Text Summarisation and Keyword Identification using NLP

Thallam Sai Vasanth*[1] and Dr. Deepak Kumar Singh[1]

[1]Department of Computer Science, Indian Institute of Information Technology Lucknow

***Abstract*—Text summarization and identification of keywords are fundamentals for Natural Language Processing (NLP), which handles and extracts useful information from larger amounts of text data. These methods have improved over time but it is still necessary to improve their clarity and precision, addressing existing gaps in accuracy. This paper explores different approaches for summarizing a text such as taking important sentences from the original text (Extractive Summarization). It includes pre-trained language models, statistical approaches, machine learning models and TF-IDF, TextRank and other methods that identify important words or phrases.**

The results show that using these methods together significantly improves how we find, manage, and analyze information. Our research aims at making text summarization and keyword extraction processes more efficient and accurate hence supporting the development of reliable NLP tools used to manage texts well.

I. Introduction

The digital era has led to an overabundance of data being available on the internet made difficult to comprehend and retrieve information. Text summarization is the process of compressing large volumes of text into summaries, along with identifying significant terms or phrases. These methods help us take in information more easily, which is really useful when dealing with larger text data, and many documents.

Extractive summarization is a basic method in natural language processing (NLP). It involves taking the most important sentences or phrases from a document to create a summary. Unlike other methods that rephrase and combine content to generate summaries. Extractive summarization directly selects and joins passages from the source text instead of abstractive methods. This way is preferred for its simplicity and ability to maintain the initial context and wording.

keyword extraction is all about finding and pulling out the most important words or phrases from the given text. This helps to highlight the main topics and themes, making it easier to understand and organize the content. It's super important for things like SEO, where it helps in improving the visibility and ranking of web pages, suggesting related content, where it assists in suggesting relevant content to users, and customer feedback, where it helps in identifying key themes and sentiments from customer reviews and feedback.

TextRank is a popular algorithm for identifying important keywords in a text, which was introduced in 2004, inspired by Google's PageRank. Created by Mihalcea and Tarau in 2004, it uses a graph-based method to find keywords. In TextRank, the text is modified into a graph where words or phrases are nodes, and the connections between them are edges. Each node's importance is determined by how central it is in the graph. This approach works very well for tasks like keyword extraction and summarizing documents.

In this paper, we will explain how extractive summarization and TextRank work in simple terms and show how they have evolved over time to help us deal with information overload. We will use real examples to make it easy to understand how these techniques can make our lives easier when dealing with lots of text.
The remainder of this paper is organized as follows:

- Section 2: Look into previous research and developments in text summarization and keyword extraction.
- Section 3: Detailed explanation of the methods that are used in our system, summarization, and keyword extraction techniques.
- Section 4:Presenting the experimental results and evaluating the system's performance on various sizes of text data.
- Section 5: End the paper by providing conclusions and discussing possible future improvements for the system.

II. Literature Review

Text summarization has been a key topic in natural language processing (NLP) form many years. This review covers the main developments in this field, from early methods to modern techniques, and highlighting the important research papers.

A. Early Methods

Early methods of text summarization primarily relied on extractive techniques. These techniques involved selecting and compiling key sentences or phrases directly from the original text to create a summary. Hans Peter Luhn[1] was one of the first to try this, using word frequency to find important sentences in a document.His method was simple but set the way for future advancements.

B. Statistical and Machine Learning Techniques

Statistical methods were used by researchers in the 1990s and early 2000s. For instance, in 2001, Gong and Liu[2] introduced a technique called Latent Semantic Analysis (LSA) to find key concepts in documents . Another key development was by Kupiec, Pedersen, and Chen in 1995[3], who used machine learning to train a system to pick important sentences based on various features .

C. Graph-Based Approaches

Graph-based methods for text summarization were introduced in the early 2000s, which changed text summarization in a major way. Mihalcea and Tarau[4] introduced the TextRank algorithm in 2004, inspired by Google's PageRank. TextRank treats sentences as nodes in a graph and links them based on their relationships, making it easier to find the most important sentences for summaries.

