International Journal of Computer Applications Technology and Research
Volume 9–Issue 08, 240-245, 2020, ISSN:-2319–8656

# Document Summarization using Graph Based Methodology

Aditya Jeswani
Student
Dwarkadas J. Sanghvi College of Engineering
Mumbai 400056, India

Shruti More
Student
Dwarkadas J. Sanghvi College of Engineering
Mumbai 400056, India

Kabir Kapoor
Student
Dwarkadas J. Sanghvi College of Engineering
Mumbai 400056, India

Sifat Sheikh
Student
Dwarkadas J. Sanghvi College of Engineering
Mumbai 400056, India

Ramchandra Mangrulkar
Associate Professor
Dwarkadas J. Sanghvi College of Engineering
Mumbai 400056, India

**Abstract**: This paper works towards constructing a short summary of documents with the help of natural language processing techniques. The authors goal is to identify the important aspects of a large piece of textual information, extract it and present it in a concise manner such that it conveys the information in a more efficiently and precisely. The proposed approach will generate a simple summarization of one or more documents which will help the readers to understand what the documents offer to them and identify their context without reading through them entirely. The existing methods for this work focus on different aspects of the text involved but the efficiency of these methods largely varies. The proposed methodology makes use of a combination of multiple aspects of text instead of a single aspect in order to improve the efficiency of summarization systems. The authors present a qualitative and quantitative analysis of their system as compared to the existing base-lines and demonstrate our system for a relevant application like news snippet generation.

**Keywords**: Extractive summarization, Multi-document summarization, Key phrase extraction, Shortest path algorithm, Textrank algorithm, GloVe embeddings, Cosine similarity.

## 1. INTRODUCTION

With the advent of the Internet, there has been a voluminous increase in the amount of data available for humans to read and understand. Going through all of the data is a mammoth task and often required more effort than the value provided by the goal being achieved. Thus, there is a need for a system which can concisely convey all the information that is available in different sources.

Document summarization is one of the most widely researched fields of Natural Language Processing. The task involves using a single or multiple document(s) belonging to a particular domain, understanding the contents of the document and then generating a paragraph which conveys the information in a concise and human readable format. The task of summarization can be carried out in two different ways – extractive and abstractive.

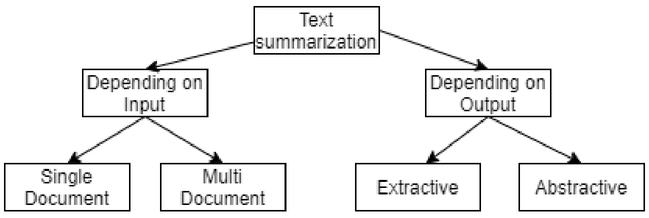


Figure 1. Text Summarization Overview

In extractive document summarization, generation of the summary uses sentences which are present in the document set provided. In this mechanism, the different sentences in the documents are analyzed for their relevance to the main idea of the document cluster, assigned a score and a rank. On the basis of the rank, the top sentences are extracted according to the length required and are then presented to the user as a summary.

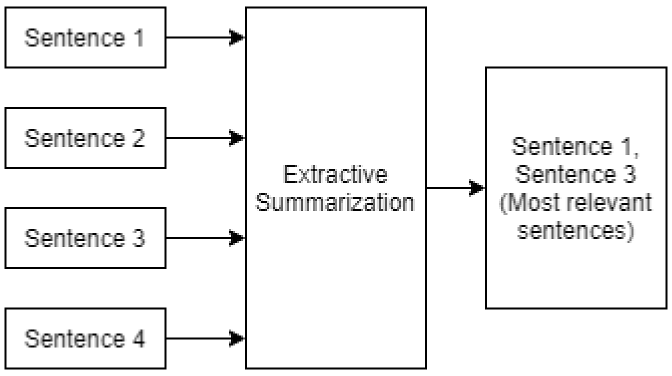


Figure 2. Extractive summarization

On the other hand, an abstractive summarizer works differently. While the initial stages are similar where the document contents are analyzed in order to identify the main idea, this summarizer does not directly pick up sentences from the document. Instead, the model uses the information and knowledge gained in order to generate new sentences on its own to create the summary. The abstractive summarizers more closely resemble how humans generate summaries, by understanding the meaning being conveyed rather than simply picking up sentences from the given documents.

