VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANASANGAMA, BELAGAVI - 590018

“New Graph Based Text Summarization Method”

Thesis submitted in partial fulfillment of the curriculum prescribed for the award of the degree of Bachelor of Engineering in Computer Science & Engineering by

1CR14CS108	R Srinath
1CR14CS085	Namratha Krishnaprasad
1CR14CS150	Tavleen K Allagh

Under the Guidance of

Dr.Jhansi Rani P
Professor & HOD
Department of CSE, CMRIT, Bengaluru

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
#132, AECS LAYOUT, IT PARK ROAD, BENGALURU - 560037

2017-18

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANASANGAMA, BELAGAVI - 590018

Certificate

This is to certify that the project entitled "New Graph Based Text Summarization Method" is a bonafide work carried out by R Srinath, Namratha Krishnaprasad and Tavleen K Allagh in partial fulfillment of the award of the degree of Bachelor of Engineering in Computer Science & Engineering of Visvesvaraya Technological University, Belgaum, during the year 2017-18. It is certified that all corrections / suggestions indicated during reviews have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the Bachelor of Engineering Degree.

-------------------------	-------------------------	-------------------------
Signature of Guide	Signature of HoD	Signature of Principal
Dr. Jhansi Rani P	**Dr. Jhansi Rani P**	**Dr. Sanjay Jain**
Professor & Head	Professor & Head	Principal
Department of CSE	Department of CSE	CMRIT,
CMRIT, Bengaluru - 37	CMRIT, Bengaluru - 37	Bengaluru - 37

<u>External Viva</u>

	Name of the Examiners	Institution	Signature with Date
1.	-------------------------	----------------------------	----------------------------
2.	-------------------------	----------------------------	----------------------------

Acknowledgement

We take this opportunity to thank all of those who have generously helped us to give a proper shape to our work and complete our BE project successfully. A successful project is fruitful culmination efforts by many people, some directly involved and some others indirectly, by providing support and encouragement.

We would like to thank **Dr. SANJAY JAIN** , Principal , CMRIT , for providing excellent academic environment in the college.

We would like to express our gratitude towards our Internal Guide **Dr. JHANSI RANI** , Professor & HOD , Dept of CSE , CMRIT , who provided guidance and gave valuable suggestions regarding the project.

We consider it a privilege and honour to express our sincere gratitude to our mentors **Dr.R.N.V.SITARAM** and **Mr.HARISH KASIVISWANATHAN** , for their valuable guidance throughout the tenure of this project work.

Srinath
Namratha K
Tavleen K Allagh

Table of Contents

List of Figures

List of Tables

Abstract

The exponential growth of text data on the WorldWide Web as well as on databases off line created a critical needfor efficient text summarizers that significantly reduce its size while maintaining its integrity. In this paper, we present a new multigraph-based text summarizer method. This method is unique in that it produces a multi-edge-irregular-graph that represents words occurrence in the sentences of the target text. This graph is then converted into a symmetric matrix from which we can produce the ranking of sentences and hence obtain the summarized text using a threshold. To test our method performance, we compared our results with those from the most popular publicly available text summarization software using a corpus of 1000 samples from 6 different applications: health, literature, politics, religion, science and sports. The simulation results show that the proposed method produced better or comparable summaries in all cases. The proposed method is fast and can be implement for real time summarization.

Chapter 1

PREAMBLE

1.1 Introduction

Text database contains high volume of text data. Document retrieval retrieves number of documents still beyond the capacity of human analysis, e.g. at the time of writing the query Summarization in Google returned more than 27,60,000 results (as on 20th Mar 2018 from Google). Thus document retrieval is not sufficient and we need a second level of abstraction to reduce this huge amount of data: the ability of summarization. This work tries to address this issue and proposes an automatic text summarization (TS) technique. Roughly summarization is the process of reducing a large volume of information to a summary or abstract preserving only the most essential items.

Radev et al. (2002) [3] defined a summary as 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. This simple definition captures three important aspects that characterize research on automatic summarization:

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

Automatic text summarization is a multi facetted endeavour that typically branches out in several dimensions, which can be grouped into several over lapping categories.Based on the methodology or technique used summarization approaches can be divided into two broad groupings-as extraction and abstraction schemes. An extractive summarization method consists of selecting important sentences, paragraphs etc. from the original document and concatenating them into shorter form. The importance of sentences is decided based on statistical and linguistic features of sentences. An abstractive summarization method consists of understanding the original text and re-telling it in fewer words. It uses linguistic methods to examine and interpret the text and then to find the new concepts and expressions to best describe it by generating a new shorter text that conveys the most important information from the original text document. Speed, simplicity, non requirement of background knowledge, and domain independency are some of the features that favour extraction, where as abstraction is domain dependent in nature and requires human knowledge and is goal oriented.

Based on volume of text documents available in the text database, it can be classified as single or multi document text summarization. If summarization is performed for a single text document, then it is called as the single document text summarization. If the summary is to be created for multiple text documents, then it is called as the multi document text summarization technique.

In the business world, companies produce massive number of large reports, which are generated on annually, monthly, weekly, or even on a daily basis. Company owners and employees want these reports effectively summarized for quick reference. There are situations when we are surrounded by summarized information and we often take it for granted. For example, we cannot think of a newspaper without headlines or books and movies without reviews and trailers. Similarly, articles, such as this one, without abstracts and summarized results would be unacceptable. Some of the other applications of the text summarization are as follows:

- Text Summarization can be used to save time.
- Text Summarization can speed up other information retrieval and text mining processes.
- Text Summarization can also be useful for text display on hand-held devices, such as PDA. For instance, a summarized version of an email can be sent to a hand-held device instead of a full email.

1.2 Problem statement

To implement a new graph-based text summarizer that can efficiently reduce the size of a text while maintaining the integrity. The proposed method is simple to implement and does not rely on the calculations of the cosine similarity between sentences to rank the them in the summary.

1.3 Related Work

Computer based Text Summarization Techniques are scarce and can be divided into six categories.

The first category is graph-based approach [4][5][6][1]. This approach is used not only to single document but also to multi-document summarization. In this category, each sentence is treated as a node. Two nodes or specifically two sentences are connected with an edge, if the two sentences have some similarity. Calculation of this similarity depends on many features. Like two nodes can be connected, if the sentences have the same words. Cosine similarity between sentences is a widely used measure. The nodes or sentences which have more edges connected to other nodes are the important ones.

The second category is the machine-learning approach [7] [8] [9] [1] . This approach was introduced in 1990s. The main methods, which use this approach, are naive-bayes methods, rich features and decision trees, hidden markov models, log linear models and neural networks. Researchers used naive-bayes methods to calculate the naive-Bayes classifier.This classifier categorizes each sentence as worthy of extraction or not. 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:

$$P(s \in S|F_1, F_2..F_k) = \frac{\prod_{i=1}^{k} P(F_i|s \in S).P(s \in S)}{\prod_{i=1}^{k} P(F_i)} \tag{1.1}$$

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 third category is clustering based approach [9] [1]. This approach is suitable for both single document and multi-document summarization. As a first step, it creates clusters of documents, which are to be summarized and then find the relationships that exist among them. Similar documents and passages are clustered together so that the related information remains in the clusters. Then each cluster is indexed, which depends on the theme of the cluster. After clustering, sentences are ranked within each cluster then their saliency scores are calculated. In the last step high score sentences from each cluster are extracted to form the summary.
Multi-document text summarization means to retrieve salient information about a topic from various sources. Given a set of documents D = (D1, D2,,Dn) on a topic

T, the task of multi-document summarization is to identify a set of model units (S1,S2,,Sn). The model units can be sentences, phrases or some generated semantically correct language units carrying some useful information. Then significant sentences are extracted from each model units and re-organized them to get multi-documents summary. Process flow of multi-document summarization can be depicted in the figure shown.

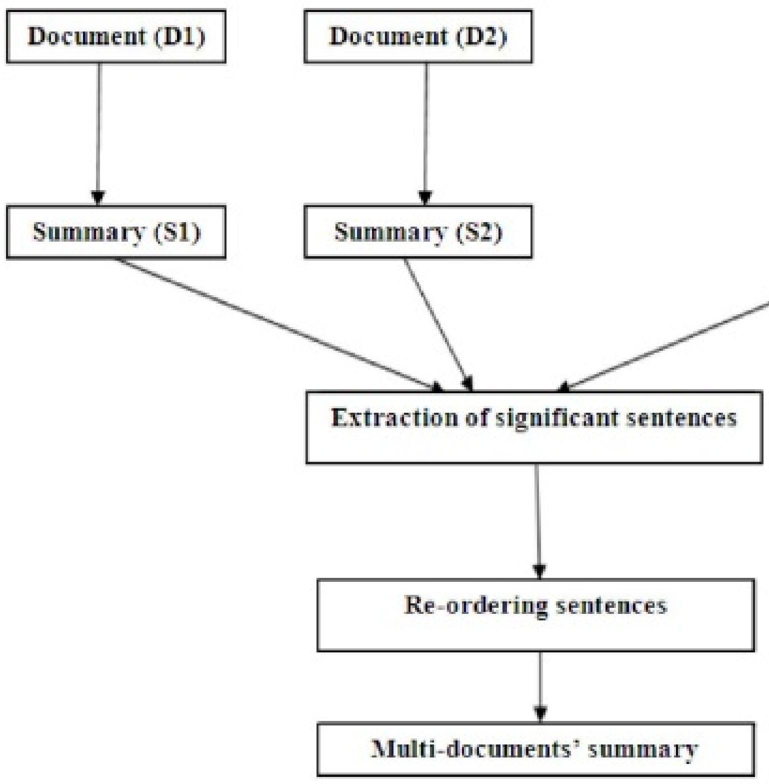


Figure 1.1: Process flow of multiple-document text summarization

The fourth category is Lexical Chaining approach [10][11][1] . A lexical chain is a sequence of related words in writing, spanning short (adjacent words or sentences) or long distances (entire text). A chain is independent of the grammatical structure of the text and in effect it is a list of words that captures a portion of the cohesive structure of the text. A lexical chain can provide a context for the resolution of an ambiguous term and enable identification of the concept that the term represents.
Examples of lexical chains are the following:
Bangalore capital city inhabitant
It is a chain of words in which a word gets inclusion in a chain if it is cohesively and coherently related to the other words already present in the chain. Each chain of

words represents the semantically related cluster of words. Words from the document are grouped together into meaningful clusters to identify various themes within a document. Then these clusters are arranged systematically to form a binary tree structure. Lexical chains start building up when the first word of the document starts. Then it checks for the second word that whether it is semantically related to the first one or not. If the words are related then the second word is also added in the first chain else second chain starts and so on.
The main steps of the construction of Lexical Chains are as follows: the construction of Lexical Chains begins with the first word of a text. To insert the next candidate word, its relations with members of existing Lexical Chains are checked. If there are such relations with all the elements of a chain then the new candidate word is inserted in the chain. Otherwise a new lexical chain can be started. This procedure includes disambiguation of words, although we allow a given word to have different meanings in the same text (i.e. belonging to different Lexical Chains).

The fifth category is Frequent Term Approach [1]. This method checks for the terms, which are frequent and semantically similar. There are some methods like term frequency inverse document frequency, tf-idf, for the calculation of frequent terms in the document. The score of a term in the document is the ratio of the quantity of terms in that document to the frequency of the quantity of documents containing that terms. The significance of term evaluation is given by the principle TFI X IDFI, where TFI is the frequency of term I in the document and IDFI is the inverted frequency of documents in which that term occurs. By computing pertinence of terms in the sentence, consequently sentences can be scored for illustration. For the calculation of semantic similarity, it checks the length of the path linking the terms, position of the terms, measures the difference of information content and the similarity between the terms. After this, summarizer filters the sentences having most frequent, semantically related terms and extracts them for summary.

The sixth category is Information Retrieval Approach [1] . This approach is an enhancement of the two graphical methods LexRank (threshold) and LexRank (continuous), proposed in 2012. In this method the main feature is logical closeness i.e., how two sentences are logically related to each other rather than just the topical closeness. In addition, sentences must be coherent in sense. Finally more related sentences are picked up in a chain to produce the logical summary. This technique is very similar to graph based approach and lexical chaining. In fact its a hybrid of two categories

1.4 Organisation of Report

Chapter 1: Introduction - Gives the basic idea of the project.

Chapter 2:Literature Review - Gives a brief overview of the survey papers and the research sources that have been studied to establish a thorough understanding of the project under consideration.

Chapter 3: About the system - Gives a brief review of the existing and proposed system along with feasibility study.

Chapter 4: Determine the vocabulary of the terms - Give a brief review of all the pre-processing steps of document summarisation.

Chapter 5: System Requirement Specification - Discusses in details about the different kinds of requirements needed to successfully complete the project.

Chapter 6: System Analysis - gives details about several analyses that are performed to facilitate taking decision of whether the project is feasible enough or not.

Chapter 7: System Design - Gives the design description of the project, conceptual and detailed design well supported with design diagrams.

Chapter 8: Implementation - Discusses the implementation details of the project .

Chapter 9: Results and Performance Analysis - Gives the snapshots and graphs of the proposed protocols.

Chapter 10: Conclusion and Future Scope - Gives the concluding remarks of the project, throwing light on its future aspects.

References Lists the websites and references referred during the project work.

