Extractive Text Summarization Using Word Similarity-based Spectral Clustering

FAHIM MORSHED, Institute of Information Technology, University of Dhaka, Bangladesh
MD. ABDUR RAHMAN, Centre for Advanced Research in Science, University of Dhaka, Bangladesh
SUMON AHMED, Institute of Information Technology, University of Dhaka, Bangladesh

Extractive Text Summarization is the process of picking the best parts of a larger text without losing any key information. This is really necessary in this day and age to get concise information faster due to digital information overflow. Previous attempts at extractive text summarization, specially in Bengali, either relied on TF-IDF or used naive similarity measures both of these suffers expressing semantic relationship correctly. The objective of this paper is to develop an extractive text summarization method for Bengali language, that uses the latest NLP techniques and extended to other low resource languages. We developed a word Similarity-based Spectral Clustering (WSbSC) method for Bengali extractive text summarization. It extracts key sentences by grouping semantically similar sentences into clusters with a novel sentence similarity calculating algorithm. We took the geometric means of individual Gaussian similarity values using word embedding vectors to get the similarity between two sentences. Then, used TF-IDF ranking to pick the best sentence from each cluster. This method is tested on four datasets, and it outperformed other recent models by 43.2% on average ROUGE scores (ranging from 2.5% to 95.4%). The method is also experimented on Turkish, Marathi and Hindi language and found that the performance on those languages often exceeded the performance of Bengali. In addition, a new high quality dataset is provided for text summarization evaluation. We, believe this research is a crucial addition to Bengali Natural Language Processing, that can easily be extended into other languages.

CCS Concepts: • **Computing methodologies → Information extraction**.

Additional Key Words and Phrases: Extractive Text Summarization, Natural Language Processing, Bengali, Hindi, Marathi, Turkish, Spectral Clustering, Word Embedding, Gaussian Similarity, Evaluation Dataset

ACM Reference Format:
Fahim Morshed, Md. Abdur Rahman, and Sumon Ahmed. 2018. Extractive Text Summarization Using Word Similarity-based Spectral Clustering. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* 37, 4, Article 111 (August 2018), 17 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Text Summarization is the process of shortening a larger text without losing any key information to increase the readability and save time for the reader. But manually summarizing very large texts is a counter-productive task due to it being more time consuming and tedious. So, developing an Automatic Text Summarization (ATS) method that can summarize larger texts reliably is really necessary to alleviate this manual labour [30]. Using ATS to summarize textual data is thus becoming very important in various fields such as news articles, legal documents, health reports, research papers, social media contents etc. ATS helps the reader to quickly and

Authors' Contact Information: Fahim Morshed, f.morshed.opee@gmail.com, Institute of Information Technology, University of Dhaka, Dhaka, Bangladesh; Md. Abdur Rahman, Centre for Advanced Research in Science, University of Dhaka, Dhaka, Bangladesh, mukul.arahman@gmail.com; Sumon Ahmed, sumon@du.ac.bd, Institute of Information Technology, University of Dhaka, Dhaka, Bangladesh.

ACM 2375-4702/2018/8-ART111
https://doi.org/XXXXXXX.XXXXXXX

efficiently get the essential information without needing to read through large amounts of texts [9]. So, ATS is being utilized in various fields, from automatic news summarization, content filtering, and recommendation systems to assisting legal professionals in going through long documents. And researchers in reviewing academic papers by condensing vast amount of informations. It can also play a critical role in personal assistants and chatbots, providing condensed information to users quickly and efficiently [28].

There are two main types of ATS: extractive and abstractive [28]. Extractive summarization, which is the focus of this paper, works by selecting a subset from the source document, maintaining the original wording and sentence structure [21]. In contrast, abstractive summarization involves synthesising new text that reflects information from the input document but does not copy from it, similar to how a human summarizes a text [20]. Both of the method has their own advantage. The abstractive summarization can simulate the human language pattern very well thus increasing the natural flow and readability of the summary. But the extractive method requires much less computation than the abstractive method while also containing more key informations from the input [13].

