504263478
Doc_7	0.08926628	0.25868869	0.653194608	0.357240991	0.648506488	0.479004791
Doc_8	0.05881452	0.30074372	0.633402576	0.326907593	0.666767031	0.497912943
Doc_9	0.09917571	0.20281943	0.660128976	0.474134487	0.593211628	0.536095595
Doc_10	0.08040565	0.32258077	0.634871975	0.367920399	0.637725445	0.49562289
Doc_11	0.10413342	0.19081609	0.663075141	0.370019082	0.557742399	0.402842281
Doc_12	0.07704985	0.22273481	0.63806402	0.30159388	0.656129397	0.382871416
Doc_13	0.07636356	0.23405381	0.648404942	0.353320914	0.682320452	0.447855117
Doc_14	0.09447827	0.24726528	0.639304119	0.357240991	0.655317363	0.357240991
Doc_15	0.07434442	0.30945655	0.644591931	0.429572187	0.666767031	0.379869541
Doc_16	0.09266198	0.23541401	0.63401016	0.263942036	0.6475678	0.542741873
Doc_17	0.01506004	0.22425296	0.63401016	0.361252618	0.655940609	0.439018405
Doc_18	0.11180111	0.25804549	0.627252017	0.447506819	0.631219077	0.489738057
Doc_19	0.0804741	0.27315917	0.616490514	0.29245425	0.648168047	0.404430535
Doc_20	0.08174351	0.25171904	0.650224842	0.520351834	0.616327082	0.511749559

Table 5.9 Tabular comparison of F-Scores of all algorithms

Comparison of F-Scores of all different Algorithms

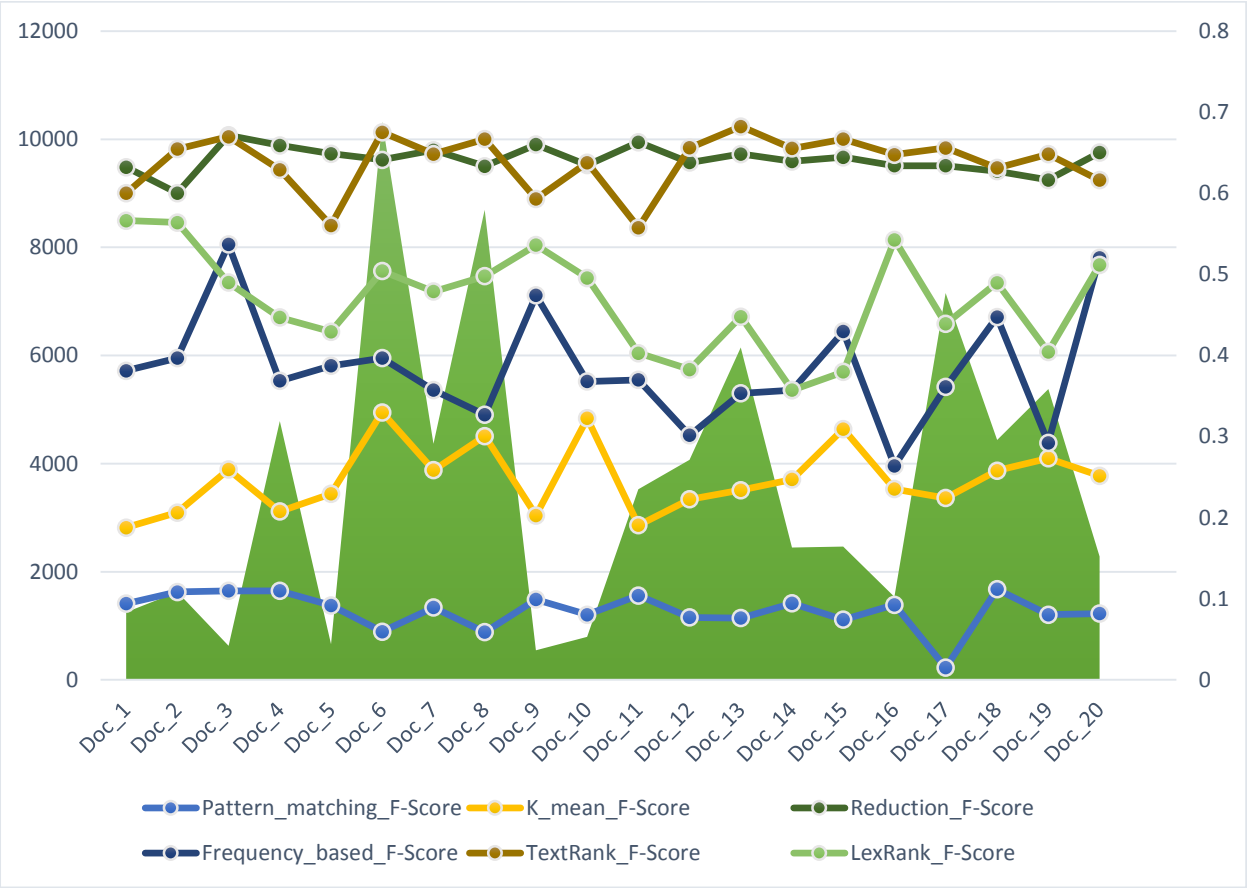


Fig. 5.9 Comparison Graph of 'F-Scores' of all different Algorithm for all Input sets