Chapter 2

LITERATURE SURVEY

There is plenty of information available on the internet. Important information can be considered by creating summary from available information. Manual creation of summary is a complicated task. Therefore, research community is developing new approaches for automatic text summarization. Automatic text summarization system creates summary. Summary is a shorter text that covers important information from original document. Summary can be created using extractive and abstractive methods. Abstractive methods are requires deep understanding of text. After understanding, it represents text into new simple notions in shorter form. Extractive approach uses linguistic and statistical approach for selection of sentences for summary.

In todays fast emerging world of information, text summarization is very vital and required tool for understanding text information. There is a lot of text material and documents available on the internet which provides information beyond requirements and creates the situation called infobesity. To select information from large amount of information from variety of sources is difficult for human beings. Due to the volume of information and unstrucutredness of information, to manually summarize information available on the internet is really challenging, complicated and difficult task. The aim of automatic text summarization is to reduce the source text into a compact version which will preserve contents and general meaning. Advantage of Summary is that it minimizes reading time and efforts.

2.1 Types of summarization

2.1.1 Based on processing technique

These types are based on the text processing techniques, whether text for summary is just selected based on statistical/linguistic features of sentences or the deep understanding of text is required. [2]

2.1.1.1 Abstractive

Abstractive summarization try to understand the main concepts by using in a document and represent them in basic natural language. For understanding and examining text, it uses linguistic methods. After understanding of text, first it finds the new notions and terms to best clarify it. By using these notions and terms, creating a new shorter text that can represent the most significant information from the original text document.

2.1.1.2 Extractive

Extractive summarization selects significant paragraphs, lines and words from original text and clubs them as summary. The selection of paragraph, line or word is based on statistical and linguistic features of sentences. Extractive summaries are created by extracting main text fragment (sentences or passages) based on statistical analysis of features (as word/phrase frequency, location or cue words) to locate the sentences to be extracted from the text. The content which is used most commonly or the content which is having most favourable location is considered as most important content. This approach does not require deep understanding of text. Extractive text summarization process is divided into two steps i.e Pre Processing and Processing. Pre Processing is controlled representation of the original text. It includes three activities: i) identification of Sentences boundary: Sentence boundary is identified with presence of dot at the end of sentence, in English. ii) Removal of Stop-Word: Common words with no meaning and which do not represent related information to the task are removed. iii) Stemming: the purpose of stemming is to find the stem or radix (source or origin) of each word, which give stress to its semantics. In processing step, features influencing the relevance of sentences are decided and calculated and then weights are assigned to these features using weight learning method. Total final score of each sentence is calculated using Feature-weight equation. Top ranked sentences are selected for final summary.

2.1.2 Based on purpose of processing

These types are based on the length of summary.

- **Indicative summaries**: Indicative summaries give shortened information on the main topics of a document. It gives a clear idea to reader that original document is worth reading. The typical lengths of indicative summaries range between 5-10% of the complete text.

- **Informative summaries**: Informative summaries provide a replacement for full document, which retain significant details and reduce volume of information. Informative summary is typically 20-30% of the original text.
- **Critical or Evaluative summaries**: Critical or Evaluative summaries capture the point of view of the author on a given subject. Reviews are typical example of that.
- **Update summaries**: In Update summaries, it is considered that user have the fundamental knowledge about the topic and requires only the current updates regarding that particular topic.

2.1.3 Based on audience

These types are based on the audience who is going to use summary i.e. whether summary is for group of people, or it is for a specific users query.

- **Generic summaries**: Generic summary result is aimed at a wide group of people, equal important is given to all major topics.
- **Query-based summaries**: The result is based on a question or query.
- **User specific or Topic specific summaries**: This type of summary is customized for the concern of exacting user or highlight only particular topic.

2.1.4 Based on Number of Document(s) and language

These types are based on number of documents to summarize and languages of text document(s). **Based on documents:**

- Single document summarization where the summary is created from single document.
- Multi document summarization where the summary is created from multiple documents.

Based on language

- Single language summarization where the summary is created from single language document.
- Multi language summarization where the summary is created to multiple language documents.

2.2 Problems with the text summarization

The Problems with extractive text summarization are[2]:

1. Sentences selected for summary generally longer, so unnecessary parts of the sentences for summary also get included & they consume space.
2. If summary size is not long enough, the important information scattered in various statements cannot captured using extractive summarization.
3. Information which is clashing may not be presented accurately.
4. Sentences frequently contain pronouns. They lose their referents when used out of context. If irrelevant sentences are clubbed together, may lead to confusing understanding of anaphors which will result in erroneous representation of original information.
5. The same problem is with multidocument summarization, because extraction of text is performed on different sources. Post processing can be used to deal with these troubles, for example, replacing pronouns with their background, replacing relative temporal expression with actual dates etc.

Problems with the abstractive text summarrization is the problem representation. Capability of system is dependent on how carefully problem is represented. The system cannot able to summarize the things that are not represented properly in problem.

2.3 Features For Extractive Text Summarization

Some features to be considered for including a sentence in final summary are[2]:

1. Title word feature :
   Sentences containing words that are same as title, are also pinpointing of the matter of the document. Such sentences are having higher chances to get included into summary.
2. Content word (Keyword) feature:
   Content words or Keywords are generally nouns. They can be determined using term frequency - inverse document frequency. Sentences which contain keywords are of higher chances to get included into summary.
3. Sentence Length feature:
   Very large and very short sentences are not considered in summary.

4. Sentence position feature:
   Sentence position matters a lot in abstractive text summarization. Usually first and/or last sentence of first and/or last paragraph of a text document are additional important and are having higher chances to get included into summary.

5. Proper Noun feature:
   Proper noun can be name of an entity, name of place and name of any concept etc. Sentences containing proper nouns are having higher chances to get included into summary.

6. Upper-case word feature:
   Sentences containing acronyms or proper names are included in summary.

7. Cue-Phrase Feature:
   Sentences containing any cue phrase are most possible to be in summaries.

8. Biased Word Feature:
   If a word appearing in a sentence is from biased list of words, then that sentence is important. Biased word list is predefined and may contain domain specific words.

9. Font based feature:
   Sentences containing words written in upper case, bold, italics or Underlined fonts are considered more important.

10. Pronouns:
    Pronouns such as they, it, he , she cannot get included in summary unless they are expanded into matching nouns.

11. Presence of non-essential information:
    Some words are indicators of pointless information e.g. because, furthermore, and additionally, and typically occur in the beginning of a sentence. True or 1 value can be taken for this feature if the sentence contains at least one of these words, and false or 0 in opposite case.

12. Sentence-to-Sentence Cohesion:
    For each sentence of document, similarity between s1 and each other sentence s of the document is calculated. By summing up all those similarity values, raw value of this feature can be obtained for specific sentence. The process is repeated for all sentences.

13. Sentence-to-Centroid Cohesion:
    For each sentence, first compute the vector representing the centroid of the document. Centroid is the arithmetic average over the corresponding coordinate values of all the sentences of the document; then computing the similarity between the centroid and each sentence; raw value of this feature can be obtained for each sentence. The process is repeated for all sentences.

14. Discourse analysis:
    Discourse level information, in a text is one of good feature for text summarization. In order to produce a coherent, assured 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.

2.4 Extractive Summarization Approaches For Multiple Language Text Summarization

Multilingual text summarization is used to summarize the source text in different language to the target language final summary[2]. SimFinderML identifies similar parts of text by calculating similarity over multiple features. It uses 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 built 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.

MEAD is platform for multi-lingual summarization and evaluation. MEAD implements multiple summarization algorithms such as position-based, centroid-based, largest common subsequence, and keywords. The methods for evaluating the quality of the summaries are both intrinsic (such as percent agreement, cosine similarity, and relative utility) and extrinsic (document rank for information retrieval). MEADs architecture consists of four stages. First, documents in a cluster are converted to MEADs internal format which is based on XML. Second, given a configuration file or command-line options, a number of features are extracted for each sentence of the cluster. Third, these extracted features are combined into a combine score for each sentence. Fourth, these scores can be further refined after considering possible cross-sentence dependencies (e.g., repeated sentences, sequential 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.

The Naive Bayesian Classification with the timestamp concept for text summarization works on different domains like international news, politics, sports and entertainment. The length of summary and compression rate can be specified as per User's need. The timestamp provides the summary an ordered look, which attain the coherent looking summary. It is used to extracts the more relevant information from the multiple documents. The word frequency is calculated. The system is compared with the existing MEAD algorithm and gives better outputs than the MEAD algorithm. The system is better precision, recall and F-Score. The timestamp procedure is also applied on the MEAD algorithm and the results are examined with this method. The results show that the proposed method results in lesser time than the existing MEAD algorithm to execute the summarization process.

2.5 Extractive Summarization Approaches For Multiple Document Text Summarization

The Multi-document summarization framework is based on event information and word embeddings [2]. The framework was developed by extending a kp centrality - single document summarization method. It involves two different strategies. I. Single layer approach combine summaries of each input document to create final summary. Ii. The waterfall approach combines summaries in cascade fashion, according to temporal sequence of documents. Event information is used in filtering stage and to improve the sentence representations. They used skip-gram model, continuous bag-of-word model and distributed representation of sentences. Event detection uses fuzzy fingerprint method. Evaluation is performed using rouge-1 and user study. Graph sum, a graph-based summarizer works on collection of documents, that determines and uses association rules to represent the connections among multiple terms during the summarization process. Graph sum uses a strategy that distinguishes between positive and negative term correlations. The graph nodes, which represent combinations of two or more terms, are first ranked by means of a Page Rank. Then, the produced node ranking is used to perform the sentence selection process.

2.6 Extractive Summarization Approaches For Multiple Language and Multiple Document Text Summarization

MINDS integrate multi lingual summarization and multi document summarization capabilities using a multiengine core summarization system that provides interactive document access through hypertext summaries[2]. It produces summaries both in English and in the original language of a document. It uses core summarization engine independent of languages. A prototype core engine has been built for English, Spanish, Russian, and Japanese documents. It uses document structure analysis and word frequency analysis as core summarization techniques. Document structure analysis involves identification of language, document structure parsing (heading, subheading, section, subsection, data and graphics gets separated for document for HTML encoding) Multilingual Sentence Segmentation and text structure heuristics (it uses rules based on document to score sentences and it is main method for scoring and selecting sentences.) Word frequency analysis sorts words of document by frequency and selects some most frequent words.

2.7 Summary Evaluation

Summary evaluation is important for text summarization[2]. Approaches to evaluation are divided into extrinsic where a summary is judged according to how much it contributes to the accomplishment of a particular task, and intrinsic wherein the quality of a summary is judged directly without reference to a particular task. Evaluation of Performance of Automatic summary can be measured using precision, recall and F-score. Precision is the number of sentences found in both system and ideal summaries divided by the number of sentences in the system summary. Recall is the number of sentences found in both system and ideal summaries divided by the number of sentences in the ideal summary. F-score is a combined measure and it combines precision and recall. A set of metrics called Recall Oriented Understudy of Gisting Evaluation (ROUGE) was introduced, it has become the standard of automatic evaluation of the summaries, and gives a score based on the similarity in the sequences of words between a human-written model summary and the machine summary. Evaluating summaries, either manually or automatically, is a hard task. The main difficulty in evaluation comes from the impossibility of building a fair 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.

Chapter 3

ABOUT THE SYSTEM

3.1 Existing System

Automatic text summarization is very challenging, because when we as humans summarize a piece of text, we usually read it entirely to develop our understanding, and then write a summary highlighting its main points. Since computers lack human knowledge and language capability, it makes automatic text summarization a very difficult and non-trivial task.

The existing system is people reading the entire document and manually writing a summary. Our method is different from other method as it does not use the cosine equation to find the similarity between the sentences. For the calculation of cosine equation, researchers have been using tf-idf but we are not calculating these factors for each word within a sentence. The other difference is that we are not identifying any relation within a sentence because within a sentence we don't have to find edges. Those edges may contribute towards redundant information in the summary. We are not tracking each word for the calculation of matrix. We are focusing on the significant words only.

3.2 Proposed System

The proposed method is multi-graph based[1]. The number of edges in the graph between two nodes is equal to the number of identical words in both sentences. According to our assumption, a word may occur in a sentence more than once, such occurrence will be added in the symmetric matrix as shown in Tables below. The total number of edges between the nodes are stored in a symmetric matrix that represents the text being summarized. Then, we perform a summation on all the values of each row (or column symmetric matrix) of the matrix to generate what we call a sum vector, which is then used for ranking the sentences as shown in table 1. Then, we

	S1	S2	S3	S4	Sum	Sentence Rank
S1	0	5	4	4	13	1
S2	5	0	3	2	10	4
S3	4	3	0	5	12	2
S4	4	2	5	0	11	3

Table 3.1: Matrix Generated by our method

arrange all the sum vector (sentence scores) from highest value to lowest value.. This approach is used to replace other graph-based methods measures: term frequency, tf, and inverse document frequency, idf, which are used for more than half a century by researchers until this date. Our new approach can be summarized as follows:
1. Generate the symmetric matrix with the exact number of edges between the nodes (i.e., the sentences).

2. Algebraically sum the rows of the matrix to produce rand vector (sentence scores).

3. Sort the ranking vector to get the sentence rank.

4. Normalize all the sentence scores to get all the sentence score values in the range 0 to 1 using MinMax Normalization Technique.