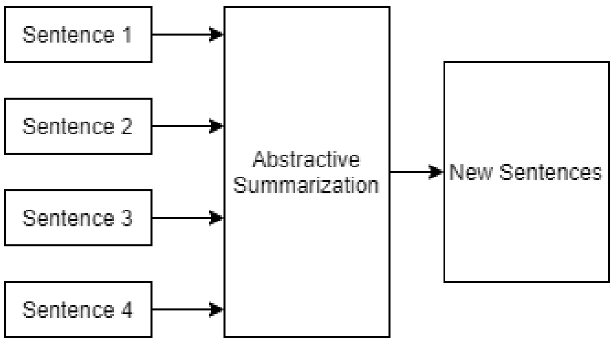


Figure 3. Abstractive Summarization

Graph-based summarization methods have yielded strong and promising results. In these approaches, sentences are treated as nodes and the relationships between them are represented by edge weights. The significance score of each sentence in a document is assumed to be related to other sentences. An edge (or link) with a corresponding weight is created if there is a relation between two sentences, while the weight between sentences in a document is used to provide a score for each sentence. In this paper, the authors propose a technique which combines the uses of sentence relations and the importance of keyphrases to carry out the task of extractive summarization.

## 2. LITERATURE REVIEW

Akash Ajampura Natesh et al [1] explores an approach different from traditional graph summarization techniques. Since nouns form an important part of the sentence, they create a graph of the available phrases where the nouns form the nodes in the graph. Pronouns in the sentences are assigned name or object references by analyzing the preceding sentence. An edge is added between the 2 nodes if the nouns occur together in the sentence, with the edge weight being the distance between the nouns. Sentence scores are assigned on the basis of noun scores for nouns in the sentence and the top sentences are selected to form the summary for the document. A special feature of their methodology is the use of pronoun resolution to ensure noun occurrences in the sentences, ensuring that a valid score for the sentence can be generated.

Shikhar Sharma et al [2] adopts a different technique for summarizing documents. After breaking the document into individual sentences, these sentences are used to form the graph. A distortion measure based on the “Squared Difference” technique is used to calculate the semantic dissimilarity between the sentences. The dissimilarity is then subtracted from 1 in order to obtain the semantic similarity between the sentences. These values are used to initialize the graph weights in order to avoid random initialization. Once the graph is created, it is passed onto the Textrank algorithm to obtain sentence scores.

Chirantana Malik et al. [3] implements a graph-based approach with a modified Textrank algorithm. Each sentence corresponds to a node in the graph. Modified Cosine similarity is used to give weights to edges which takes into account different levels of importance of words in each sentence. Textrank score is calculated for each node of the graph by considering the average weight of the edges incident to it for giving importance to the weights associated with the edges. Summarization is done here by selecting top ‘n’ number of sentences based on their Textrank value and then arranging them by their index.

Kang Yang et al. [4] proposes a methodology based on an integrated graph model used along with Textrank. POS tagging is performed for each word, forming word-POS pairs. For context analysis, bigrams and trigrams are constructed. Thus, three separate structures are created- word-POS, bigram and trigram. Then three undirected weighted graphs are built for the sentences in a document which correspond to the three structures constructed earlier. Graphs from different sources are integrated in a Naive Bayesian fashion. Textrank is performed to calculate score of each node or sentence. Sentences with the highest score are selected as per the compression rate.

Hakim et al [5] presents new ways to expand and try to improve graph-based multi-document extractive summarization models by exploring how key phrases can be used in the process of text summarization to produce better summaries. The intuition behind this approach is that the key phrases of a document cluster represent the core ideas and topics of the cluster. Therefore, by taking into account those key phrases, and more specifically, by considering the similarity between those key phrases and the various sentences in the cluster, it can evaluate better that which sentences are the most important.

Erkan et al [6] propose a multi-document extractive summarization system which serves as the baseline model. The centrality score is computed using the LexRank method, then this score is modified to include a different key phrase score that represents the sentence’s similarity to the key phrases in the document cluster. The key phrase score is computed using 3 approaches. First using only the key phrases that is equal to the number of phrases present in the given sentence, second using the key phrases equal to the sum of the cosine similarity between the TF-IDF representations of each key phrase (Feinerer [7] et al.); and third, by calculating the key phrase’s importance using the scores/ranking provided by the pke package for each key phrase [7]. After computing the final modified score, the author has used ROUGE metric for evaluation.