D. Neural Network and Deep Learning Models

Text summarization has been significantly improved by deep learning techniques. In 2015, Rush, Chopra[5], and Weston created a neural network model that could generate summaries by learning from large data sets. Then in 2017, the Transformer model introduced by Vaswani et al[6]. became the basis for many of the best performing text summarization systems.

E. Keyword Identification and TextRank

Keyword identification, which is closely related to summarization, has also made a lot of progress. The TextRank algorithm is still widely used for keyword extraction. Research by Rose et al[7]. in 2010 on Rapid Automatic Keyword Extraction (RAKE), along with later improvements to TextRank, have increased the accuracy and efficiency of extracting keywords from text.

Text summarization has progressed from basic methods based on word frequencies to advanced deep learning models. Each phase brought new techniques that improved the ability to process and summarize text. As research continues, methods like TextRank and pretrained models will keep making it easier to handle large amounts of text effectively.

III. Research Methods

In our research, we worked on extractive summarization and the TextRank algorithm to find important keywords. To implement these methods accurately, we used the Gensim library, a robust open-source python library for unsupervised topic modeling and natural language processing (NLP) tasks.

A. Extractive Summarization

Extractive summarization is a technique in natural language processing that creates summaries by selecting and combining the most important sentences or phrases from the original text. It constructs a shorter summary using the exact wording from the source data. The steps involved in our extractive summarization process are :

1) Data Preprocessing: The first step is cleaning the text by removing unnecessary characters, punctuation, and common words like "the" or "and". The text is then tokenized into individual words or phrases, and the sentences are separated. This preprocessing ensures the summary focuses on the most meaningful content. It also reduces the text's complexity, making it easier for the algorithm to identify key sentences.

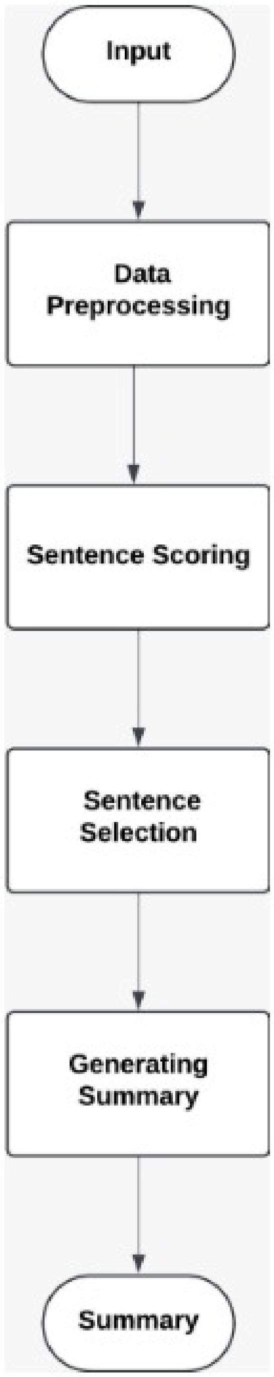

Fig. 1. Working of Extractive Summarization

2) Sentence Scoring: Each sentence in the text is scored based on various features such as word frequency, sentence length, position in the text, and statistical models like TF-IDF (Term Frequency-Inverse Document Frequency). These methods assign a score to each sentence, higher scores indicate more important sentences.

3) Sentence Selection: Sentences with the highest scores are selected to for the summary. The number of sentences chosen depends on how long the summary should be. This can be done by setting a threshold score and selecting all sentences that exceed this threshold.

4) Generating Summary: To generate the summary using the selected sentences. The sentences are combined in the same order as they appear in the original text, ensuring coherence and readability. The final summary contains the most important information from the original text.