"As we can see in above plot that graph-based automatic summarization techniques have better 'F-Scores' as compare to tradition approaches for summarization, all these have evenly weighted 'recall' & 'precision' "

CHAPTER 6: CONCLUSION

6.1 Conclusion

In this this, the general overview of the automatic text summarization is described and analysed based on some parameters e.g. precision, recall and F-measure. The status, and state of automatic text summarization has intrinsically changed through recent years. Most of these approaches and the linguistic resources developed to aid them were built for the languages such as English. Therefore, it has benefited many other areas also; e.g. information retrieval, information extraction, text classification, news summarization and searching.

The summarizers producing human-quality texts are difficult to design, and even so more difficult to evaluate. The results of different research projects (*generated through various algorithms*) are also difficult to compare because the reported results often do not discuss the characteristics of the corpora. Hence, all the results (*summaries*) generated were collected in this work and analyzed how closed they were to those generated by human subjects, and finally evaluated for statistical significance

TextRank used the structure of the next and the known parts of speech for words to assign a score to words that are keyboard for the text. The algorithm gives more value to nodes with lots of connections, and gives more influence in steps to better connected nodes, so it enforces itself and eventually finds its stable score. Here we have used Levenshtein distance as function between words instead of co-occurrence relation used in the existing work.

By Constructing the similarity graph of sentences which gives a better view of important sentences compared to the centroid based approach, which is open to over-speculation of the data in the bunch of the documents. It quite promising to implement all these graph based summarization algorithms. In LexRank, tried to make use of more of the information in the graph, and even got better results in the most of the cases.

All graphs based algorithms are quite insensitive to noisy data that often occurs as a result of imperfect thematic document in cluster based algorithms. In graph-based approaches natural language dialect furnishes us with numerous new ways of information processing with applications to several problems such archive clustering, word sense disambiguation, prepositional expression attachment.

In traditional supervised and semi-supervised learning, one could not make powerful utilization of the features likelihood with each labelled or unlabelled data. In these approaches features serves as intermediate nodes on a path from unclassified to classified.

The mathematical model based approach for extraction of key sentences has yielded better results compared to those by simple term weighting methods.

6.2 Future Scope

In the future, we would like to integrate the best components of each and every system with extraction-based automatic text summarization (ATS) systems other than the one we have developed, improve the performance of the system further by introducing other sources of knowledge necessary for automatic summarization, and explore other interesting applications of the all these related algorithms. There are lot of scope for research due to fact that the automatic text summarizations (ATS) task has not been finished yet and there is still a lot to be investigated and to be improved. And improving the quality of the summarises generated by all these techniques. Here in the report analysis based on mostly extractive text summarization techniques, and will try to develop a technique for abstractive text summarization and explored all other neighbour techniques on summarisation.

REFERENCESES

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

[2]. Edmundson, H. P. (1969). New methods in automatic extracting. Journal of the ACM, 16(2):264–285

[3]. Salton, G.; Buckley, C. Term-weighting approaches in automatic text retrieval. Information Processing and Management 24, 513-523. 1988. Reprinted in: Sparck-Jones, K.; Willet, P. (eds.) Readings in I.Retrieval. Morgan Kaufmann (1997) 323-328.

[4]. Jing, H. and K. R. McKeown (2000). Cut and Paste-Based Text Summarization. In Proceedings of the 6th Applied Natural Language Processing Conference and the 1st Meeting of the North American Chapter of the Association for Computational Linguistics, pp. 178–185, Seattle, WA.

[5]. Mani, I. and M. T. Maybury (editors) (1999). Advances in Automatic Text Summarization. MIT Press, Cambridge, MA.