Jonas et al [8] provides us with a method that results in a smooth summary. Most of the graph-based summarization techniques suffers from sudden topic shifts. This problem could be solved by using Shortest Path Algorithm suggest by the author in the paper ‘Extraction based summarization using a shortest path algorithm’. The method first divides the entire documents such that sentences form the nodes of the graph. Costs of edges between nodes are based on number of overlapping words between two sentences, more similar the sentence implies less cost. A special feature of this method is that a node has an edge to its following sentence too, so this might result in smooth summary. For constructing summary, chose a path from the first sentence to last sentence and include all sentences in the path. This method results in smooth summary and summaries of varying length. After computing the model, the author has used ROUGE metric for evaluation.

Madhurima et al [9] suggest a different approach for graph-based summarization. Instead of using Textrank algorithm, the authors use clustering technique. After POS tagging, pronoun resolution and stop word removal, each sentence acts as the node of the graph. Cosine similarity is used to assign weight to the edges between two nodes. After graph construction, clustering coefficient and average clustering coefficient is computed for each node. The special feature of this methodology is applying info map clustering algorithm to partitioning graph into subgraphs and selecting subgraphs having coefficient greater than the average clustering coefficient.

Flourian Boudin et al [10] describes pke, an open source python-based keyphrase extraction toolkit. It provides an end-to-end keyphrase extraction pipeline in which each component can be easily modified or extended to develop new approaches.

## 3. PROPOSED METHODOLOGY

The approach combines two aspects of making document summaries useful – keyword extraction [5] and graph representation of document contents – to build a Multi-Document extractive summarization model. While both graph summarization and keyword extraction have been implemented in the past, a model which combines the power of both the techniques has not been examined in detail. The complete methodology can be broken down into three stages:

#### *3.1.1 Stage 1: Pre-processing*

In the first stage, the model reads the contents of all (or the single) documents(s) which are given by the user, such that the documents belong to the same domain. Once all the input documents are read, the contents of the documents are split from the paragraphs into the corresponding individual sentences. Once the application obtains a list of sentences, they are pre-processed to remove stopwords and resolve pronouns in consecutive sentences.

#### *3.1.2 Stage 2: Graph Builder*

Keywords are extracted using an open-source tool, pke. Shortest path algorithm is generated to ensure summary generated is smooth and it also reduces the corpus size. The graph constructed consists of sentences given as output from the shortest path algorithm as the nodes and edge weights as the sum of sentence-sentence similarity and sentence-keyword similarity.

#### *3.1.3 Stage 3: Summary Generation*

The weighted graph is passed to the Textrank module to get sentence importance scores and top sentences are extracted to form the summary.

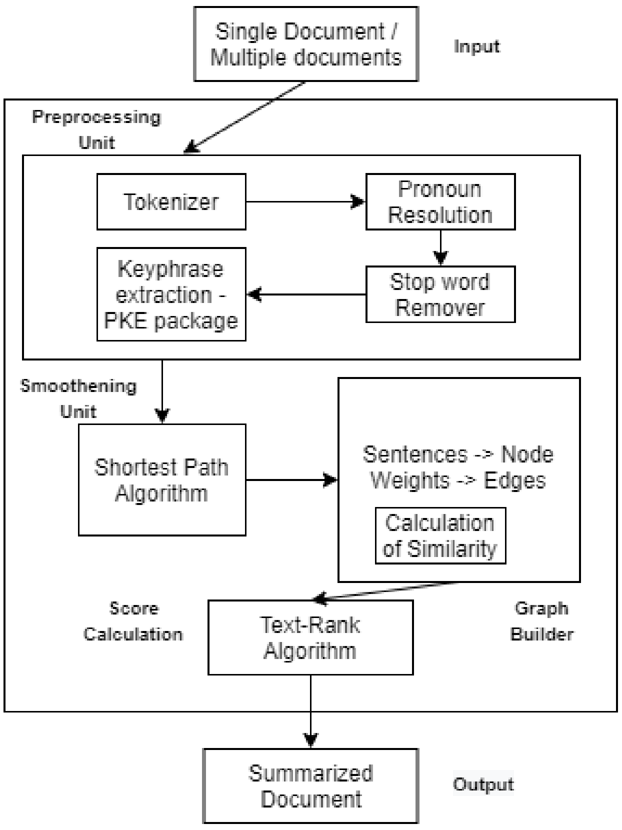


Figure 4. Proposed Model

## 4. IMPLEMENTATION

The different modules involved right from reading the input files to ranking sentences according to content are described in the following subsections.

