Document Summarization And Mind Map Generation

ChiragB.Rajpal[1], Abhijeet B. Ingale[2], Aishwarya D. Kore[3], Akriti A. Singh[4]
[1, 2, 3, 4] Dept of Information Technology,
[1, 2, 3, 4] Sinhgad College of Engineering, Pune, India
Savitribai Phule Pune University

Abstract- *In this era of internet, as huge amount of data is available today, we need to understand the information in a faster and in a much efficient way, so that we don't miss out important information. On understanding this need, we thought of a need for automatization, to retrieve and understand the information in shorter period of time. Text Summarization helps achieve this goal by generating summary of the text (which consists of important information) with meaningful statements and removing redundancy. The aim of this work is to study different algorithms for text summarization analyse, design and also implement the summarization techniques using abstractive and extractive based techniques, to generate concise and meaningful summaries of complex articles and documents. Moreover, generating the graphical representation of the text called, the mind map to help the reader get a better understanding of the document.*

Keywords- Text Summarization, Mind-map, Semantic Analysis

I. INTRODUCTION

Humans are surrounded with lot of information available from sources like newspapers, records, websites, books, etc. They need to understand all information in faster and in an efficient way for better utilization of that information. Therefore, there arises a need for automation tool.

The software tool, that we are proposing consists of two important modules i.e. Document Summarization and Mind Map Generation. The first, gives us summary which consists of important sentences of the document. These sentences when combined together generate complete meaning of the document but in a less amount of words. The summary of the document is generated by extracting some of the features of the text like sentence length, position, frequency etc. Sentences are also processed for finding out the similarities between them and thus to avoid redundancy. Humans have a tendency to understand graphical contents easier than in text format. After the generation of summarized document, the latter module will generate a graphical Mind Map of the summary. Mind Map consists of cluster of nodes representing the keywords which are mapped in way that gives the end user a complete understanding of the document.

II. ARCHITECTURE

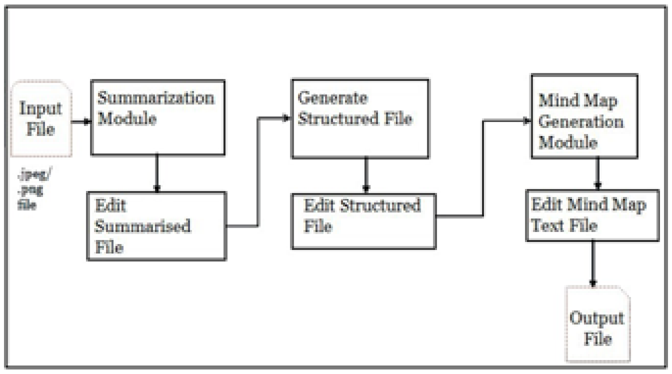


Figure1: Overview of System Architecture

A. Architecture of Document Summarization module

It converts to a summarized document from the series of following steps.

1. First, the document is pre-processed to eliminate the stop words present in it.
2. Secondly, frequency of each word in the entire document is calculated.
3. Third, we calculate the sentence score of each sentence based of the following factors:

a. **Cue Phrase Score**- refers to the score of each sentence depending on occurrence of cue phrases (important phrases taken as input from the user)
b. **Title Resemblance** - the occurrence of similar words present in the title.
c. **Sentence Length**-How long is the sentence compared to other sentences present in the document.
d. **Sentence Position**-The sentences occurring at the beginning or end are considered important and likewise.

e. **Sentence Occurrence**-Occurrence of similar sentences are removed by taking reference of cosine similarity algorithm.
f. **Frequency score**-summation of frequency of each word in the sentence becomes its frequency score.

4. And finally sentences with more sentence score are ranked and the top sentences are displayed in the summary after asking the user the amount of compression required. Optionally, user can ask for mind map which can be generated after processing the summarized documents and generating a structured file.

The user provides the text file as an input. The input is then sent through tokenize module, where the file is tokenized into sentences for the calculation of number of sentences. Then the stop-words are removed from the input file and is processed for the rank calculation of the individual sentences.