5. Apply the cut-off mechanism using the required threshold to produce the summary.

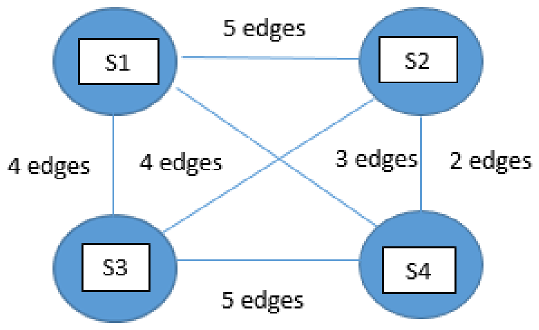


Figure 3.1: Frequency of word occurance analysis example

Table 3.1 shows comparisons between the sentences and themselves as 0. For example, the comparison S1-S1, S2-S2, S3-S3, and S4-S4 are 0 in the matrix. Actually,

Word in Sentence 1	Match(s) in Sentence 2
S1W1	
S1W2	S2W2
S1W3	
S1W4	
S1W5	
S1W6	
S1W7	S2W6
S1W8	
S1W9	S2W1
S1W10	
S1W11	
S1W12	S2W11
S1W13	
S1W14	
S1W15	
S1W16	
S1W17	
S1W12	S2W11
S1W18	
Totak edges	5

Table 3.2: Number of edges for our example

the comparison(s) between S1 to S1 is 100 percent but we are assuming it as 0 in the matrix as it has no impact on the summarization process.

The proposed method is efficient, simple and fast. Almost all methods have preprocessing schemes. For, our algorithm we tried to test its performance with and without preprocessing. We found that its performance has increased significantly when the preprocessing scheme was employed. Pre-processing reduces the size of the matrix by a considerable amount, which increases the performance and accuracy of the algorithm. Pre-processing mainly includes removal of articles, prepositions and meaningless words (like a sentences starting with a bracket or any special character). Advanced stage of pre-processing includes tagging (nouns, pronouns, adjectives etc.), semantics, stemming were not used in this research. Such advanced preprocessing will be considered for further research in the future.

3.2.1 Feasibility Study:

There are three parts in feasibility study:

Operational Feasibility:

This system can be implemented in the business sector due to the boom in this sector in recent years. The system is expected to operate properly irrespective of external conditions with the exceptions being poor hardware and power constraints.

Technical Feasibility:

The technology being used to implement the system is one of the latest trends, which is best suited for the proposed system. The proposed equipment for the system to be implemented is a website which can be run on any device with a browser.

Financial and Economic Feasibility:

The necessary technology to implement the system is freely available. The system can be developed and operated in the existing hardware and software infrastructure. This will reduce the man power which will therefore, benefit the economy of the business sector.

Chapter 4

DETERMINING THE VOCABULARY

4.1 Tokenization

What does Tokenization mean?
Tokenization is the act of breaking up a sequence of strings into pieces such as words, keywords, phrases, symbols and other elements called tokens. Tokens can be individual words, phrases or even whole sentences. In the process of tokenization, some characters like punctuation marks are discarded. The tokens become the input for another process like parsing and text mining.Tokenization is used in computer science, where it plays a large part in the process of lexical analysis.NLTK provides a number of tokenizers in the tokenize module [12] [13].
Tokenization relies mostly on simple heuristics in order to separate tokens by following a few steps:

- Tokens or words are separated by whitespace, punctuation marks or line breaks
- White space or punctuation marks may or may not be included depending on the need
- All characters within contiguous strings are part of the token. Tokens can be made up of all alpha characters, alphanumeric characters or numeric characters only.

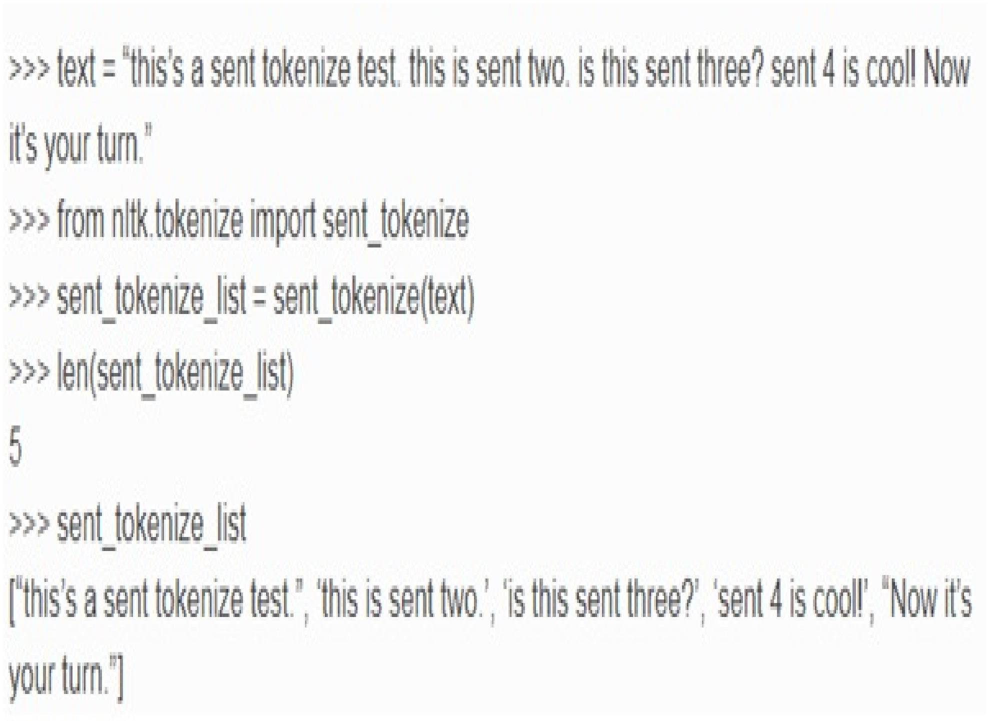
These issues of tokenization are language-specific. It thus requires the language of the document to be known. Language identification based on classifiers identification that use short character subsequences as features is highly effective; most languages have distinctive signature patterns.

Tokenizers is used to divide strings into lists of substrings. For example, Sentence tokenizer can be used to find the list of sentences and Word tokenizer can be used to find the list of words in strings.

4.1.1 Sentence Tokenizer

Sentence Tokenize or Sentence boundary disambiguation which is also known as sentence breaking, is the problem in natural language processing of deciding where sentences begin and end.
There are many NLP tools available which have sentence tokenise fuction such as NLTK,OpenNLP etc.

How to use sentence tokenize in NLTK? [12] After installing nltk and nltk_data , you can launch python and import sent_tokenize tool from nltk as shown in the figure 4.1 to get a list of sentences as the output.



```
>>> text = "this's a sent tokenize test. this is sent two. is this sent three? sent 4 is cool! Now
it's your turn."
>>> from nltk.tokenize import sent_tokenize
>>> sent_tokenize_list = sent_tokenize(text)
>>> len(sent_tokenize_list)
5
>>> sent_tokenize_list
["this's a sent tokenize test.", 'this is sent two.', 'is this sent three?', 'sent 4 is cool!', "Now it's
your turn."]
```

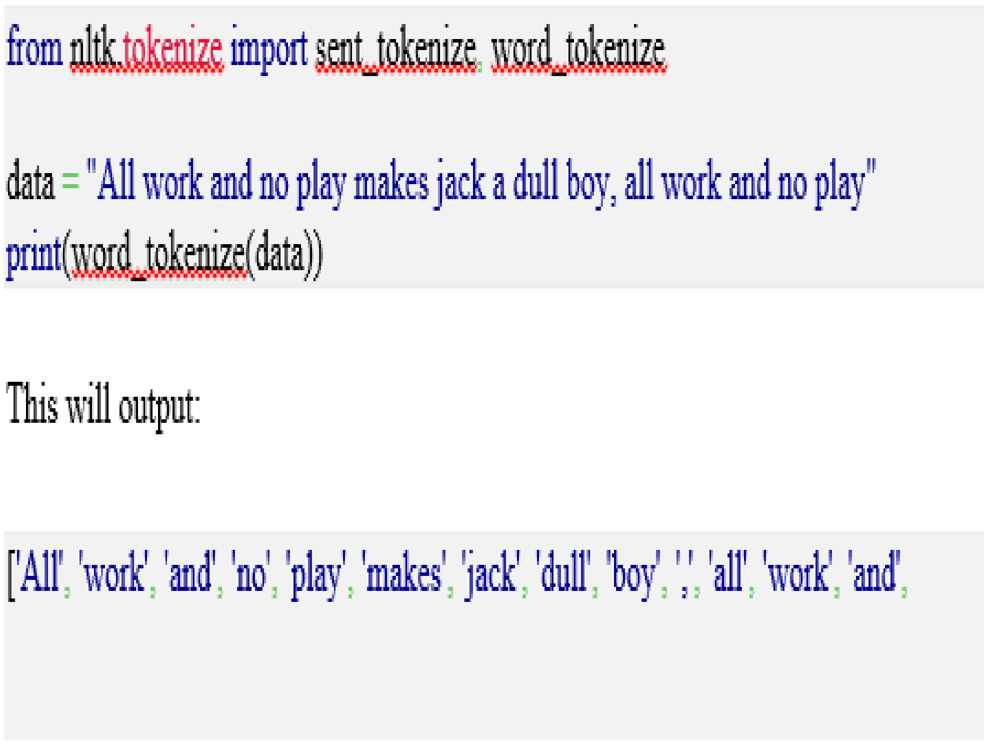
Figure 4.1: Example for sentence tokenization using NLTK

sent_tokenize uses an instance of PunktSentenceTokenizer from the nltk. tokenize.punkt module. This instance has already been trained on and works well for many European languages. So it knows what punctuation and characters mark the end of a sentence and the beginning of a new sentence.

4.1.2 Word Tokeniser

Word Tokenize is the act of breaking a sentence into words.This process returns a list all the words in a given input sentence.

How to use word tokenize in NLTK? A sentence or data can be split into words using the method word_tokenize()which can be imported and used as shown below.



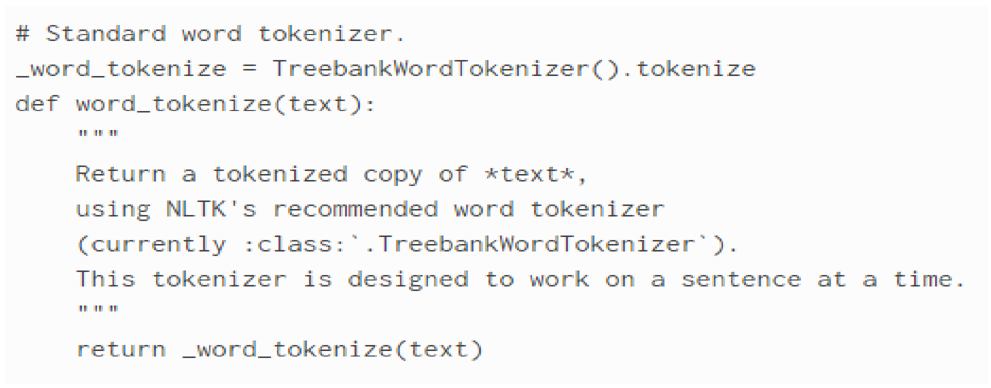
```
from nltk.tokenize import sent_tokenize, word_tokenize

data = "All work and no play makes jack a dull boy, all work and no play"
print(word_tokenize(data))
```

This will output:

```
['All', 'work', 'and', 'no', 'play', 'makes', 'jack', 'dull', 'boy', ',', 'all', 'work', 'and',
```

Figure 4.2: Example for word tokenization using NLTK

word_tokenize is a wrapper function that calls tokenize by the TreebankWordTokenizer.The code for the same is shown in figure 4.3



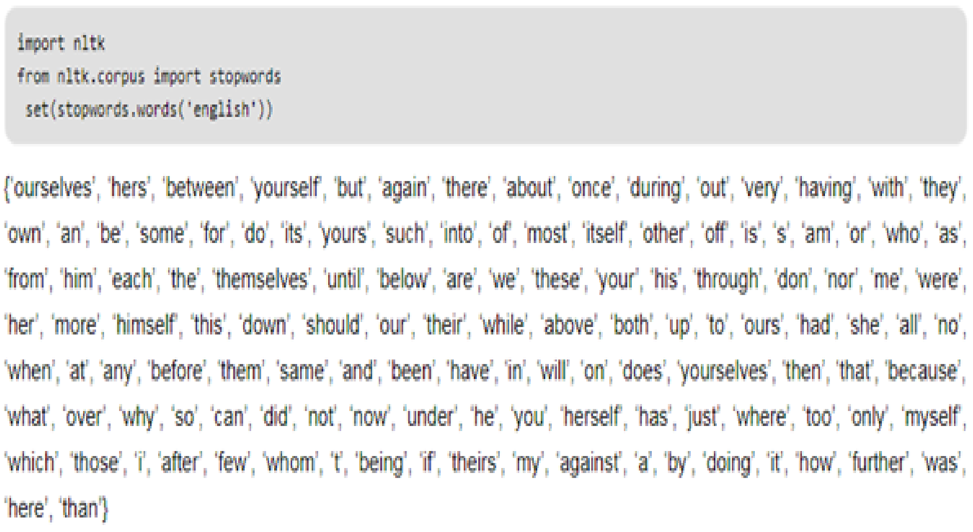
```
# Standard word tokenizer.
_word_tokenize = TreebankWordTokenizer().tokenize
def word_tokenize(text):
    """
    Return a tokenized copy of *text*,
    using NLTK's recommended word tokenizer
    (currently :class:`.TreebankWordTokenizer`).
    This tokenizer is designed to work on a sentence at a time.
    """
    return _word_tokenize(text)
```

Figure 4.3: Standard word tokenizer

4.2 Dropping common terms: stop words

Sometimes, some extremely common words which would appear to be of little value in helping select documents matching a user need are excluded STOP WORDS from the vocabulary entirely. These words are called stop words.[12]

We would not want these words taking up space in our database, or taking up valuable processing time. For this, we can remove them easily, by storing a list of words that you consider to be stop words. NLTK(Natural Language Toolkit) in python has a list of stopwords stored in 16 different languages. You can find them in the nltk_data directory. home/pratima/nltk_data/corpora/stopwords is the directory address.