[6]. Baldwin, B., R. Donaway, E. Hovy, E. Liddy, I. Mani, D. Marcu, K. McKeown, V. Mittal, M. oens, D. Radev, K. Sparck-Jones, B. Sundheim, S. Teufel, R. Weischedel, and M. White (2000). An Evaluation Road Map for Summarization Research

[7]. James Allan, Jaime Carbonell, George Doddington,, Jonathan Yamron, and Yiming Yang. (1998). Topic detection and tracking pilot study: Final report. In Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop

[8]. G. Erkan and D. Radev, (2004) “LexRank: Graph-based Lexical Centrality as Salience in Text Summarization,” Journal of Artificial Intelligence Research, vol. 22, pp. 457-479

[9]. Xiaojun Wan, Jianwu Yang (2006). Improved affinity graph based multi-document summarization. In HLT-NAACL, pages 181-184

[10]. Julian Kupiec , Jan Pedersen , Francine Chen,(1995), A trainable document summarizer, Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval, p.68-73, July 09-13,

[11]. Conroy, John M, O'leary, Dianne P (2001), "Text summarization via hidden markov models", Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, page:406-407.

[12]. Osborne. (2002.) Using maximum entropy for sentence extraction. the ACL Workshop on Automatic Summarization.

[13]. I. Mani. J. Benjamin's, (2001), Automatic Summarization Publ. Co. Amsterdam Philadelphia

[14]. Zajic, B. Dorr, J. Lin, and R. Schwartz. 2006. Sentence compression as a component of a multi document summarization system. the ACL DUC Workshop.

[15]. Lin, C.-Y. And E. Hovy (2000). The automated acquisition of topic signatures for text summarization. In Proceedings of the 18th COLING Conference, Germany.

[16]. Karen Sparck Jones & Julia R. Galliers,(1995), Evaluating Natural Language Processing Systems: An Analysis and Review. Journal of Informatics, 2, 95-105, 145

[17]. Ronald Brandow, Karl Mitze, and Lisa F. Rau. (1995), "Automatic condensation of electronic publications bysentence selection. Information Processing and Management", 31(5):675-685,

[18]. G. Erkan and D. Radev, (2004) "LexRank: Graph-based Lexical Centrality as Salience in Text Summarization," Journal of Artificial Intelligence Research, vol. 22, pp. 457-479, .

[19]. Rada Mihalcea and Paul Tarau, (2004.), "Text-rank: Bringing Order into Texts," Proceeding of the Conference on Empirical Methods in Natural Language Processing, Barcelona, Spain,

[20]. Xiaojun Wan, (2007), "TimedTextRank: Adding the Temporal Dimension to Multi-Document Summarization," Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, pp. 867-868

[21]. Jing, H. (2000). Sentence Reduction for Automatic Text Summarization. In Proceedings of the 6th Applied Natural Language Processing Conference, pp. 310–315, Seattle, WA

[22]. Mani, I.; House, D.; Klein, G.; Hirschman, L.; Obrsl, L.; Firmin, T.; Chrzanowski, M.; Sundheim, B. (1998). The TIPSTER SUMMAC Text Summarization Evaluation. MITRE Technical Report MTR 98W0000138. The MITRE Corporation

[23]. Martin Hassel (2004) "Evaluation of automatic text summarization", Stockholm, KTH, University of Numerical analysis and computer science, Sweden.

[24]. G. Salton, (1989), "Automatic Text Processing: the transformation, analysis, and retrieval of information by computer," Addison Wesley Publishing Company, USA,

[25]. Shanmugasundaram Hariharan, Thirunavukarasu Ramkumar and Rengaramanujam Srinivasan, (2012), "Enhanced Graph Based Approach for Multi Document Summarization," The International Arab Journal of Information Technology,

Appendix A: Code Snippets

```
class Reduction:
        functionPunctuation = ' ,-'
        contentPunctuation = '.?!\n'
        punctuationCharacters = functionPunctuation+contentPunctuation
        sentenceEndCharacters = '.?!'

        def isContentPunctuation(self, text):
                for c in self.contentPunctuation:
                        if text.lower() == c.lower():
                                return True
                return False

        def isFunctionPunctuation(self, text):
                for c in self.functionPunctuation:
                        if text.lower() == c.lower():
                                return True
                return False

        def isFunction(self, text, stopWords):
                for w in stopWords:
                        if text.lower() == w.lower():
                                return True
                return False

        def tag(self, sampleWords, stopWords):
                taggedWords = []
                for w in sampleWords:
                        tw = Word()
                        tw.Text = w
                        if self.isContentPunctuation(w):
                                tw.Type = WordType.ContentPunctuation
                        elif self.isFunctionPunctuation(w):
                                tw.Type = WordType.FunctionPunctuation
                        elif self.isFunction(w, stopWords):
                                tw.Type = WordType.Function
                        else:
                                tw.Type = WordType.Content
                        taggedWords.append(tw)
                return taggedWords
```