Calculation of Rank

After all the calculation the scores are stored in a matrix of all sentences. Then the cumulative score of all the sentences is calculated by adding the scores of various features with respect to that sentence. Finally, the ranking of sentences is performed based on the cumulative scores, the sentence with highest score is allocated highest rank and so on.The user is inquired about the required number of sentences in the summary and accordingly the sentences are selected and returned to the user based on their ranks.

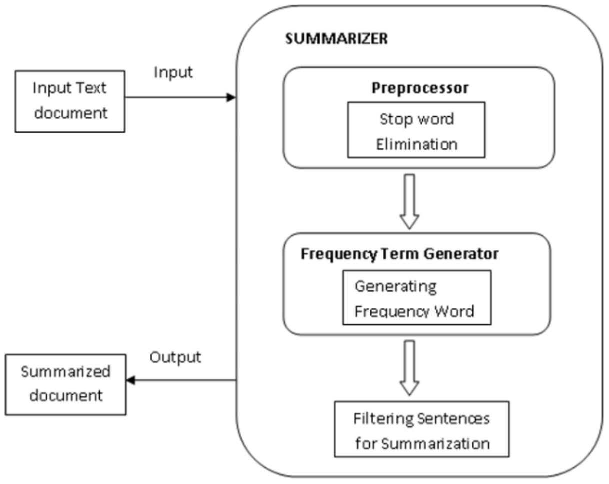


Figure 2: Architecture of Frequency Calculation

Figure 2.Describes how the text document or handwritten text document gets converted to a summarized document from the series of following steps. Firstly, the pre-processing of the document happens which eliminates the stop words present in it. Secondly, frequency of each word is calculated. And finally sentences with more frequency are included in the summarized document. Optionally, user can ask for mind map which can be generated from the summarized documents.

B. **Architecture of Mind Map Generation module**

The following Figure 3 is system description to illustrate all the steps of the Mind Map Generation system.
It consists of 6 main modules:

- Frontend
- Preprocessing
- Extractor Algorithm
- Word Tab-Space Representation
- Text to Mind Map
- Mind Map Graphical Representation

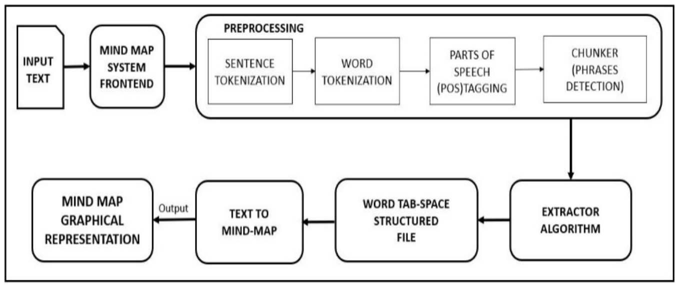


Figure 3: Architecture of Mind Map Generation

The role of each module is as follows:

1. **Mind Map System Frontend:** It is the first phase in the system which is responsible for the input text.
2. **Preprocessing:** It contains 4 sub steps:
   - **Sentence tokenization:** Creates array or list of sentences from input text paragraph.
   - **Word tokenization:** Creates array of words from each tokenized sentence.
   - **Parts of Speech tagging:** Each word from word array is tagged according to the English grammar like noun, pronoun, verb etc. For eg, word ‘car’ is tagged as Noun.
   - **Chunker:** It breaks the sentence into groups (of words) containing sequential words of sentence, that belong to a noun group, verb group, etc.
   - Phrases are identified like noun phrases, verb phrases, adverbial phrases, etc.
3. **Extractor Algorithm:** It takes all preprocessing steps outputs as input and generates a Word tab -space structured file. Basically, algorithm identifies the subject and predicate of sentence. If subject of sentence matches with title of mind map then the predicate is included

instructed file. It identifies which word or phrases is related to mind map and the level in which it will be displayed in mind map.

4. **Word Tab-Space Representation:** The output of Extractor is a structured file where each node i.e. words that will be in mind map represented in tab- spaced format. It describes hierarchical structure of mind map represented in textually.

For example,

Gandhiji
Born
in Gujrat.