B. TextRank Algorithm for Keyword Identification

The TextRank algorithm is a method that uses a graph-based ranking model to find the most important keywords or phrases in a text. It works in a similar way to Google's PageRank algorithm for ranking web pages. A simplified explanation of TextRank algorithm works for keyword identification:

1) Text Preprocessing: Cleaning and preprocessing the text to remove noise and prepare it for keyword extraction. The following steps were taken:
Tokenization: The text was tokenized into sentences and words using the NLTK library.
Stop Words Removal: Common stop words and punctuation were removed to reduce noise and improve the quality of the keywords.
Stemming/Lemmatization: Stemming and lemmatization are techniques used to reduce words to their root words.

2) Graph Construction: The TextRank algorithm builds a graph where each word or phrase in the input data is represented as a node. The connections between these nodes, called edges, show the relationships and similarities between the words. TextRank creates stronger connections between words that often appear together or have similar meanings.

3) Node Scoring/Edge weighting: Each node (word or phrase) is given an initial score. The scores are then repeatedly updated based on how important the neighboring nodes are, similar to how web pages are ranked in PageRank. The formula for updating the scores looks at both the current score of a node and the scores of the adjacent nodes

4) Rank Calculation: Using PageRank algorithm to calculate the importance of each node (word). Iteratively update the rank scores until they converge.

- Initialize a rank score for each vertex.
- Iterative PageRank formula :
  PR(Vi) = (1 - d) + d * (PR(Vj) / outdegree(Vj))
  where d is damping factor, Vi is node, Vj is adjacent node.

5) Keyword Extraction: After the scores stabilize, select the nodes with the highest scores as the keywords. These keywords are the most important terms/ phrase from the given input data.