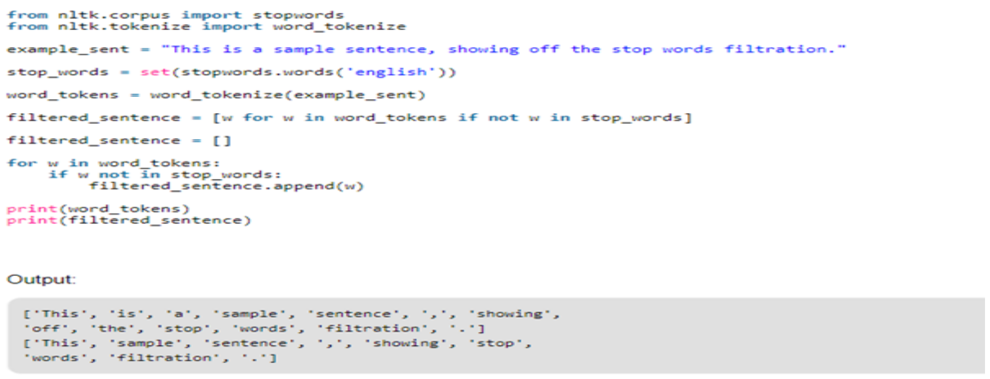


```
import nltk
from nltk.corpus import stopwords
 set(stopwords.words('english'))
```

```
{'ourselves', 'hers', 'between', 'yourself', 'but', 'again', 'there', 'about', 'once', 'during', 'out', 'very', 'having', 'with', 'they',
'own', 'an', 'be', 'some', 'for', 'do', 'its', 'yours', 'such', 'into', 'of', 'most', 'itself', 'other', 'off', 'is', 's', 'am', 'or', 'who', 'as',
'from', 'him', 'each', 'the', 'themselves', 'until', 'below', 'are', 'we', 'these', 'your', 'his', 'through', 'don', 'nor', 'me', 'were',
'her', 'more', 'himself', 'this', 'down', 'should', 'our', 'their', 'while', 'above', 'both', 'up', 'to', 'ours', 'had', 'she', 'all', 'no',
'when', 'at', 'any', 'before', 'them', 'same', 'and', 'been', 'have', 'in', 'will', 'on', 'does', 'yourselves', 'then', 'that', 'because',
'what', 'over', 'why', 'so', 'can', 'did', 'not', 'now', 'under', 'he', 'you', 'herself', 'has', 'just', 'where', 'too', 'only', 'myself',
'which', 'those', 'i', 'after', 'few', 'whom', 't', 'being', 'if', 'theirs', 'my', 'against', 'a', 'by', 'doing', 'it', 'how', 'further', 'was',
'here', 'than'}
```

Figure 4.4: Set of English stop words in NLTK

In NLTK removal of stop words is done as shown in the figure 4.5



```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

example_sent = "This is a sample sentence, showing off the stop words filtration."

stop_words = set(stopwords.words('english'))

word_tokens = word_tokenize(example_sent)

filtered_sentence = [w for w in word_tokens if not w in stop_words]

filtered_sentence = []

for w in word_tokens:
    if w not in stop_words:
        filtered_sentence.append(w)

print(word_tokens)
print(filtered_sentence)
```

Output:

```
['This', 'is', 'a', 'sample', 'sentence', ',', 'showing',
'off', 'the', 'stop', 'words', 'filtration', '.']
['This', 'sample', 'sentence', ',', 'showing', 'stop',
'words', 'filtration', '.']
```

Figure 4.5: Example in NLTK to depicting the removal of stop words

4.3 Normalization

Normalization can refer to different techniques depending on context. Here, we use normalization to refer to rescaling an input variable to the range between 0 and 1.Normalization requires that you know the minimum and maximum values for each attribute.This can be estimated from training data or specified directly if you have deep knowledge of the problem domain.You can easily estimate the minimum and maximum values for each attribute in a dataset by enumerating through the values.

Normalization can be useful, and even required in some machine learning algorithms when your time series data has input values with differing scales.It may be required for algorithms, like k-Nearest neighbors, which uses distance calculations and Linear Regression and Artificial Neural Networks that weight input values.

Normalization requires that you know or are able to accurately estimate the minimum and maximum observable values. You may be able to estimate these values from your available data. If your time series is trending up or down, estimating these expected values may be difficult and normalization may not be the best method to use on your problem.

$$y = \frac{(x - min)}{(max - min)} \tag{4.1}$$

Where the minimum and maximum values pertain to the value x being normalized. For example, for the temperature data, we could guesstimate the min and max observable values as 30 and -10, which are greatly over and under-estimated. We can then normalize any value like 18.8 as follows:

$$y = \frac{(x - min)}{(max - min)}$$
$$y = \frac{(18.8 - (-10))}{(30 - (-10))}$$
$$y = 0.72$$

You can see that if an x value is provided that is outside the bounds of the minimum and maximum values, that the resulting value will not be in the range of 0 and 1. You could check for these observations prior to making predictions and either remove them from the dataset or limit them to the pre-defined maximum or minimum values. You can normalize your dataset using the scikit-learn object MinMaxScaler.Good practice usage with the MinMaxScaler and other rescaling techniques is as follows:

1. Fit the scaler using available training data. For normalization, this means the training data will be used to estimate the minimum and maximum observable values. This is done by calling the fit() function.

2. Apply the scale to training data. This means you can use the normalized data to train your model. This is done by calling the transform() function

3. Apply the scale to data going forward. This means you can prepare new data in the future on which you want to make predictions.

4.4 Stemming and Lemmatization

For grammatical reasons, documents are going to use different forms of a word, such as organize, organizes, and organizing. Additionally, there are families of derivationally related words with similar meanings, such as democracy, democratic, and democratization. In many situations, it seems as if it would be useful for a search for one of these words to return documents that contain another word in the set.

Stemming and Lemmatization are the basic text processing methods for English text. The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. Stemming and lemmatization are essential for many text mining tasks such as information retrieval, text summarization, topic extraction as well as translation.

4.4.1 Stemming

In linguistic morphology and information retrieval, stemming is the process for reducing inflected (or sometimes derived) words to their stem, base or root formgenerally a written word form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. Algorithms for stemming have been studied in computer science since the 1960s. Many search engines treat words with the same stem as synonyms as a kind of query expansion, a process called conflation. Stemming programs are commonly referred to as stemming algorithms or stemmers.

The best-known and most popular stemming approach for English is the Porter stemming algorithm, also known as the Porter stemmer. It is a collection of rules (or, if you prefer, heuristics) designed to reflect how English handles inflections. For example, the Porter stemmer chops both apple and apples down to appl, and it stems berry and berries to berri.

If we apply a stemmer to queries and indexed documents, we can increase recall by matching words against their other inflected forms. It is critical that we apply the same stemmer to both queries and documents.

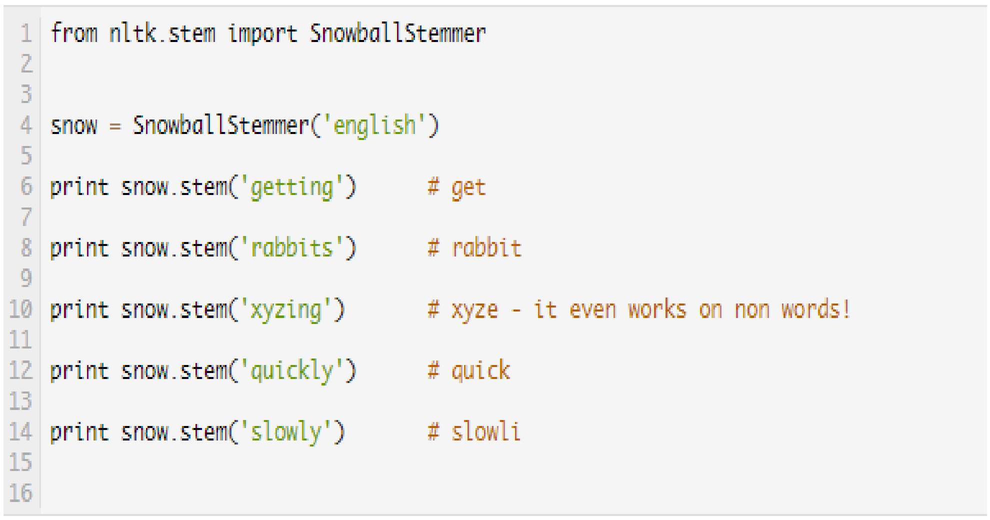
You can find an implementation of the Porter stemmer in any major natural language processing library, such as NLTK and the Stanford NLP suite. You can find stemmers for other languages (or create your own) in Snowball.

Just as using a knife to chop a mushroom stem may leave a bit of the stem or cut into the cap, stemming algorithms sometimes remove too little or too much. For example, Porter stems both meanness and meaning to mean, creating a false equivalence. On the other hand, Porter stems goose to goos and geese to gees, when those two words should be equivalent.

In general, stemming algorithms err on the side of being too aggressive, sacrificing precision in order to increase recall.

4.4.1.1 How do stemmers work

Stemmers are extremely simple to use and very fast. They usually are the preferred choice. They work by applying different transformation rules on the word until no other transformation can be applied.

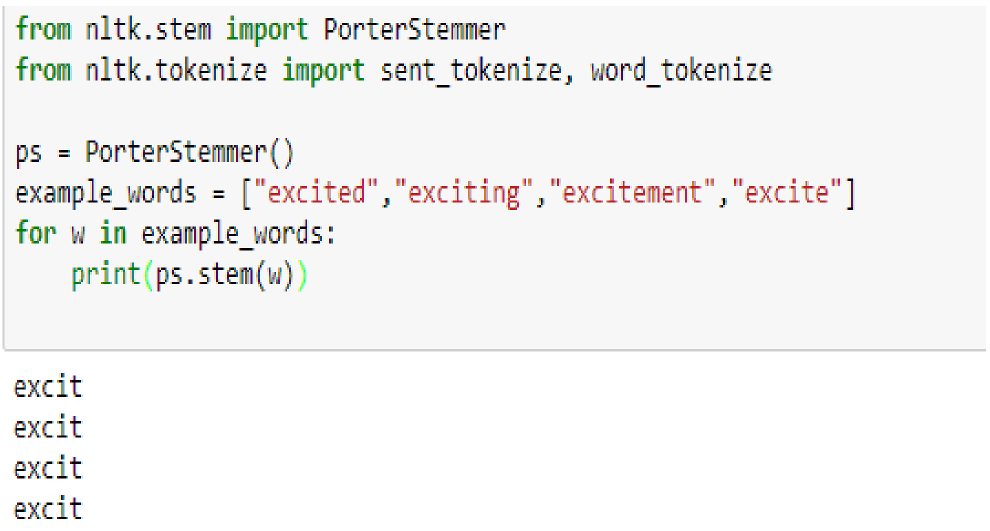


```
from nltk.stem import SnowballStemmer


snow = SnowballStemmer('english')

print snow.stem('getting')      # get

print snow.stem('rabbits')      # rabbit

print snow.stem('xyzing')       # xyze - it even works on non words!

print snow.stem('quickly')      # quick

print snow.stem('slowly')       # slowli


```

Figure 4.6: Example of Stemming

4.4.1.2 How to use Stemmer in NLTK:

NLTK provides several famous stemmers interfaces, such as **Porter stemmer, Lancaster Stemmer, Snowball Stemmer** and etc. In NLTK, using those stemmers is very simple.



```
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize

ps = PorterStemmer()
example_words = ["excited","exciting","excitement","excite"]
for w in example_words:
    print(ps.stem(w))
```

```
excit
excit
excit
excit
```

Figure 4.7: Example of Porter Stemmer

4.4.2 Lemmatization:

Lemmatisation (or lemmatization) in linguistics, is the process of grouping together the different inflected forms of a word so they can be analysed as a single item.

In computational linguistics, lemmatisation is the algorithmic process of determining the lemma for a given word. Since the process may involve complex tasks such as understanding context and determining the part of speech of a word in a sentence (requiring, for example, knowledge of the grammar of a language) it can be a hard task to implement a lemmatiser for a new language.

In many languages, words appear in several inflected forms. For example, in English, the verb to walk may appear as walk, walked, walks, walking. The base form, walk, that one might look up in a dictionary, is called the lemma for the word. The combination of the base form with the part of speech is often called the lexeme of the word.

Lemmatisation is closely related to stemming. The difference is that a stemmer operates on a single word without knowledge of the context, and therefore cannot discriminate between words which have different meanings depending on part of speech.