The above example shows Word Tab Space representation, 'Gandhiji' is root node and its child node is 'born' which has one tab implying it is second level of mind map and 'in Gujrat' has two tabs implying it is third level of mind map.

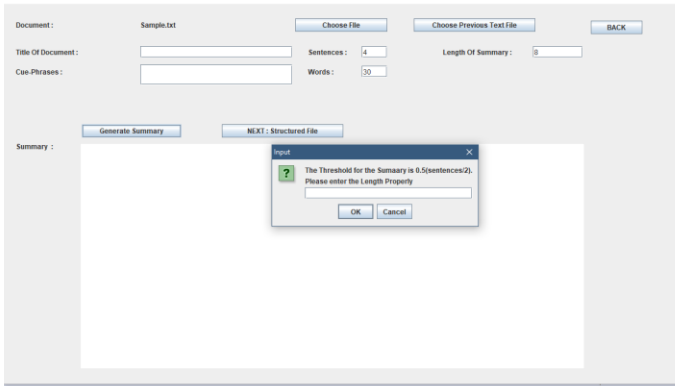
Also, this structured file/text is editable and user can add new words or shift words to any level of mind map.

5. **Text to Mind Map:** The structured file is inputted to this module. This module creates graph consisting of nodes and edges. Nodes are the words from structured file and edges represents the links between words which are present on adjacent levels. Repeated words are removed and mind map is drawn with no redundancy of information.
6. **Mind Map Graphical Representation:** This module displays the generated mind map in graphical format. Each level containing nodes has its own color. It generates jpeg format image which can be saved for future use.

III. IMPLEMENTATION

For Text Summarization module, Java in-build library functions were used for tokenization, design and classes for each step of summarization process were implemented. The threshold values were decided using trial and error method such that performance of summarization is maximized.

For identifying different grammar semantics and for part of speech tagging we used OpenNLP, a Java based open source library. Extractor class is implemented for generating intermediate structured file (tab-newline format) as mentioned above.This class contains algorithm which identifies nodes and linking and then generate editable structured format.



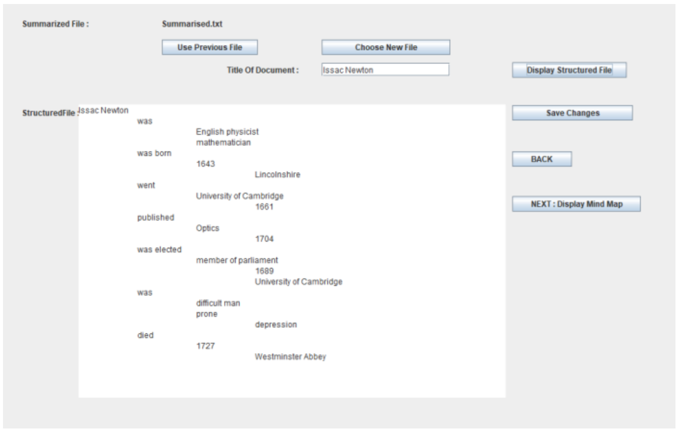
Document : Sample.txt
Title Of Document :
Sentences : 4
Length Of Summary : 8
Cue-Phrases :
Words : 30

The Threshold for the Sumaary is 0.5(sentences/2).
Please enter the Length Properly

Figure 4: Text Summarization GUI

A user-friendly GUI as shown in figure 4, is developed in Java, provides user with variety of input options like browse text file or copy paste text directly in text box. One can mention important cue phrases , number of words or sentences according to their requirement.

As shown in Figure 5, Structured File representation is generated which will be used for mind map generation. This representation is editable. New information can be added or errors can be removed using this feature which can generate correct mind map.



Summarized File : Summarised.txt
Title Of Document : Issac Newton

StructuredFile :
```
Issac Newton
	was
		English physicist
		mathematician
	was born
		1643
			Lincolnshire
	went
		University of Cambridge
			1661
	published
		Optics
			1704
	was elected
		member of parliament
			1689
			University of Cambridge
	was
		difficult man
		prone
			depression
	died
		1727
			Westminster Abbey
```

Figure 5: Structured File Representation and Editing GUI

For drawing nodes and edges,a python library named Graphwiz is used. It involves process of generating adjacency matrix from structured format. Each level of mind map has different color nodes. Redundancy is removed by drawing similar nodes once. If user is not satisfied with results, he can always go back to structured file representation editing window and do changes according.