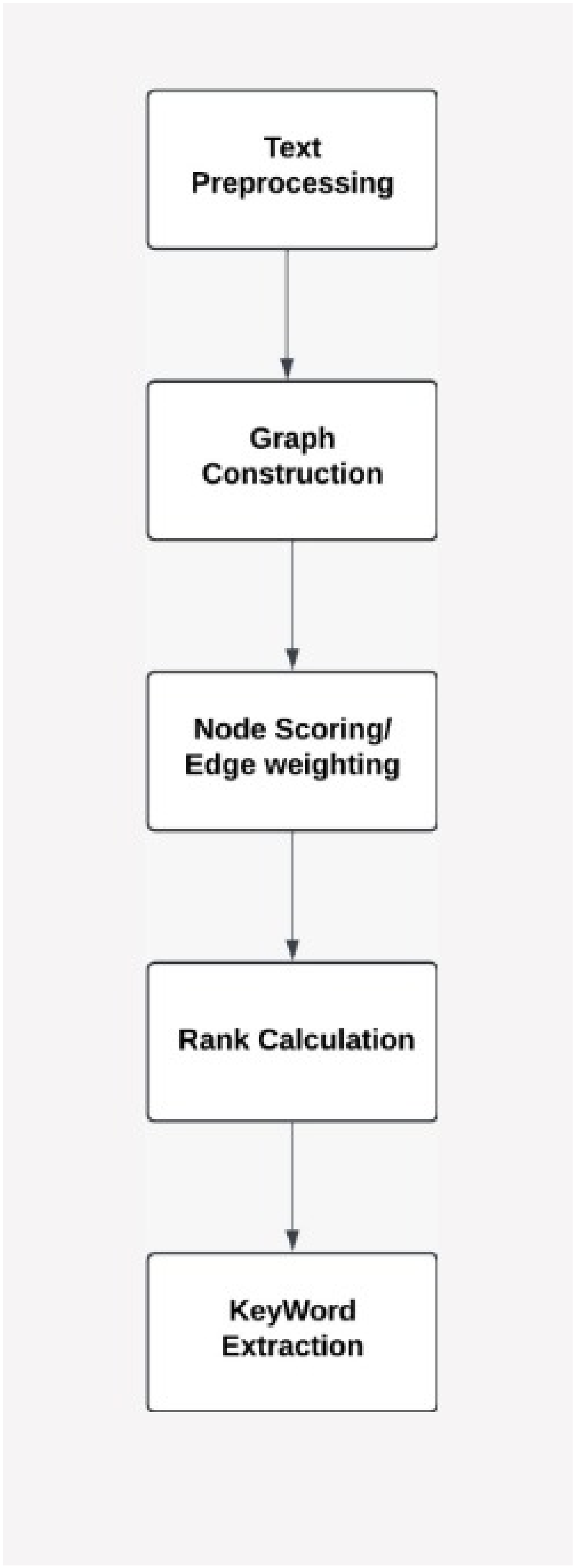

Fig. 2. Working of TextRank Algorithm for keyword indentification.

C. PageRank Algorithm

PageRank is an algorithm created by Larry Page and Sergey Brin at Stanford University for their search engine Google. It is used calculate the weight of the web pages in the search results. The webpages are ranked using a graph-based algorithm. The algorithm measures how important each page is based on the number and quality of other pages that linked to it. Working of the PageRank algorithm:

1) Graph Representation: The web is represented as directed graph where each webpage is node and the hyperlinks between the pages act as directed edges pointing from one node to another node. Note: If a node i.e, a webpage has more number of edges that means it is linked to many webpages that is considered as important.

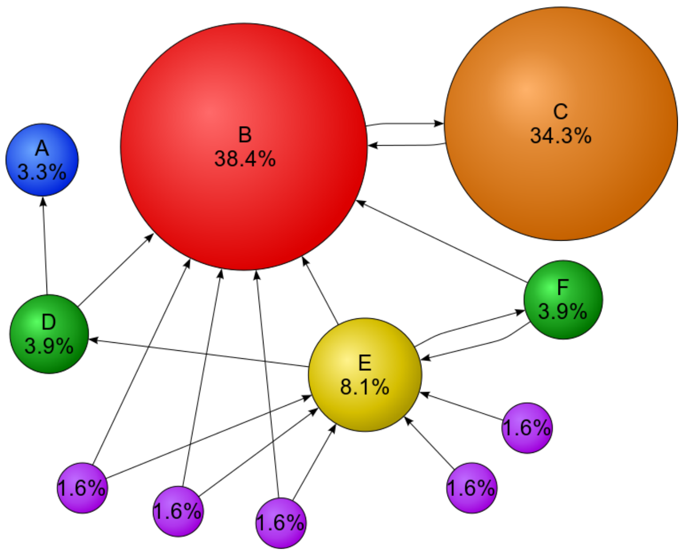

Fig. 3. Illustration of Graph

2) Initialization: Initially, each page is assigned to an equal pagerank value. If there are N pages, each page starts with a rank of 1/N.

3) Calculating the PagaRank: The rank of a page is calculated using the ranks of the pages that are adjacent to it. PagaRank is calculated iteratively using the formula:

$$PR(A) = \frac{1-d}{N} + d\left(\sum_i \frac{PR(B_i)}{L(B_i)}\right)$$

Where;

- PR(A) is the PageRank of page A.
- d is the damping factor, set to 0.85.
- N is the total number of pages.
- PR(Bi) is the PageRank of pages, Bi which is adjacent to page A.
- L(Bi) is the number of outbound links from page Bi.

damping factor, d accounts for the probability that a user will continue clicking on links (usually set to 0.85). The remaining 1-d represents the probability that a user will jump to a random page.

4) Convergence: The PageRank algorithm runs repeatedly, updating the importance scores for each iteration. It continues running these iterations until the PageRank values converge, meaning they don't change much from one iteration to the next. Convergence is typically reached after 20-100 iterations, depending on the size and structure of the network of web pages being analyzed.