### 4.1 Algorithms Used

#### *4.1.1 TextRank*

The TextRank algorithm determines the similarity of each sentence with other sentences in a given text. Based on this, TextRank scores are given to each sentence. Every sentence is stored as a node of a graph. The values are iterated over multiple times until they converge. The sentence with the highest score is the sentence which is the most similar to other sentences. Cosine similarity is used as a similarity measure for TextRank.

#### *4.1.2 Dijkstra's Shortest Path Algorithm*

Dijkstra's Shortest Path algorithm finds the shortest path between nodes of a graph. It uses a queue for storing and querying partial solutions sorted by distance from the start. Time complexity of this algorithm with a min-Priority queue is

$$O( |V| + |E| * \log|V| )$$

where |V| is the number of vertices and |E| is the number of edges.

This algorithm is used in the project to smoothen the flow of sentences. It selects the sentences in the sequence in which they were present in the input given by the user.

## 4.2 Pre-processing

In the first stage, the model reads the contents of all (or the single) document(s) which are given by the user, such that the documents belong to the same domain. A single domain for all the documents ensures that the summary generated is meaningful and comprehensible by the user. Once all the input documents are read, the text from the documents is extracted. The text is then cleaned i.e. unnecessary white spaces and lines are removed. The neuralcoref and spaCy libraries are used to perform pronoun resolution. Anaphoric ambiguities are removed by using pronoun resolution. The text obtained after pronoun resolution is tokenized. The sentences are then processed to remove stopwords which do not contribute to the meaning of the document content. The stopwords used for reference are from the NLTK corpus. Once tokenized text is obtained without stopwords, it is passed to TextRank module.

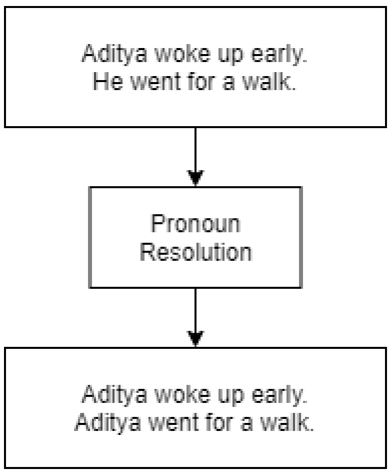


Figure 5. Pronoun Resolution

## 4.3 Keyphrase Extraction

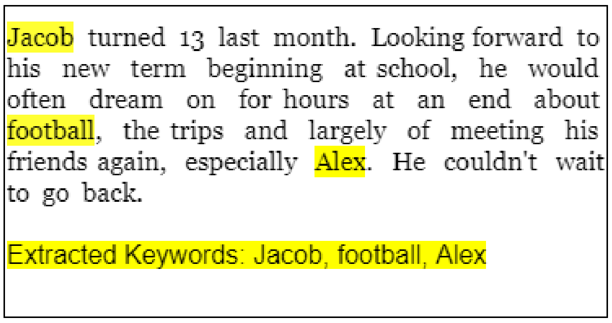
The next step is keyphrase extraction. Keyphrases have an important role in document summarization as they convey the essence of the document with the help of clear, concise and direct words. The extraction is carried out with the help of a pre-trained keyphrase extraction toolkit, pke [10]. It is an open-source toolkit which provides an end-to-end keyphrase extraction pipeline in which each component can be easily modified or extended to develop new models [10]. A similarity matrix is developed by considering sentence-sentence similarity as well as sentence-keyword similarity. To obtain keyphrases, the cleaned text obtained after pronoun resolution is used. Keyphrase extraction is performed using two methods, YAKE and TextRank, with the three best keyphrases being selected in both cases.