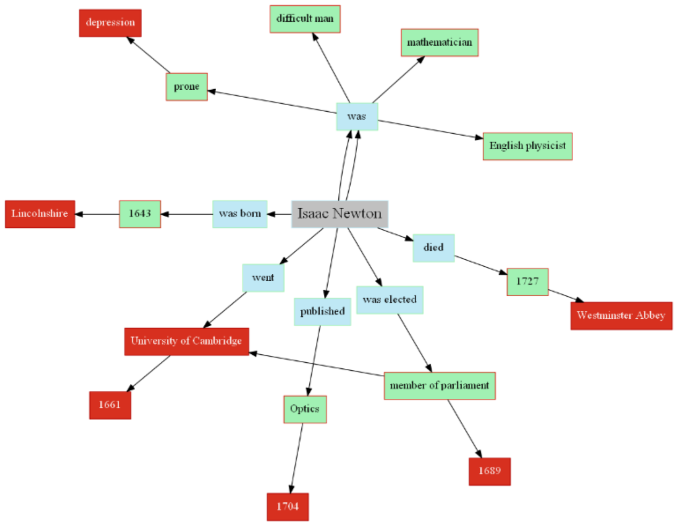


Figure 6: Mind MapOutputWindow displaying mind map for Issac Newton Information

IV. RESULTS

To estimate the capabilities of system we provided system with same sample paragraph which was used and mentioned in reference paper [1] Fig 3. Various measures were considered for comparison of results amongst different summarization methods.

COMPARISON TABLE(with reference to[1]):

Table 1: Comparison table of precision, recall, and f-measure

Evaluation	Precision	Recall	F-measure
Heading wise Summarizer	0.651	0.640	0.645
MS-Word summarizer	0.669	0.459	0.544
Auto-summarizer	0.481	0.151	0.230
Free-summarizer	0.514	0.331	0.403
Our summarizer	0.435	0.666	0.526

Following measures were used for comparison mentioned in table 1.

Precision:

Number of relevant sentences retrieved / Total Number of sentences retrieved:

10/23= 0.4347

Recall:

Number of relevant sentences retrieved/Total Number of Relevant sentences:

10/15 = 0.666

F-measure = $\frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$ = 0.526

A precision of 0.4347 and a recall of 0.666 with time complexity of 5 secs of each module is achieved. This system is capable of summarizing the document up to 50% of its actual length(meaningfully).

V. CONCLUSION AND FUTURE WORK

This tool can save lot of time and cost for any individual who needs to understand long articles, and books quickly. Our system analyzes the summarized document and uses the existing nodes for mind map rather than creating new ones i.e. no redundancy is tolerated in mind maps too. User friendly interface and editing feature which enables user to change or add information, if they are not satisfied with results which makes this system more reliable and usable.

In future scope, we can use other semantic methods for finding relationships between nodes in mind map to further increase the reliability and efficiency. The nodes can be represented using pictures searched on internet. Multi-level mind maps are possible if we know the linking between two mind maps.

REFERENCES

[1] Krishnaveni, P., &Balasundaram, S. R. (2017). Automatic text summarization by local scoring and ranking for improving coherence. 2017 International Conference on Computing Methodologies and Communication (ICCMC).
[2] Moratanch, N., and S. Chitrakala. "A survey on extractive text summarization." Computer, Communication and Signal Processing (ICCCSP), 2017 International Conference on. IEEE, 2017.
[3] Abdeen, M., et al. "Direct automatic generation of mind maps from text with M 2 Gen." Science and Technology for Humanity (TIC-STH), 2009 IEEE Toronto International Conference. IEEE, 2009.
[4] Jo, Taeho. "K nearest neighbor for text summarization using feature similarity." 2017 International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE). IEEE, 2017.

[5] Andhale, Narendra, and L. A. Bewoor. "An overview of text summarization techniques." 2016 international conference on computing communication control and automation (ICCUBEA). IEEE, 2016.