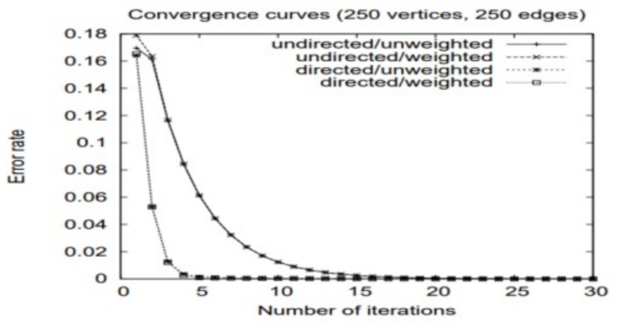

Fig. 4. Plot showing error rate vs number of iterations

D. Gensim Library

Gensim is an open-source Python library that focuses on topic modeling and Natural Language Processing tasks. It provides efficient implementations for various algorithms, including TextRank for keyword extraction and summarization tasks. It's widely used for processing large volumes of text data and extracting meaningful insights. Key features:

1) Topic Modeling: Gensim is widely used for its implementation of various topic modeling algorithms that can identify topics present in a collection of documents. Some of the major topic modeling algorithms it supports are:

- Latent Dirichlet Allocation (LDA): This algorithm automatically discovers topics by analyzing the patterns of words that frequently occur together across the documents.
- Latent Semantic Analysis (LSA): This technique extracts topics or concepts by identifying patterns in the relationships between documents and the terms they contain.
- Hierarchical Dirichlet Process (HDP): This is an extension of LDA that automatically determines the appropriate number of topics based on the data, rather than requiring it to be specified beforehand.

2) Document Similarity: Gensim provides tools to compare documents and find ones that are similar in content. It does this by converting each document into a numeric vector representation, using models like Word2Vec or Doc2Vec. Then it can calculate the similarity between vector representations of different documents using mathematical measures like cosine similarity.

3) Compatibility: Gensim works well together with other commonly used Python libraries for scientific computing and machine learning, such as:

- NumPy - This library provides support for large numerical arrays and mathematical functions.
- SciPy - This library has modules for optimization, linear algebra, integration and statistics.
- scikit-learn - This is a machine learning library with tools for data mining, data analysis and model building.

Gensim integrates very well with these libraries make it easier to use Gensim's text processing and topic modeling capabilities as part of larger machine learning workflows and pipelines.

E. Internal Working of Gensim

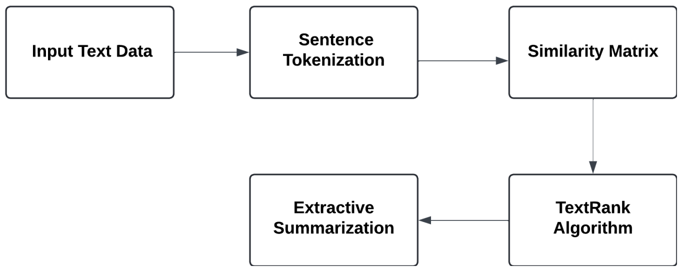

Fig. 5. Working of Gensim

1) Input Text Data: The process starts with the original text data file, which is a lengthy document containing multiple paragraphs and sentences.

2) Sentence Tokenization: The first step is to tokenize the input text into individual sentences using natural language processing techniques known as Tokenization. Each sentence is treated as a distinct unit for the summarization process. Tokenization at the sentence level is crucial because the TextRank algorithm operates on sentences rather than the entire text. By breaking down the text into individual sentences, each sentence can be independently analyzed, scored, and later selected for the summary.

3) Similarity Matrix: After splitting the text into sentences, create a matrix that shows how similar each pair of sentences is to each other. This matrix helps understand the relationships between sentences. These relationships are important for the TextRank algorithm, which uses them to connect sentences in a graph. Cosine similarity is commonly used to measure how similar two sentences are. It compares the angles between their vectors.