However, stemmers are typically easier to implement and run faster, and the reduced accuracy may not matter for some applications.

Lemmatization tries to take a similarly careful approach to removing inflections. Lemmatization does not simply chop off inflections, but instead relies on a lexical knowledge base like WordNet to obtain the correct base forms of words.

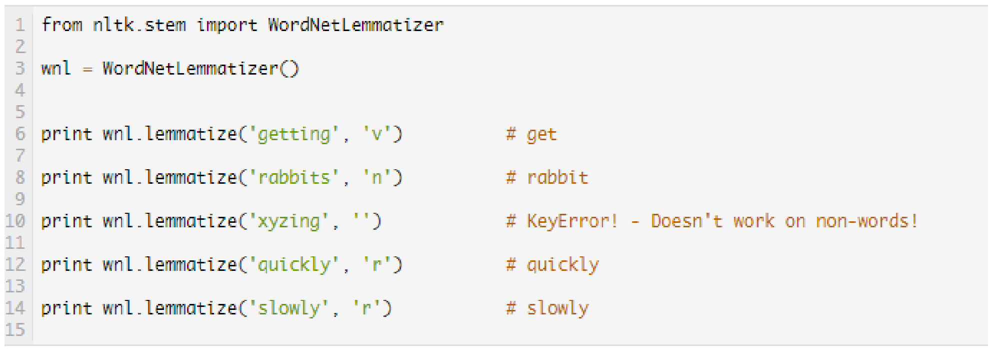
For example, WordNet lemmatizes geese to goose and lemmatizes meanness and meaning to themselves. In these examples, it outperforms than the Porter stemmer.

But lemmatization has limits. For example, Porter stems both happiness and happy to happi, while WordNet lemmatizes the two words to themselves. The WordNet lemmatizer also requires specifying the words part of speechotherwise, it assumes the word is a noun. Finally, lemmatization cannot handle unknown words: for example, Porter stems both iphone and iphones to iphon, while WordNet lemmatizes both words to themselves.

In general, lemmatization offers better precision than stemming, but at the expense of recall.

4.4.2.1 How do lemmatizers work

As previously mentioned, lemmatizers need to know about the part of speech. This is a substantial dissadvantage since the task of Part-Of-Speech tagging is prone to errors. Heres how to properly use a lemmatizer:



```
from nltk.stem import WordNetLemmatizer

wnl = WordNetLemmatizer()


print wnl.lemmatize('getting', 'v')         # get

print wnl.lemmatize('rabbits', 'n')         # rabbit

print wnl.lemmatize('xyzing', '')           # KeyError! - Doesn't work on non-words!

print wnl.lemmatize('quickly', 'r')         # quickly

print wnl.lemmatize('slowly', 'r')          # slowly

```

Figure 4.8: Example of Lemmatization

4.4.3 Canonicalization:

As weve seen, stemming and lemmatization are effective techniques to expand recall, with lemmatization giving up some of that recall to increase precision. But both techniques can feel like crude instruments.

If we generalize from stemming and lemmatization, what we have are ways to group together the related forms of a word, assigning them all a canonical form. While its easy to rely on heuristics like Porter stemming and WordNet lemmatization, theres nothing to stop us from building our own knowledge base for canonicalization.

For example, we can use the equivalence classes from Porter stemming and Wordnet lemmatization as candidates, and then use further processingeither automatic or editorialto improve precision by estimating word similarity.

The automatic detection of word similarity could be based on corpus analysis (e.g., word embeddings like word2vec) or searcher behavior (queries, clicks from those queries, etc). Ideally, we use all available signals to train a machine-learned model.

Editorial detection requires human judgements. These could be at the level of token-token pairs, query-query pairs (to assess the tokens in context), or query-document pairs (to assess the end-to-end effect of query expansion.

Building your own knowledge base for canonicalization may sound like a lot of work, and it can be. I recommend that you start by leveraging off-the-shelf tools like Porter and WordNet, and then determine how much you want to invest in improving on them.

4.4.4 Stemmers vs Lemmatizers

- Both stemmers and lemmatizers try to bring inflected words to the same form

- Stemmers use an algorithmic approach of removing prefixes and suffixes. The result might not be an actual dictionary word.

- Lemmatizers use a corpus. The result is always a dictionary word.

- Lemmatizers need extra info about the part of speech they are processing. Calling can be either a verb or a noun (the calling)

- Stemmers are faster than lemmatizers.

4.4.5 When to use stemmers and when to use lemmatizers

- If speed is important, use stemmers (lemmatizers have to search through a corpus while stemmers do simple operations on a string)

- If you just want to make sure that the system you are building is tolerant to inflections, use stemmers (If you query for best bar in New York, youd accept an article on Best bars in New York 2016)

- If you need the actual dictionary word, use a lemmatizer. (for example, if you are building a natural language generation system)

Chapter 5

SYSTEM REQUIREMENT SPECIFICATION

Software requirement Specification is a fundamental document, which forms the foundation of the software development process. It not only lists the requirements of a system but also has a description of its major feature. An SRS is basically an organization's understanding (in writing) of a customer or potential client's system requirements and dependencies at a particular point in time (usually) prior to any actual design or development work. It's a two-way insurance policy that assures that both the client and the organization understand the other's requirements from that perspective at a given point in time.

The SRS also functions as a blueprint for completing a project with as little cost growth as possible. The SRS is often referred to as the "parent" document because all subsequent project management documents, such as design specifications, statements of work, software architecture specifications, testing and validation plans, and documentation plans, are related to it. It is important to note that an SRS contains functional and nonfunctional requirements only; it doesn't offer design suggestions, possible solutions to technology or business issues, or any other information other than what the development team understands the customer's system requirements to be.

5.1 Functional Requirement

Functional Requirement defines a function of a software system and how the system must behave when presented with specific inputs or conditions. These may include calculations, data manipulation and processing and other specific functionality. In this system following are the functional requirements:

- Input test case must not have compilation and runtime errors.
- The application must not stop working when kept running for even a long time.
- The application must function as expected for every set of test cases provided.
- The application should generate the output for given input test case and input parameters.
- The application should generate on-demand services

5.2 Non- Functional Requirement

Non-functional requirements are the requirements which are not directly concerned with the specific function delivered by the system. They specify the criteria that can be used to judge the operation of a system rather than specific behaviors. They may relate to emergent system properties such as reliability, response time and store occupancy. Non-functional requirements arise through the user needs, because of budget constraints, organizational policies, the need for interoperability with other software and hardware systems or because of external factors such as -

- Product Requirements
- Organizational Requirements
- User Requirements
- Basic Operational Requirements

In systems engineering and requirements engineering, a non functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. This should be contrasted with functional requirements that define specific behaviour or functions. The plan for implementing non-functional requirements is detailed in the system architecture. Broadly, functional requirements define what a system is supposed to do and non- functional requirements define how a system is supposed to be. Functional requirements are usually in the form of system shall do ¡requirement¿, an individual action of part of the system,perhaps explicitly in the sense of a mathematical function, a black box description input, output, process and control functional model or IPO Model. In contrast, non-functional requirements are in the form of system shall be requirement, an overall property of the system as a whole or of a particular aspect and not a specific function.The systems' overall properties commonly mark the difference between whether the development project has succeeded or failed.Non-functional requirements of our project include:

- Response time The time the system takes to load and the time for responses on any action the user does.
- Processing time - How long is acceptable to perform key functions or export /import data?
- Throughput The number of transactions the system needs to handle must be kept in mind.
- Storage - The amount of data to be stored for the system to function.
- Growth Requirements As the system grows it will need more storage space to keep up with the efficiency.
- Locations of operation - Geographic location, connection requirements and the restrictions of a local network prevail.
- Architectural Standards The standards needed for the system to work and sustain.

5.2.1 Organisational Requirements

Process Standards: IEEE standards are used to develop the application which is the standard used by the most of the standard software developers all over the world.Design Methods: Design is one of the important stages in the software engineering process. This stage is the first step in moving from problem to the solution domain.In other words, starting with what is needed design takes us to work how to satisfy the needs.

5.2.2 User Requirements

The user requirements document (URD) or user requirements specification is a document usually used to software engineering that specifies the requirements the user expects from software to be constructed in a software project. Once the required information is completely gathered it is documented in a URD, which is meant to spell out exactly what the software must do and becomes part of the contractual agreement. A customer cannot demand feature not in the URD, whilst the developer cannot claim the product is ready if it does not meet an item of the URD. The URD can be used as a guide to planning cost, timetables, milestones, testing etc. The explicit nature of the URD allows customers to show it to various stakeholders to make sure all necessary features are described. Formulating a URD requires negotiation to determine what is technically and economically feasible. Preparing a URD is one of those skills that lies between a science and economically feasible. Preparing a URD is one of those skills that lies between a science and an art,requiring both software technical skills and interpersonal skills.

5.3 Hardware Requirements

Processor Speed : 2.20 GHZ
RAM : 3 GB
Storage : 320 GB
Keyboard : Standard 102 keys
Mouse : Standard 3 buttons

5.4 Software Requirements

Operating system : Windows 7 and above
Coding Language : python
Tools : anaconda navigator,nltk,Jupyter notebook

5.5 Description on the tools

Anaconda Navigator is a desktop graphical user interface included in Anaconda that allows you to launch applications and easily manage conda packages, environments and channels without the need to use command line commands.

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

Natural Language Processing with Python provides a practical introduction to programming for language processing. Written by the creators of NLTK, it guides the reader through the fundamentals of writing Python programs, working with corpora, categorizing text, analyzing linguistic structure, and more. The online version of the book has been been updated for Python 3 and NLTK 3.

5.6 Software quality attributes

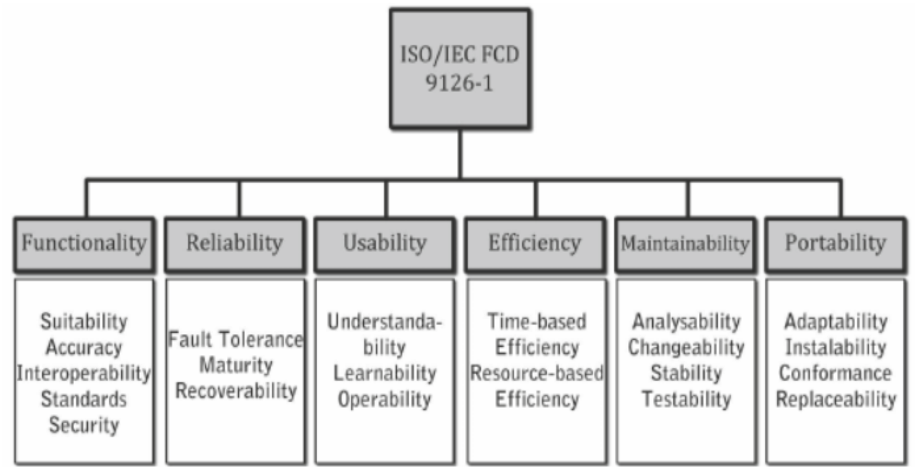


Figure 5.1: Software Quality Attributes

Functionality: the capability of the software to provide functions which meet stated and implied needs when the software is used under specified conditions.

- Reliability: the capability of the software to maintain its level of performance when used under specified conditions.
- Usability: the capability of the software to be understood, learned, used and liked by the user, when used under specified conditions.
- Efficiency: the capability of the software to provide the required performance,relative to the amount of resources used, under stated conditions.
- Maintainability: the capability of the software to be modified. Modifications may include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications.
- Portability: the capability of software to be transferred from one environment to another.

Chapter 6

SYSTEM ANALYSIS

6.1 Overview

Analysis is the process of finding the best solution to the problem. System analysis is the process by which we learn about the existing problems, define objects and requirements and evaluates the solutions. It is the way of thinking about the organization and the problem it involves, a set of technologies that helps in solving these problems. Feasibility study plays an important role in system analysis which gives the target for design and development.

6.2 Feasibility Study

All systems are feasible when provided with unlimited resource and infinite time. But unfortunately, this condition does not prevail in practical world. So, it is both necessary and prudent to evaluate the feasibility of the system at the earliest possible time. Months or years of effort, thousands of rupees and untold professional embarrassment can be averted if an ill-conceived system is recognized early in the definition phase. Feasibility and risk analysis are related in many ways. If project risk is great, the feasibility of producing quality software is reduced. In this case there are three primary areas of interest: -

6.2.1 Performance Analysis

For the complete functionality of the project work, the project is run with the help of healthy networking environment. Normally, the OS is windows 7. The main theme of this project is design a system that converts signs into speech. Performance analysis is done to find out whether the proposed system is time efficient and accurate. It is essential that the process of performance analysis and definition must be conducted in parallel.

6.2.2 Technical Analysis

System is only beneficial only if it can be turned into information systems that will meet the organizations technical requirement. Simply stated this test of feasibility asks whether the system will work or not when developed and installed, whether there are any major barriers to implementation. Regarding all these issues in technical analysis.

There are several points to focus on: -

Changes to bring in the system: All changes should be in positive direction, there will be increased level of efficiency and better customer service.

Required skills: Platforms and tools used in this project are widely used. So the skilled manpower is readily available in the industry.

Acceptability: The structure of the system is kept feasible enough so that there should not be any problem from the users point of view.

6.2.3 Economic Analysis

Economic analysis is performed to evaluate the development cost weighed against the ultimate income or benefits derived from the developed system. For running this system, we simply need a computer. So, the system is economically feasible enough.