### *4.3.1 YAKE*

One implementation is by using YAKE library. The candidate selection is done using bigrams to analyse keyphrases. A one-word window is used to refer to a single word before and after the current word to understand the context. This implementation does not use stemming of words and the stoplist used is the collection of stopwords from NLTK library. The threshold for extraction is set at 0.3 which is required to set a limit for redundant phrases.

### *4.3.2 TextRank*

The second implementation is the TextRank extractor of pke. The sentences are stemmed in this implementation and the window size is set to 3. This implementation uses Parts-Of-Speech tagging to analyse the words in the given text. For selection of the keywords, we analyse only the top 60% of the scored words to limit the processing needed.



Jacob turned 13 last month. Looking forward to his new term beginning at school, he would often dream on for hours at an end about football, the trips and largely of meeting his friends again, especially Alex. He couldn't wait to go back.

Extracted Keywords: Jacob, football, Alex

Figure 6. Keyphrase Extraction

## 4.4 Graph Building

Once the keyphrases for the document have been extracted, these can be used to evaluate sentence importance and calculate edge weights for the graphs. The sentences which were extracted from the document are represented in the form of a graph structure. The nodes of the graph represent the different sentences obtained and the edge weights are symbolic of the relation between the sentences. The weights are a combination of two aspects – sentence semantic similarity and coherence with keyphrases. The similarity measure used is the cosine similarity which uses a vector representation of sentences and keyphrases. We use GloVE embeddings to vectorize the textual aspects.

## 4.5 Shortest Path

Simply extracting the important and relevant sentences however does not always guarantee a summary that can be easily understood by the reader. Hence, the authors pass the processed corpus to the Shortest Path module. It helps to reduce the size of the input corpus. It also helps ensure we have a smooth flow from one sentence to another. The graph fed to the algorithm consists of sentences as the nodes with similarity values calculated as edge weights. The authors use Modified Dijkstra's Algorithm, where higher edge weights are considered to find a smooth path from the first to the last sentence. In this manner, we get a set of reduced sentences, which is then passed to the TextRank module.

## 4.6 TextRank

The sentences obtained from the Shortest Path module are used to re-initialize the graph weights, having an advantage over random initialization for TextRank. After re-building the graph using cosine similarity for edge weights, it is passed to the TextRank module which continuously iterates till convergence to obtain a ranking of sentences according to their importance for summary generation.

## 4.7 Dataset

For training the model to understand optimal parameters, the authors made use of the DUC 2004 dataset provided by NIST. The data provided consisted of a number of tracks, each track consisting of a cluster of 50 folder and each folder in turn consisted of 8-10 documents. The data used for the purpose of testing and deriving parameters belonged to Task 1 and 2 of the dataset. These tasks had corresponding model summaries written by multiple authors, where every author had written summaries for not all but a few of the document clusters. Hence, the processing of the dataset before any kind of testing required the authors to extract the relevant text data from the documents, prepare it in a format suitable to be parsed and map it to the summary by an author to carry out effective evaluation of the model parameters. Since the end model is unsupervised, the data was required to see how the summaries respond to the tuning of parameters and in turn helped establish the final parameters in the unsupervised model.

## 5. EXPERIMENTATION, RESULTS AND DISCUSSIONS

### 5.1 Performance Metric

The authors have used the ROUGE metric for evaluation of the generated summaries. Lin introduced a set of metrics called Recall-Oriented Understudy for Gisting Evaluation (ROUGE) to automatically determine the accuracy of a summary by comparing it to a reference summary. Various Rouge metrics for evaluation are as follows:

#### *5.1.1 ROUGE-N*

Rouge-N metric is a measure of overlapping words which considers N-grams between a model generated summary and a reference summary. The value of N can be 1 i.e. ROUGE-1 represents unigram overlap between the 2 summaries, a value of N = 2 represents bigram overlap and so on.

#### *5.1.2 ROUGE-L*

In this evaluation scheme, the longest common subsequence is identified between the reference and the model generated summary. Rouge-L is more flexible as compared to Rouge-N, but has the drawback that it requires all the N-grams in consecutive positions.