4) TextRank Algorithm: Appling the TextRank algorithm to the similarity matrix to rank sentences by importance. TextRank mimics the PageRank algorithm used by Google to rank web pages. It treats each sentence as a node in a graph, with edges (represented by the similarity matrix) connecting related sentences. The mathematical formulation of TextRank algorithm is mentioned earlier. The steps followed in the Algorithm are:

- Initialize Scores: Start with an initial score for each sentence.
- Iterative Updates: Recalculate scores repeatedly until they stabilize:
- Score Calculation: For each sentence, update its score based on the scores of sentences connected to it. Damping Factor: Adjust scores considering a damping factor that helps refine the ranking.

5) Extractive Summary: Once the TextRank algorithm has converged, and each sentence has been assigned a final score representing its importance, the extractive summary can be generated. This is done by selecting the top-ranked sentences based on their TextRank scores, up to a desired summary length. The extractive summary consists of these highly scored and most important sentences from the original document, preserving the key information while reducing redundancy and less relevant content.

IV. The Results and Discussion

In this section, we present the outcomes of our research on extractive summarization and keyword identification using the TextRank algorithm. We discuss the effectiveness of our methods, the role of the Gensim library, and the insights gained from our experiments.

A. Text Data

The experiments were performed using a diverse set of documents from various domains. The data included:

- News Articles: A collection of news articles gathered from multiple online news sources and websites.
- Research Papers: A selection of academic research papers in PDF format, covering different fields and subject areas.

Note, all the text data is in .txt format

B. Extractive Summarization Results

The summarization process involved scoring sentences based on their content and selecting the most significant ones to create concise summaries. We applied extractive summarization to a diverse set of text data.

Key Findings:

- Accuracy: Our extractive summarization method effectively identified the most important sentences from the original text in the summaries. The key information was preserved, and the summaries maintained the proper context.
- Efficiency: Using the Gensim library, we achieved efficient text preprocessing and sentence scoring. The summarization process was fast, even for longer documents, demonstrating the scalability of our approach.
- Readability: The generated summaries were easy to read and understand, making them suitable for various applications such as news aggregation, document preview, and quick information retrieval.

C. *Keyword Identification Results*

We used the TextRank algorithm to identify keywords from the same set of documents. The keywords were extracted based on their importance scores, calculated using the graph-based TextRank approach.

Keyfindings:

- Relevence: Keywords Reflect Main Topics: The keywords identified by TextRank closely matched the main topics of the documents. They accurately represents the key concepts and themes, improving their usefulness for indexing and search optimization.
- Precision: Filtering of Less Important Terms: The TextRank algorithm efficiently filtered out less important words and phrases. By focusing on the most significant terms, it contributed to generating a concise set of keywords.

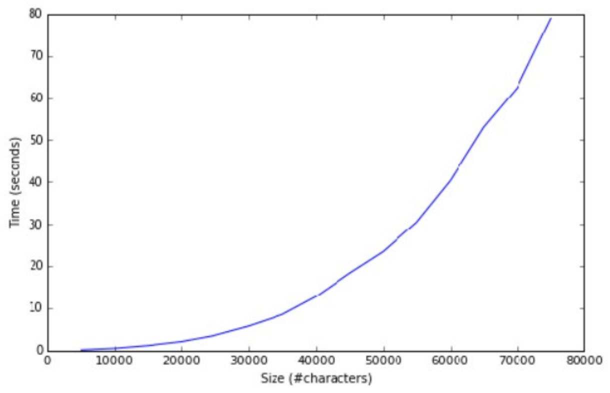

Fig. 6. Execution time vs size of the text document

D. *Role of Gensim Library*

The Gensim library played a pivotal role in implementing both extractive summarization and keyword identification tasks. Its wide array of tools and functionalities facilitated efficient text processing and analysis, contributing significantly to the success of our research.