6.3 Summary

The main aim of this chapter is to find out whether the system is feasible enough or not. For these reasons different kinds of analysis, such as performance analysis, technical analysis, economic analysis etc. is performed.

6.4 F-measure

In statistical analysis of binary classification, the F1 score (also F-score or F-measure) is a measure of a test's accuracy. It considers both the precision p and the recall r of the test to compute the score: p is the number of correct positive results divided by the number of all positive results returned by the classifier, and r is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive). The F1 score is the harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

Chapter 7

SYSTEM DESIGN

7.1 Overview

Design is a meaningful engineering representation of something that is to be built. It is the most crucial phase in the developments of a system. Software design is a process through which the requirements are translated into a representation of software. Design is a place where design is fostered in software Engineering. Based on the user requirements and the detailed analysis of the existing system, the new system must be designed. This is the phase of system designing. Design is the perfect way to accurately translate a customers requirement in the finished software product. Design creates a representation or model, provides details about software data structure, architecture, interfaces and components that are necessary to implement a system. The logical system design arrived at as a result of systems analysis is converted into physical system design.

7.1.1 System development methodology

System development method is a process through which a product will get completed or a product gets rid from any problem. Software development process is described as a number of phases, procedures and steps that gives the complete software. It follows series of steps which is used for product progress. The development method followed in this project is waterfall model.

7.1.1.1 Model phases

The waterfall model is a sequential software development process, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Requirement initiation, Analysis, Design, Implementation, Testing and maintenance.

- **Requirement Analysis:** This phase is concerned about collection of requirement of the system. This process involves generating document and requirement review.
- **System Design:** Keeping the requirements in mind the system specifications are translated in to a software representation. In this phase the designer emphasizes on:-algorithm, data structure, software architecture etc.
- **Coding:** In this phase programmer starts his coding in order to give a full sketch of product. In other words, system specifications are only converted in to machine readable compute code.
- **Implementation**: The implementation phase involves the actual coding or programming of the software. The output of this phase is typically the library, executables, user manuals and additional software documentation
- **Testing:** In this phase all programs (models) are integrated and tested to ensure that the complete system meets the software requirements. The testing is concerned with verification and validation.
- **Maintenance:** The maintenance phase is the longest phase in which the software is updated to fulfill the changing customer need, adapt to accommodate change in the external environment, correct errors and oversights previously undetected in the testing phase, enhance the efficiency of the software.

7.1.1.2 Reason for choosing Waterfall Model as development method

- Clear project objectives.
- Stable project requirements.
- Progress of system is measurable.
- Strict sign-off requirements.
- Helps you to be perfect.
- Logic of software development is clearly understood.
- Production of a formal specification
- Better resource allocation.
- Improves quality. The emphasis on requirements and design before writing a single line of code ensures minimal wastage of time and effort and reduces the risk of schedule slippage.

- Less human resources required as once one phase is finished those people can start working on to the next phase.

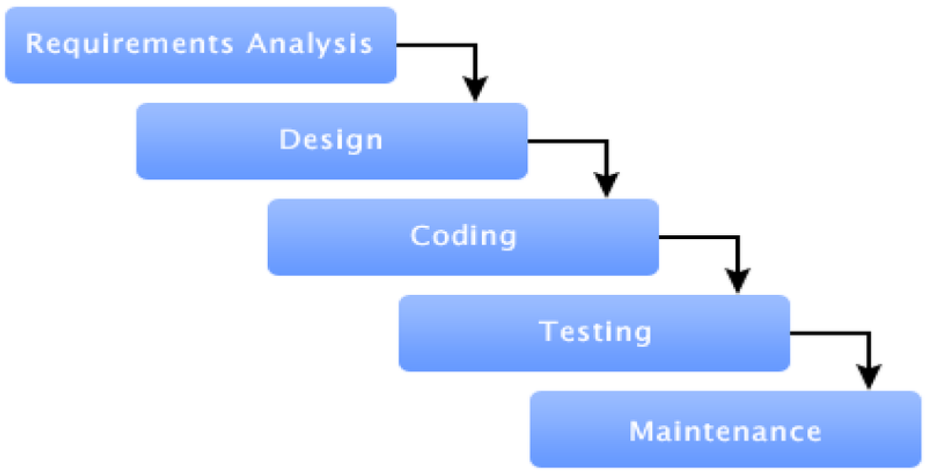
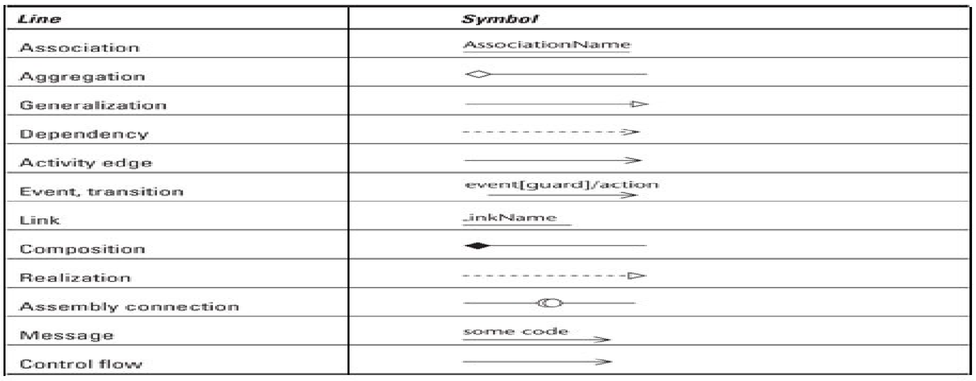


Figure 7.1: Waterfall model

7.2 Data flow diagram

A data flow diagram (DFD) is graphic representation of the ”flow” of data through an information system. A data flow diagram can also be used for the visualization of data processing (structured design). It is common practice for a designer to draw a context-level DFD first which shows the interaction between the system and outside entities.DFDs show the flow of data from external entities into the system, how the data moves from one process to another, as well as its logical storage. There are only four symbols: 1. Squares representing external entities, which are sources and destinations of information entering and leaving the system. 2. Rounded rectangles representing processes, in other methodologies, may be called ’Activities’, ’Actions’, ’Procedures’,’Subsystems’ etc. which take data as input, do processing to it, and output it. 3.Arrows representing the data flows, which can either, be electronic data or physical items. It is impossible for data to flow from data store to data store except via a process, and external entities are not allowed to access data stores directly. 4. The flat three-sided rectangle is representing data stores should both receive information for storing and provide it for further processing.



Line	*Symbol*
Association	AssociationName
Aggregation	◇———
Generalization	———▷
Dependency	- - - - - - ->
Activity edge	———>
Event, transition	event[guard]/action ———>
Link	LinkName
Composition	◆———
Realization	- - - - - - -▷
Assembly connection	———○———
Message	some code ———>
Control flow	———>

Figure 7.2: Symbols used in UML

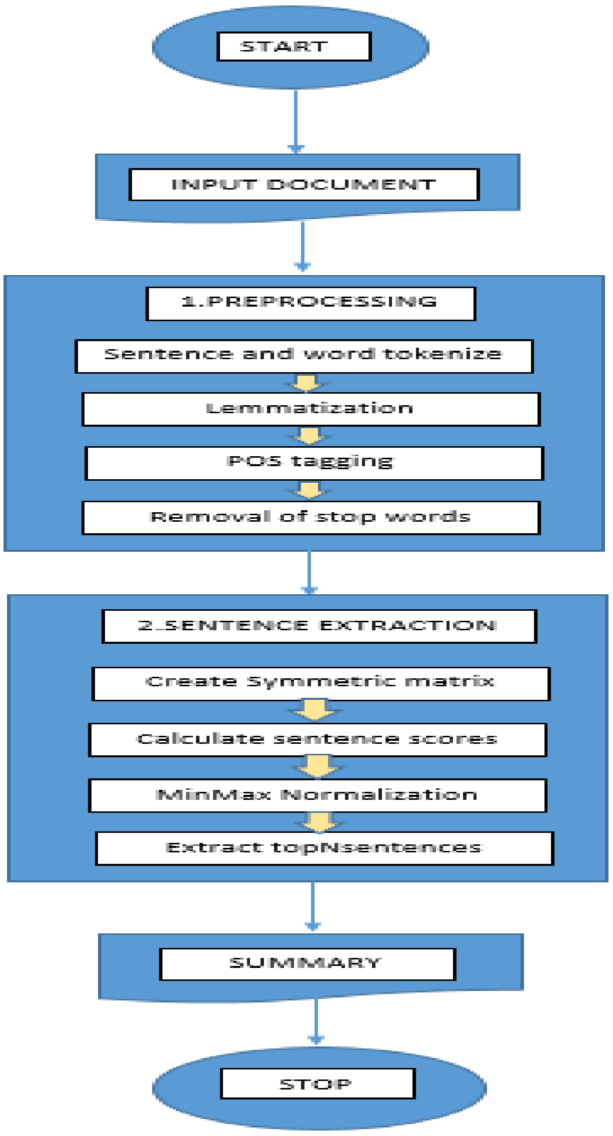


Figure 7.3: Data Flow Diagram For Summarization

Chapter 8

IMPLEMENTATION

8.1 Introduction

The implementation phase of the project is where the detailed design is actually transformed into working code. Aim of the phase is to translate the design into a best possible solution in a suitable programming language. This chapter covers the implementation aspects of the project, giving details of the programming language and development environment used. It also gives an overview of the core modules of the project with their step by step flow. The implementation stage requires the following tasks:

- Careful planning.
- Investigation of system and constraints.
- Design of methods to achieve the changeover.
- Evaluation of the changeover method.
- Correct decisions regarding selection of the platform.
- Appropriate selection of the language for application development.

8.2 Code

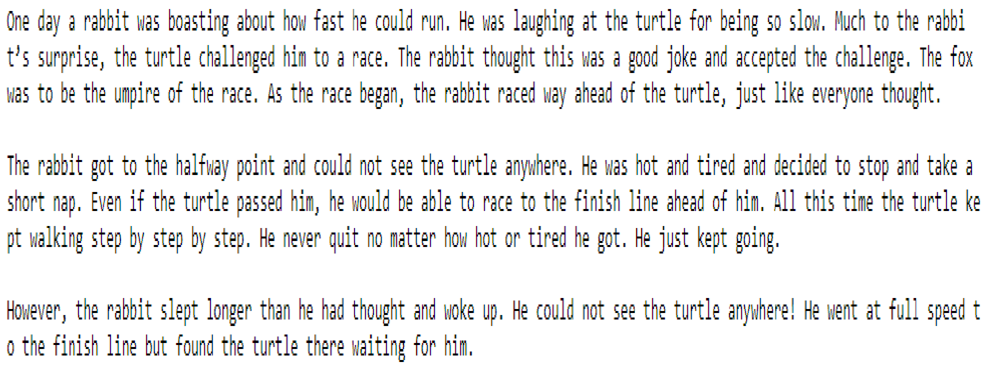
8.2.1 Pre-processing

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing.

The code for pre-processing an input document is shown below:

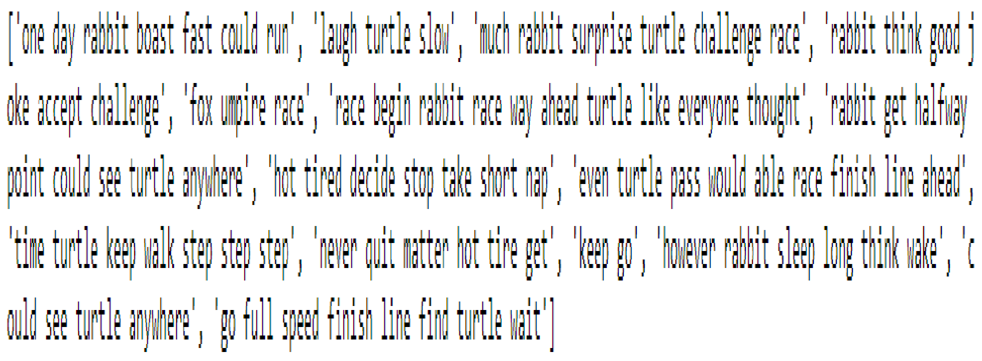
```
import nltk
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import wordnet
from nltk.tokenize import sent_tokenize
import re
from nltk import word_tokenize
parentFolder = 'c:\\Users\\Nammu\\Anaconda3'
inputFileName="pracnltk.txt"
inpFile = open(parentFolder+'/'+inputFileName,"r")
inputDoc = inpFile.read()
print(inputDoc)
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
lmtzr = WordNetLemmatizer()
stop = set(stopwords.words('english'))
from nltk.corpus import wordnet
def get_wordnet_pos(treebank_tag):

    if treebank_tag.startswith('J'):
        return wordnet.ADJ
    elif treebank_tag.startswith('V'):
        return wordnet.VERB
    elif treebank_tag.startswith('N'):
        return wordnet.NOUN
    elif treebank_tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN

import nltk
import re

from nltk import pos_tag
def preProcessSentence(raw_sentence):
    clean_sentence = re.sub("[^a-zA-Z]", " ", raw_sentence
    clean_sentence = re.sub(' +', ' ',clean_sentence)
    words =clean_sentence.split()
    tokensList = [w for w in words]
    pos_tag = nltk.pos_tag(tokensList)
    lemmatized_sent = []
    for i in range(len(pos_tag)):
        lemmatized_sent.append(lmtzr.lemmatize(pos_tag[i][0].lower(),


        get_wordnet_pos(pos_tag[i][1])))
    lemmatized_sent= " ".join(lemmatized_sent)
    lowercaseTokens =lemmatized_sent.lower().split()
    meaningful_words = [w for w in lowercaseTokens if not w in stop]
    return( " ".join(meaningful_words))

import nltk
import re

from nltk import pos_tag
def getNouns(raw_sentence):
    clean_sentence = re.sub("[^a-zA-Z]", " ", raw_sentence)
    clean_sentence = re.sub(' +', ' ',clean_sentence)
     #replacing multiple spaces with a sing space
    words =clean_sentence.split()
    tokensList = [w for w in words]
    pos_tag = nltk.pos_tag(tokensList)
    nouns = set()
    for i in range(len(pos_tag)):
        if (pos_tag[i][1] == 'NN' or pos_tag[i][1] == 'NNP' or


 pos_tag[i][1] == 'NNS' or pos_tag[i][1] == 'NNPS' or pos_tag[i][1] == 'JJ'):
            nouns.add(lmtzr.lemmatize(pos_tag[i][0].lower(),

            get_wordnet_pos(pos_tag[i][1])))
    usefulNouns = [w for w in nouns if not w in stop]
     #save all non stop words in the variable
    return( " ".join(usefulNouns))
    return usefulNouns
from nltk import sent_tokenize
```

```
def preProcessDoc(raw_doc):
    sent_tokenize_list = sent_tokenize(inputDoc)
    lemmatized_sentences =[]
    for i in range(len(sent_tokenize_list)):
        lemmatized_sentences.append(preProcessSentence
        (sent_tokenize_list[i]))
    return(lemmatized_sentences)
preprocessedDoc=preProcessDoc(inputDoc)
print(preprocessedDoc)
length = len(preprocessedDoc)
print(length)
```



```
One day a rabbit was boasting about how fast he could run. He was laughing at the turtle for being so slow. Much to the rabbi
t's surprise, the turtle challenged him to a race. The rabbit thought this was a good joke and accepted the challenge. The fox
was to be the umpire of the race. As the race began, the rabbit raced way ahead of the turtle, just like everyone thought.

The rabbit got to the halfway point and could not see the turtle anywhere. He was hot and tired and decided to stop and take a
short nap. Even if the turtle passed him, he would be able to race to the finish line ahead of him. All this time the turtle ke
pt walking step by step by step. He never quit no matter how hot or tired he got. He just kept going.

However, the rabbit slept longer than he had thought and woke up. He could not see the turtle anywhere! He went at full speed t
o the finish line but found the turtle there waiting for him.
```

Figure 8.1: Document to be summarized



```
['one day rabbit boast fast could run', 'laugh turtle slow', 'much rabbit surprise turtle challenge race', 'rabbit think good j
oke accept challenge', 'fox umpire race', 'race begin rabbit race way ahead turtle like everyone thought', 'rabbit get halfway
point could see turtle anywhere', 'hot tired decide stop take short nap', 'even turtle pass would able race finish line ahead',
'time turtle keep walk step step step', 'never quit matter hot tire get', 'keep go', 'however rabbit sleep long think wake', 'c
ould see turtle anywhere', 'go full speed finish line find turtle wait']
```

Figure 8.2: Pre-processed text of the given document

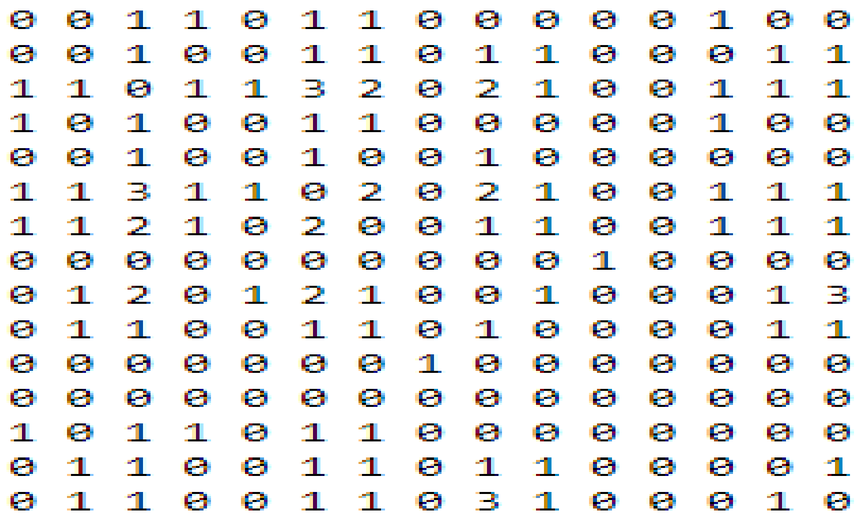
8.2.2 Sentence Scoring

The code below is used to find the similarity between two sentences(number of common words). Each sentence is considered as a node and the number of edges between

two nodes is the number of common words.
The total number of edges is stored in a symmetric matrix that represents the text being summarized. Then, we sum the values of the rows of the symmetric matrix to generate what we call a sum vector which is then used to rank sentences.

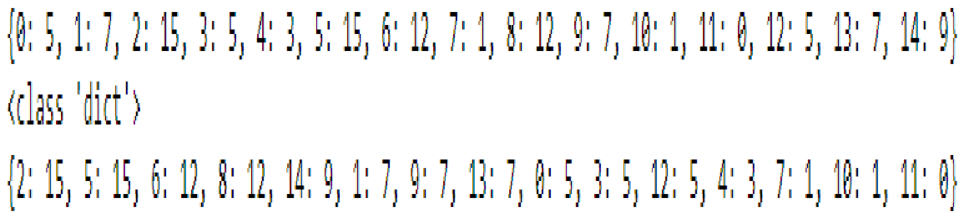
```
extract_nouns=[]
sent_tokenize_list = sent_tokenize(inputDoc)
sentenceNouns =[]
for i in range(len(sent_tokenize_list)):
    extract_nouns.append(getNouns(sent_tokenize_list[i]))
print(extract_nouns)
#From collections import OrderedDict
def duplicates(raw_sent):
    s = raw_sent
    return(' '.join(OrderedDict((w,w) for w in s.split()).keys()))
dup = []
for i in range(len(extract_nouns)):
    dup.append(duplicates(extract_nouns[i]))
print(dup)
a = [[0] * length for i in range(length)]
 #used to create a 2Darray with 0s of dim. length*length
for i in range(len(dup)):
    w = word_tokenize(dup[i])
    for j in range(len(dup)):
        count=0
        if i<j:
            v = word_tokenize(dup[j])
            for l in range(len(w)):
                for k in range(len(v)):
                    if w[l]==v[k]:
                        count=count+1
            a[i][j]=count
        elif i>j:
            a[i][j]=a[j][i]
        else:
            a[i][j]=0
for row in a:
    print(' '.join([str(elem) for elem in row]))
```

```
import numpy
sentenceScores=dict(enumerate(numpy.sum(a,axis=0)))
 #none implies sum of full array
print(sentenceScores)
print(type(sentenceScores))
sortedSentenceScores = dict(sorted(sentenceScores.items(),key=

lambda sentenceIDScore:sentenceIDScore[1], reverse=True))
print (sortedSentenceScores)
```



```
0  0  1  1  0  1  1  0  0  0  0  0  1  0  0
0  0  1  0  0  1  1  0  1  1  0  0  0  1  1
1  1  0  1  1  3  2  0  2  1  0  0  1  1  1
1  0  1  0  0  1  1  0  0  0  0  0  1  0  0
0  0  1  0  0  1  0  0  1  0  0  0  0  0  0
1  1  3  1  1  0  2  0  2  1  0  0  1  1  1
1  1  2  1  0  2  0  0  1  1  0  0  1  1  1
0  0  0  0  0  0  0  0  0  0  1  0  0  0  0
0  1  2  0  1  2  1  0  0  1  0  0  0  1  3
0  1  1  0  0  1  1  0  1  0  0  0  0  1  1
0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
1  0  1  1  0  1  1  0  0  0  0  0  0  0  0
0  1  1  0  0  1  1  0  1  1  0  0  0  0  1
0  1  1  0  0  1  1  0  3  1  0  0  0  1  0
```

Figure 8.3: Square matrix



```
{0: 5, 1: 7, 2: 15, 3: 5, 4: 3, 5: 15, 6: 12, 7: 1, 8: 12, 9: 7, 10: 1, 11: 0, 12: 5, 13: 7, 14: 9}
<class 'dict'>
{2: 15, 5: 15, 6: 12, 8: 12, 14: 9, 1: 7, 9: 7, 13: 7, 0: 5, 3: 5, 12: 5, 4: 3, 7: 1, 10: 1, 11: 0}
```

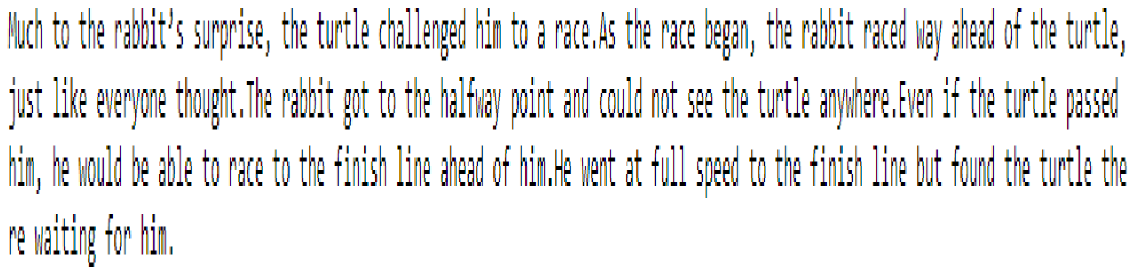
Figure 8.4: Sentence Scores

8.2.3 Normalization and Sentence Extraction

We use normalization to refer to rescaling an input variable to the range between 0 and 1. Normalization requires that you know the minimum and maximum values for each attribute. Here we use MinMax Normalization technique and extract top N sentences that satisfies the threshold.

```
topNSentences=10
sentenceScoreThreshold=0.5 #after normalization we take only between 0.5 to 1
from sklearn.preprocessing import MinMaxScaler
 #estimator scales tranlates transforms puts in range of 0-1
minMaxNormalizer = MinMaxScaler()
sentenceScoreArray=numpy.array(list(sortedSentenceScores.values())
,dtype=int).reshape(len(sortedSentenceScores), 1)
print(sentenceScoreArray.shape)
minMaxNormalizer.fit(sentenceScoreArray)
normalizedSentenceScoresNestedList=minMaxNormalizer.transform
(sentenceScoreArray).tolist()
normalizedSentenceScoresList = [score for scores in

normalizedSentenceScoresNestedList for score in scores]
print(minMaxNormalizer.transform(sentenceScoreArray).tolist())
normalizedSentenceScoresList=normalizedSentenceScoresList
[0:topNSentences]
print(normalizedSentenceScoresList) #print scores of topNsentences
normalizedSentenceScores=dict(zip(list(sortedSentenceScores.keys()),
normalizedSentenceScoresList))
#sortedsentenceScores
print(normalizedSentenceScores)
selectedSentenceScores = {sentenceID: normalizedSentenceScores[sentenceID]

for sentenceID in normalizedSentenceScores if normalizedSentenceScores[sentenceID]

>sentenceScoreThreshold}
print(selectedSentenceScores)
print(len(selectedSentenceScores))
parentFolder = 'c:\\Users\\Nammu\\Anaconda3'
inputFileName="output.txt"
inpFile = open(parentFolder+'/'+inputFileName,"w")
from nltk import sent_tokenize
def getSentencesInDoc(raw_doc):
    sent_tokenize_list = sent_tokenize(raw_doc)
    return(sent_tokenize_list)
```

```
originalDocSentences=getSentencesInDoc(inputDoc)
#Print Summary of input file
k=0
j=0
for k in range(len(originalDocSentences)):
    for sentenceID in selectedSentenceScores.keys():
         if(k == sentenceID):
                inpFile.writelines(originalDocSentences[sentenceID])

inpFile = open(parentFolder+'/'+inputFileName,"r")
outputDoc = inpFile.read()
print(outputDoc)
```



```
Much to the rabbit's surprise, the turtle challenged him to a race.As the race began, the rabbit raced way ahead of the turtle,
just like everyone thought.The rabbit got to the halfway point and could not see the turtle anywhere.Even if the turtle passed
him, he would be able to race to the finish line ahead of him.He went at full speed to the finish line but found the turtle the
re waiting for him.
```

Figure 8.5: Summary of the Document

Chapter 9

TESTING,RESULTS AND PERFORMANCE ANALYSIS

9.1 Introduction

Testing is an important phase in the development life cycle of the product. This is the phase where the error remaining from all the phases was detected. Hence testing performs a very critical role for quality assurance and ensuring the reliability of the software. Once the implementation is done, a test plan should be developed and run on a given set of test data. Each test has a different purpose, all work to verify that all the system elements have been properly integrated and perform allocated functions. The testing process is actually carried out to make sure that the product exactly does the same thing what is suppose to do. Testing is the final verification and validation activity within the organization itself. In the testing stage following goals are tried to achieve:-

- To affirm the quality of the project.
- To find and eliminate any residual errors from previous stages.
- To validate the software as the solution to the original problem.
- To provide operational reliability of the system.