### 5.2 Evaluation

For evaluating on the proposed summary generation scheme, the authors utilized the DUC 2004 dataset, specifically the data limited to Task 1 and 2. The dataset consisted of a total of 50 document clusters pertaining to a news published in the media. Testing on the dataset involved tuning of different parameters and the results obtained over the 50 sets is as mentioned in the table below.

For evaluation 2 models were used, YAKE and TextRank, provided by PKE. YAKE was the first model to be tried and it yielded the ROUGE values as shown in Table 1.

**Table 1. ROUGE values for YAKE keyphrase extraction**

| | Precision | Recall | F-Measure |
|---|---|---|---|
| **ROUGE-1** | 0.4427 | 0.1731 | 0.2472 |
| **ROUGE-2** | 0.0721 | 0.0275 | 0.0395 |
| **ROUGE-L** | 0.2207 | 0.0862 | 0.1232 |

Since the values obtained through YAKE were not optimal, the authors implemented keyphrase extraction using PKE's TextRank model which showed a marked improvement over the previous extraction model. The results obtained as a result are highlighted in Table 2.

**Table 2. ROUGE values for TextRank keyphrase extraction**

| | Precision | Recall | F-Measure |
|---|---|---|---|
| **ROUGE-1** | 0.5138 | 0.1668 | 0.2505 |
| **ROUGE-2** | 0.1064 | 0.0346 | 0.052 |
| **ROUGE-L** | 0.2626 | 0.0853 | 0.1281 |

The comparison between the 2 models with different parameters is further highlighted through the graphs below. The graphs reflect how the evaluation metric values changed in correspondence to the change in parameters used in the model.