Advantages of using Gensim:

- Ease of Implementation:
  Gensim provided simple and in-buit functions for text preprocessing, tokenization, and applying the TextRank algorithm. These tools streamlined the implementation process, ensuring minimal errors and smoother development.
- Performance:
  Gensim's optimized algorithms ensured rapid processing times, even when dealing with large volumes of text data. This performance is crucial for real-time applications and scenarios where efficiency is most important.
- Support for Multiple NLP Tasks:
  Gensim excels in supporting various NLP tasks, including topic modeling, word embeddings, and similarity queries.Its comprehensive toolkit provided us with a wide range of functionalities to address our research needs effectively.

E. *Discussion*

The results of our research showes the effectiveness of extractive summarization and the TextRank algorithm for keyword identification. However, there are areas for improvement and future research. For example, integrating more advanced NLP techniques, such as deep learning models, could further improve the accuracy and smoothness of summarization and keyword extraction.

V. Conclusion

In this study, we aimed to explore the effectiveness of extractive summarization and keyword identification using the TextRank algorithm, supported by the Gensim library. We implemented these methods to summarize documents and extract key terms, evaluating their accuracy and efficiency.

A. *Restatement of Aims and Methodological Approach*

Our primary goal was to investigate extractive summarization and keyword identification techniques to minimize textual information effectively. We used the TextRank algorithm, leveraging the graph-based approach to rank sentences and keywords based on their importance. The Gensim library facilitated the implementation of these methods, ensuring efficient text processing and analysis.

B. *Summary of Findings*

Our research demonstrated the efficiency of extractive summarization in producing concise summaries while preserving the essential content of the original documents. The TextRank algorithm successfully identified relevant keywords, capturing the main topics and themes. Through rigorous evaluation, we confirmed the accuracy, efficiency, and scalability of our methods across diverse text genres.

C. *Evaluation of Theoretical and Practical Contribution*

- Advancing NLP Knowledge: Our study helps people understand the field of natural language processing (NLP). We demonstrate how effective extractive summarization and keyword identification techniques can be.
- Explaining TextRank: We explain how the TextRank algorithm ranks sentences and keywords by treating them like points in a graph. This includes its role in ranking sentences and keywords using a graph-based approach.
- Showcasing Gensim: We highlight the Gensim library as a versatile tool for NLP tasks. Our research shows how Gensim can be used effectively, making it easier for others to use and learn from our research.

D. *Recommendations for Future Research*

- Future research could explore the integration of advanced machine learning techniques, such as deep learning models, to further improvement of accuracy and sophistication of summarization and keyword extraction.
- Exploring the application of extractive summarization and keyword identification in multilingual and multimedia contexts could broaden the scope and applicability of the research.

Our study demonstrates the effectiveness of extractive summarization and keyword identification techniques, highlighting their theoretical significance and practical utility. By leveraging the TextRank algorithm and the Gensim library, we provide valuable tools for efficiently processing and extracting information from textual data.

Acknowledgment

I would like to express my sincere gratitude to my supervisor, Dr. Deepak Kumar Singh, professor at the Indian Institute of Information Technology, Lucknow. His guidance, support, and encouragement were invaluable throughout this research. His expertise and insights were instrumental in the successful completion of this study.

References

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

[2] Gong, Y., & Liu, X. (2001). Generic text summarization using relevance measure and latent semantic analysis. Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 19-25.

[3] Kupiec, J., Pedersen, J., & Chen, F. (1995). A trainable document summarizer. Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 68-73.

[4] Mihalcea, R., & Tarau, P. (2004). TextRank: Bringing order into texts. Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, 404-411.

[5] Rush, A. M., Chopra, S., & Weston, J. (2015). A neural attention model for abstractive sentence summarization. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 379-389.

[6] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30, 5998-6008.

[7] Rose, S., Engel, D., Cramer, N., & Cowley, W. (2010). Automatic keyword extraction from individual documents. Text Mining: Applications and Theory, 1-20.

Sign here

Vasanth Deepak

(Thallam Sai Vasanth) (Dr. Deepak Kumar Singh)