During testing the major activities are concentrated on the examination and modification of the source code. The test cases executed for this project are listed below. Description of the test case, steps to be followed; expected result, status and screenshots are explained with each of the test cases.

9.2 Testing Methodologies

There are many different types of testing methods or techniques used as part of the software testing methodology. Some of the important types of testing are:

9.2.1 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level. Using white box testing we can derive test cases that:

- Guarantee that all independent paths within a module have been exercised atleast once.
- Exercise all logical decisions on their true and false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Execute internal data structure to assure their validity.

9.2.2 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings,structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot see into it. The test provides inputs and responds to outputs without considering how the software works. It uncovers a different class of errors in the following categories:

- Incorrect or missing function.
- Interface errors.
- Performance errors.
- Initialization and termination errors.
- Errors in objects.

Advantages:

- The test is unbiased as the designer and the tester are independent of each other.
- The tester does not need knowledge of any specific programming languages.

- The test is done from the point of view of the user, not the designer.
- Test cases can be designed as soon as the specifi

  cations are complete.

9.2.3 Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases. Test strategy and approach Field testing will be performed manually and functional tests will be written in detail. **Test objectives**:

- All Components must work properly.
- Proper coordinates should be sent by the Android app to the
- The entry screen, messages and responses must not be delayed in the Android app.

9.3 Result and Performance Analysis

ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation.[14][15] It is essentially of a set of metrics for evaluating automatic summarization of texts as well as machine translation. It works by comparing an automatically produced summary or translation against a set of reference summaries. To evaluate a summarization system you have two types of summaries. One is the system generated summaries that is referred to as peer summaries and then you have the reference summaries or gold standard summaries known as model summaries.
System Summary (what the machine produced):
Alice was found playing football in the ground.

Reference Summary (gold standard usually by humans):
Alice was playing football in the ground.

If we consider just the individual words, the number of overlapping words between the system summary and reference summary is 7. This however, does not tell you much as a metric. To get a good quantitative value, we can actually compute the precision and recall using the overlap.

Recall in the context of ROUGE means how much of the reference summary is the system summary recovering or capturing? If we are just considering the individual words, it can be computed as:

$$Recall = \frac{no\_of\_overlapping\_words}{total\_words\_in\_reference\_summary} \tag{9.1}$$

In this example, the Recall would thus be:

$$Recall = \frac{7}{7} = 1$$

This means that all the words in the reference summary has been captured by the system summary.

A machine generated summary (system summary) can be extremely long, capturing all words in the reference summary. But, much of the words in the system summary may be useless, making the summary unnecessarily verbose. This is where precision comes into play. In terms of precision, what you are essentially measuring is, how much of the system summary was in fact relevant or needed? Precision is measured as:

$$Precision = \frac{no\_of\_overlapping\_words}{total\_words\_in\_system\_summary} \tag{9.2}$$

In this example, the Precision would thus be:

$$Precision = \frac{7}{8} = 0.875$$

This simply means that 7 out of the 8 words in the system summary were in fact relevant or needed.

The precision aspect becomes really crucial when you are trying to generate summaries that are concise in nature. Therefore, it is always best to compute both the Precision and Recall and then report the F-Measure.

$$F\_measure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{9.3}$$

Code for calculating Recall, Precision and F_measure:

```
def recall(a, b):
    c = a.intersection(b)
    return float(len(c)) / (len(b))
def precision(a, b):
    c = a.intersection(b)
    return float(len(c)) / (len(a))
    #return float(len(c)) / (len(a) + len(b) - len(c))
parentFolder = 'c:\\Users\\Nammu\\Anaconda3'
inputFileName="pracnltk_summarizer.txt"
inpFile = open(parentFolder+'/'+inputFileName,"r")
inputDoc = inpFile.read()
```

```
system_summary_tokenize_list1 = word_tokenize(outputDoc)
meaningful_words_list1=[w for w in system_summary_tokenize_list1 if not w in stop]

#print(system_summary_tokenize_list1)
reference_summary_tokenize_list2 = word_tokenize(inputDoc)
meaningful_words_list2=[w for w in reference_summary_tokenize_list2

if not w in stop]
# The intersection is ['dog', 'cat']
# union is ['dog', 'cat', 'rat', 'mouse]
words1 = set(meaningful_words_list1)
words2 = set(meaningful_words_list2)
recall_value=recall(words1, words2)
precision_value=precision(words1,words2)
f_measure=(2*recall_value*precision_value)/(recall_value+precision_value)
print(recall_value)
print(precision_value)
print(f_measure)
```

9.3.1 Examples

1. **Original Document**- *Gallery unveils interactive tree A Christmas tree that can receive text messages has been unveiled at London's Tate Britain art gallery. The spruce has an antenna which can receive Bluetooth texts sent by visitors to the Tate. The messages will be "unwrapped" by sculptor Richard Wentworth, who is responsible for decorating the tree with broken plates and light bulbs. It is the 17th year that the gallery has invited an artist to dress their Christmas tree. Artists who have decorated the Tate tree in previous years include Tracey Emin in 2002. The plain green Norway spruce is displayed in the gallery's foyer. Its light bulb adornments are dimmed, ordinary domestic ones joined together with string. The plates decorating the branches will be auctioned off for the children's charity ArtWorks. Wentworth worked as an assistant to sculptor Henry Moore in the late 1960s. His reputation as a sculptor grew in the 1980s, while he has been one of the most influential teachers during the last two decades. Wentworth is also known for his photography of mundane, everyday subjects such as a cigarette packet jammed under the wonky leg of a table.*

   - **Summary**- *Gallery unveils interactive tree A Christmas tree that can receive text messages has been unveiled at London's Tate Britain art gallery.The* *messages will be "unwrapped" by sculptor Richard Wentworth, who is responsible for decorating the tree with broken plates and light bulbs.It is the 17th year that the gallery has invited an artist to dress their Christmas tree.Artists who have decorated the Tate tree in previous years include Tracey Emin in 2002.*

- **Recall** = 0.6
- **Precision** = 0.6818181818181818
- **F-Measure** = 0.6382978723404256

2. **Original Document** - *Jarre joins fairytale celebration French musician Jean-Michel Jarre is to perform at a concert in Copenhagen to mark the bicentennial of the birth of writer Hans Christian Andersen. Denmark is holding a three-day celebration of the life of the fairy-tale author, with a concert at Parken stadium on 2 April. Other stars are expected to join the line-up in the coming months, and the Danish royal family will attend. "Christian Andersen's fairy tales are timeless and universal," said Jarre. "For all of us, at any age there is always - beyond the pure enjoyment of the tale - a message to learn." There are year-long celebrations planned across the world to celebrate Andersen and his work, which includes The Emperor's New Clothes and The Little Mermaid. Denmark's Crown Prince Frederik and Crown Princess Mary visited New York on Monday to help promote the festivities. The pair were at a Manhattan library to honour US literary critic Harold Bloom "the international icon we thought we knew so well". "Bloom recognizes the darker aspects of Andersen's authorship," Prince Frederik said. Bloom is to be formally presented with the Hans Christian Andersen Award this spring in Anderson's hometown of Odense. The royal couple also visited the Hans Christian Anderson School complex, where Queen Mary read The Ugly Duckling to the young audience. Later at a gala dinner, Danish supermodel Helena Christensen was named a Hans Christian Andersen ambassador. Other ambassadors include actors Harvey Keitel and Sir Roger Moore, athlete Cathy Freeman and Brazilian soccer legend Pele.*

   - **Summary** - *Jarre joins fairytale celebration French musician Jean-Michel Jarre is to perform at a concert in Copenhagen to mark the bicentennial of the birth of writer Hans Christian Andersen."Christian Andersen's fairy tales are timeless and universal," said Jarre."Bloom recognizes the darker aspects of Andersen's authorship," Prince Frederik said.Bloom is to be formally presented with the Hans Christian Andersen Award this spring in Anderson's hometown of Odense.The royal couple also visited the Hans Christian Anderson School complex, where Queen Mary read The Ugly Duckling* *to the young audience.Later at a gala dinner, Danish supermodel Helena Christensen was named a Hans Christian Andersen ambassador.*

- **Recall** = 0.933
- **Precision** = 0.875
- **F-Measure** = 0.9032258064516129

Chapter 10

CONCLUSION AND FUTURE SCOPE

The rate of information growth due to the World Wide Web has called for a need to develop efficient and accurate summarization systems. Although research on summarization started about 50 years ago, there is still a long trail to walk in this field. Over time, attention has drifted from summarizing scientific articles to news articles, electronic mail messages, advertisements, and blogs. Both abstractive and extractive approaches have been attempted, depending on the application at hand. Usually, abstractive summarization requires heavy machinery for language generation and is difficult to replicate or extend to broader domains. In contrast, simple extraction of sentences have produced satisfactory results in large-scale applications, specially in multi-document summarization. The recent popularity of effective newswire summarization systems confirms this claim.

In this paper, we have described a general overview of automatic text summarization. The status, and state, of automatic summarising has radically changed through the years. It has specially benefit from work of other asks, e.g. information retrieval, information extraction or text categorization. Research on this field will continue due to the fact that text summarization task has not been finished yet and there is still much effort to do, to investigate and to improve. Definition, types, different approaches and evaluation methods have been exposed as well as summarization systems features and techniques already developed. In the future we plan to contribute to improve this field by means of improving the quality of summaries, and studying the influence of other neighbour tasks techniques on summarization.

The proposed method is simple to implement and does not rely on the calculations of the cosine similarity between sentences to rank the them in the summary. Such a process is a complex and not exact due to semantic assumptions. When comparing our method with the output results of the other real time summarizers on 1,000 text samples, we found that our results are better or comparable to those online summarizers. It is expected that our results will improve if an efficient preprocessing routine is used prior to the implementation of the proposed method. Such practice is common in Text summarization. Finally, the proposed method can be used for real time applications.

We mention some of the possible future extensions of this research. In this thesis, we focused on summarization of news articles belonging to sports and technical domain. The techniques proposed here are adaptable across other domains.

One of the future plans may be to apply the topic-focused summarization framework to news articles or blogs and to extend the work in the machine leaning approaches. Topic- focused summaries of news articles would be lot more accurate and valuable to users. It would be more interesting to work on topic modeling and summarization in the domain of social media in future.

The rate at which the information is growing is tremendous. Hence it is very important to build a multilingual summarization system and this research could be a stepping stone towards achieving that goal provided there is availability of online lexical databases in other languages. The work presented by the thesis can also be applicable to multi document summarization by using minimal extensions.

The thesis has used evaluation metrics Precision, Recall and F-measure to measure performance gain over existing systems with ROUGE tool. In future work, new metrics can be investigated which can be used in automatic evaluation environment to measure the overall quality such as grammar, readability, prominence and relativeness.

The state of the art summarization systems are all extractive in nature, but the community is gradually progressing towards abstractive summarization. Although a complete abstractive summarization would require deeper natural language understanding and processing, a hybrid or shallow abstractive summarization can be achieved through sentence compression and textual entailment techniques. Textual entailment helps in detecting shorter versions of text that entail with same meaning as original text. With textual entailment we can produce more concise and shorter summaries.

The Implemented system in this thesis can work as framework for the research community to understand and extend the applicability of cognitive and symbolic approach in various domains of business needs.

Research in summarization continues to enhance the diversity and information richness, and strive to produce coherent and focused answers to users information need.

References

[1] Saif alZahir,Qandeel Fatima and Martin Cenek:New Graph-Based Text Summarization Method

[2] Saiyed Saziyabegum and Priti S. Sajja, PhD:Review on Extractive Text Summarization Approaches

[3] Dipanjan Das Andre F.T. Martins:A Survey on Automatic Text Summarization

[4] Gunes Erkan and Dragomir R. Radev:LexRank: Graph-based Lexical Centrality as Salience in Text Summarization, Journal of Artificial Intelligence Research 22, pp. 457-479, 2004.

[5] Shanmugasundaram Hariharan and Rengaramanujam Srinivasan:Studies on Graph based Approaches for Single and Multi Document Summarizations, International Journal of Computer Theory and Engineering, Vol. 1, No. 5, Dec, 2009.

[6] Shanmugasundaram Hariharan, Thirunavukarasu Ramkumar and Rengaramanujam Srinivasan:Enhanced Graph Based Approach for Multi-Document Summarization, the International Arab Journal of Information Technology, Vol. 10, No. 4, July 2013.

[7] Dipanjan Das and Andre F.T. Martins :A Survey on Automatic Text Summarization, 21st November 2007.

[8] Vishal Gupta and Gurpreet Singh Lehal:A Survey of Text Summarization Extractive Techniques, Journal of Emerging Technologies in Web Intelligence, Vol. 2, No. 3, August 2010.

[9] Elena Lloret:Text Summarization: An Overview, TEXT-MESS (TIN 2006-15265-C06-01), 2006.

[10] Claudia Sofia Oliveira Santos :, ALEXIA Acquisition of Lexical Chains for Text Summarization, February 2006.

[11] Maheedhar Kolla:Automatic Text Summarization Using Lexical Chains: Algorithms and Experiments, 2004.

[12] About nltk https://textminingonline.com/dive-into-nltk-part-i-getting-started-w

[13] http://www.nltk.org/

[14] About ROUGE http://kavita-ganesan.com/rouge-howto/#.Wx4kLvmFOM8

[15] http://rxnlp.com/how-rouge-works-for-evaluation-of-summarization-tasks/#.Wx4kkfmFOM9