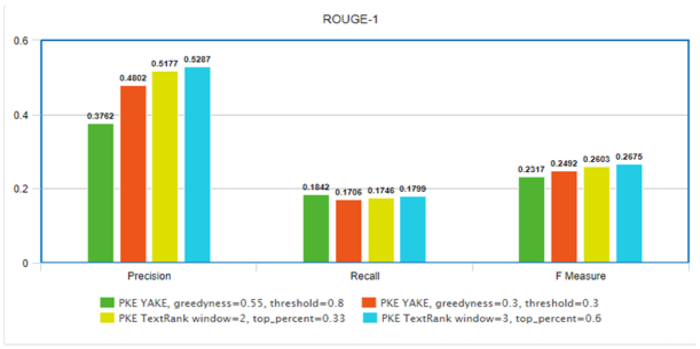


Figure 7. Comparison of ROUGE-1 for YAKE and TextRank

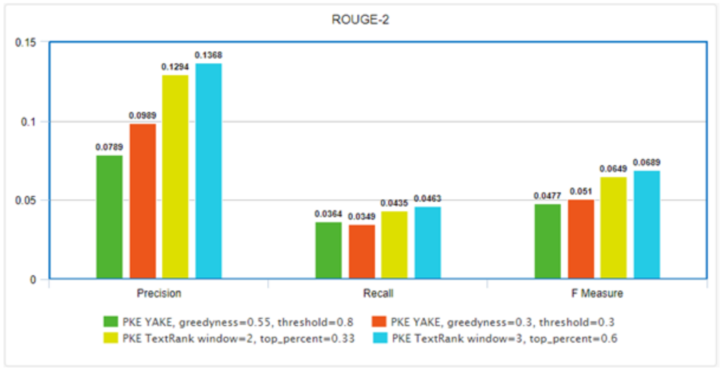


Figure 8. Comparison of ROUGE-2 for YAKE and TextRank

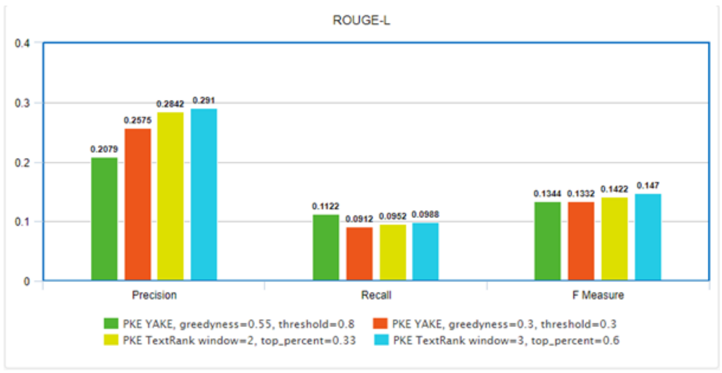


Figure 9. Comparison of ROUGE-L for YAKE and TextRank

## 6. CONCLUSION AND FUTURE SCOPE

Due to the information overload in the internet, news articles, contents on social Medias, automatic document summarization has received a great deal of attention. The proposed model serves the purpose of summarizing a document or a set of documents in a concise manner. The model combines the advantages of different techniques to generate accurate summaries for different documents belonging to the same domain given by the user. Unlike other text summarizers available, the authors have successfully implemented Shortest Path Algorithm in the model to provide crisp summaries. The combined implementation of TextRank, Keyphrase Generator and Shortest Path Algorithm has helped to achieve greater accuracy in generating concise summaries. The proposed methodology is able to cover up drawbacks of traditional summarization techniques and produce good results.

Automatic summarization evaluation is still a very promising research area with numerous challenges ahead. The project can be extended to include features like reading and writing from PDF's and generating summaries as per user specified lengths. An application of this is convenient text-to-speech for blind people; the idea here is to scan and examine over a page from a book, and then read a summary of the page rather than the entire text. This is an effective way to provide page by page synopsis rather than the whole book. The implemented system can be used to provide summaries in different languages in future. Document Visualization is also another topic for research regarding automatic text summarization. Integration into a document visualization tool can be done to visualize documents or document clusters in a number of ways, including as points on a graph. Moreover, the development of more focused summaries using Abstractive Summarization can be applied to achieve more consistent evaluation and to a better convergence between human and automatic evaluation strategies.

## 7. ACKNOWLEDGMENTS

The authors would like to express their gratitude towards their guide, Dr. Ramchandra Mangrulkar, the Head of Department, Dr. Meera Narvekar and the Principal, Dr. Hari Vasudevan for the opportunity provided to them in order to carry out implementation and research on this topic.

## 8. REFERENCES

[1] Akash Ajampura Natesh, Somaiah Thimmaiah Balekuttira and Annapurna P Patil. Graph Based Approach for Automatic Text Summarization in International Journal of Advanced Research in Computer and Communication Engineering Vol. 5, Special Issue 2, October 2016.

[2] Agrawal Nitin, Shikhar Sharma, Prashant Sinha, and Shobha Bagai. "A graph based ranking strategy for automated text summarization." DU J. Undergrad. Res. Innov 1, no. 1 (2015).

[3] Mallick, Chirantana, Ajit Kumar Das, Madhurima Dutta, Asit Kumar Das, and Apurba Sarkar. "Graph-based text summarization using modified TextRank." In Soft Computing in Data Analytics, pp. 137-146. Springer, Singapore, 2019.

[4] Yang, Kang, Kamal Al-Sabahi, Yanmin Xiang, and Zuping Zhang. "An integrated graph model for document summarization." Information 9, no. 9 (2018): 232.

[5] Dunia Hakim, The Role of Key Phrases in Extractive Graph-Based Multi-Document Text Summarization, Stanford University Department of Computer Science dunia@stanford.edu.

[6] Gunes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. Journal of artificial intelligence research, 22:457–479.

[7] Ingo Feinerer and Kurt Hornik. 2017. wordnet: WordNet Interface. R package version 0.1-14.

[8] Jonas Sjobergh, Kenji Araki, "Extraction based summarization using a shortest path algorithm", Proceedings of the Annual Meeting of the Association for Natural Language Processing, 2006.

[9] Dutta M., Das A.K., Mallick C., Sarkar A., Das A.K. (2019) A Graph Based Approach on Extractive Summarization, Emerging Technologies in Data Mining and Information Security: Advances in Intelligent Systems and Computing, vol 813. Springer, Singapore.

[10] Boudin and Florian, pke: an open source python-based keyphrase extraction toolkit, Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations, December 2016.

[11] Mihalcea, Rada, and Paul Tarau. "Textrank: Bringing order into text." In Proceedings of the 2004 conference on empirical methods in natural language processing, pp. 404-411. 2004.

[12] Radev, D. R., Hovy, E., and McKeown, K. (2002). Introduction to the special issue on summarization. Computational Linguistics. 28(4):399–408. [1, 2]