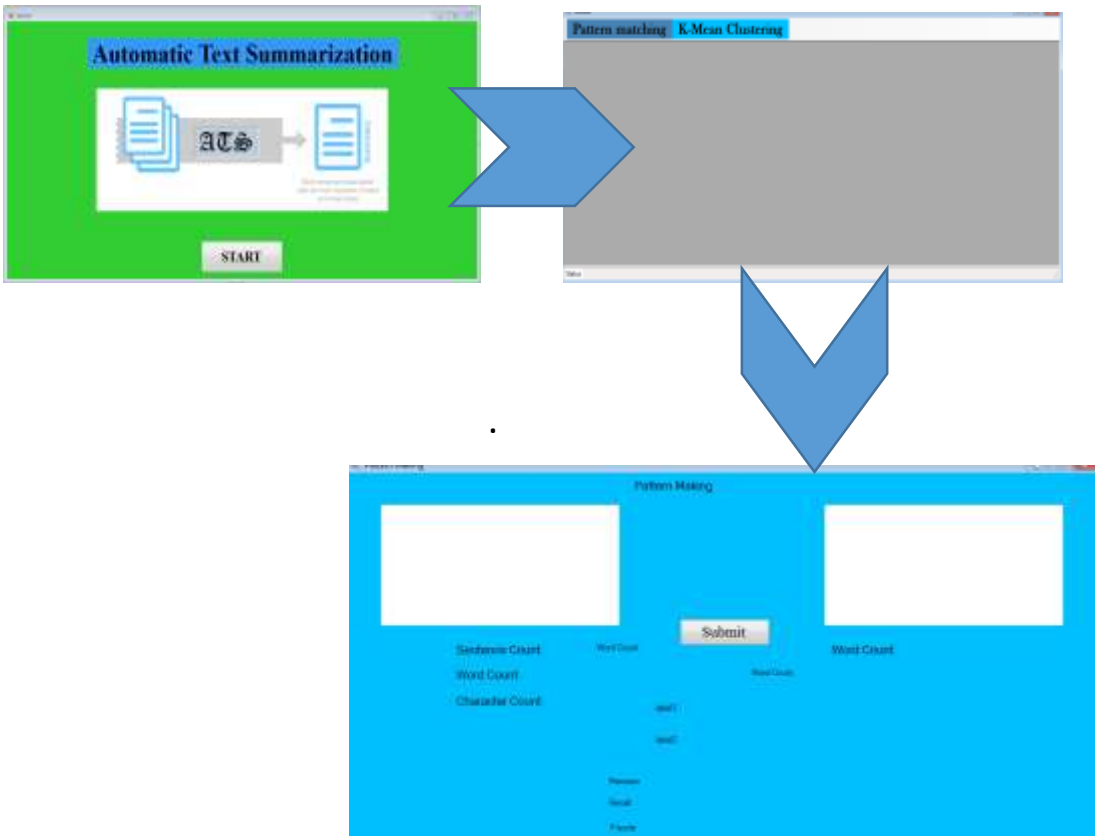
Reduction of the sentences

```
def unique_everseen(iterable, key=None):
    "List unique elements, preserving order. Remember all elements ever seen."
    # unique_everseen('AAAABBBCCDAABBB') --> A B C D
    # unique_everseen('ABBCcAD', str.lower) --> A B C D
    seen = set()
    seen_add = seen.add
    if key is None:
        for element in itertools.ifilterfalse(seen.__contains__, iterable):
            seen_add(element)
            yield element
    else:
        for element in iterable:
            k = key(element)
            if k not in seen:
                seen_add(k)
                yield element

def lDistance(firstString, secondString):
    "Function to find the Levenshtein distance between two words/sentences - got
    if len(firstString) > len(secondString):
        firstString, secondString = secondString, firstString
    distances = range(len(firstString) + 1)
    for index2, char2 in enumerate(secondString):
        newDistances = [index2 + 1]
        for index1, char1 in enumerate(firstString):
            if char1 == char2:
                newDistances.append(distances[index1])
            else:
                newDistances.append(1 + min((distances[index1], distances[index1
```

TextRanking Algorithm (function to calculate L-Distance between two phases)

Appendix A: Snapshots of the system




```
Type "copyright", "credits" or "license()" for more information.
>>> ========== RESTART ==========
>>>

++++++++++++++++++++++++++#####++++++++++++++++++++++++++++++++++++++++

.....................Frequency based Algorithm.....................
++++++++++++++++++++++++++#####++++++++++++++++++++++++++++++++++++++++

Reading articles/Doc10_Finan_washingtonpost_889.htm.txt
Actual Text:

[3527]

Summarized Text

[34]

Precision=
0.451609090806
Recall=
0.409090909091
F-Score=
0.58064516129

++++++++++++++++++++++++++#####++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++#####++++++++++++++++++++++++++++++++++++++++

Reading articles/Doc11_Finan_washingtonpost_1010.htm.txt
```

Snapshot of Frequency based summarization system (Running , *command line*)

_*******_