The key approach to extractive summarization is implementing a sentence selection method to classify which sentences will belong in the summary. For this purpose, previously various simplistic ranking based methods were used to rank the sentences and identify the best sentences as the summary. These ranking methods used indexing [4], statistical [8] or Term Frequency-Inverse Document Frequency (TF-IDF) [6, 26, 27] based techniques to score the sentences and select the best scoring ones. But these methods fail to capture the semantic relationships between sentences of the input due to being simplistic in nature. To capture the semantic relationships between sentences, graph based extractive methods are effective due to the using of sentence similarity graph in their workflow [9]. Graph based methods represent the sentences as nodes of a graph, and the semantic similarity between two sentences as the edge between the nodes [21]. Popular graph based algorithms like LexRank [10] and TextRank [18] build graphs based on cosine similarity of the bag-of-word vectors. LexRank uses PageRank [22] method to score the sentences from the graph while TextRank uses random walk to determine which sentences are the most important to be in the summary. Graph-based methods like TextRank and LexRank offer a robust way to capture sentence importance and relationship, ensuring that the extracted summary covers the key information while minimizing redundancy [9].

Clustering-based approaches are a subset of graph-based approach to extractive text summarization. Here, sentences are grouped into clusters based on their semantic similarity to divide the document into topics, and one representative sentence from each cluster is chosen to form the summary [19]. Clustering reduces redundancy by ensuring that similar sentences are grouped together and only the most representative sentence is selected. This method is effective in summarization of documents with multiple topics or subtopics by picking sentences from each topic. An example of this method can be seen with COSUM [2] where the summarization is achieved using k-means clustering on the sentences and picking the most salient sentence from each cluster to compile in the final summary.

Despite the advancements of ATS in other languages, it remains an under-researched topic for Bengali due to Bengali being a low-resource language. Early attempts at Bengali text summarization relied on traditional methods like TF-IDF scoring to select the best scoring sentences to form the summary [1, 6, 26, 27]. These TF-IDF based approaches, while simple, faced challenges in capturing the true meaning of sentences. This is because TF-IDF based methods treats words as isolated terms resulting in synonyms of words being regarded as different terms [28]. To solve this problem, graph-based methods were introduced in Bengali to improve summarization quality by incorporating sentence similarity but they were still limited by the quality of word embeddings used for the Bengali language. With the advent of word embedding models like FastText [12], it became possible to represent words in a vector space model, thus enabling more accurate sentence similarity calculations. However, existing models that use word embeddings, such as Sentence Average Similarity-based Spectral Clustering (SASbSC) method [23], encountered issues with sentence-similarity calculation when averaging word vectors

to represent the meaning of a sentence with a vector. This method failed in most similarity calculation cases because words in a sentence are complementary to each other rather than being similar, leading to inaccurate sentence representations after averaging these different word vectors. As a result, word-to-word relationships between sentences get lost, reducing the effectiveness of the method.

In this paper, we propose a new clustering-based text summarization approach to address the challenge of calculating sentence similarity accurately. Our method improves upon previous attempts at graph-based summarization methods [5, 23] by focusing on the individual similarity between word pairs in sentences rather than averaging word vectors. We showed that the use of this novel approach greatly improved the accuracy, coverage and reliability of the output summaries due to having a deeper understanding of the semantic similarity between sentences. To calculate sentence similarity, we used the geometric mean of individual word similarities. The individual word similarities were achieved using Gaussian kernel function on a pair of corresponding word vector from each sentence. The word pairs are selected by finding the word vector with the smallest Euclidean distance from the target sentence. Thus, we get the semantic similarity between two sentences which can be used to build an affinity matrix to graphically represent the relationship between the sentences. This graph is clustered into groups to divide the document into distinct topics. One sentence from every cluster is selected to reduce redundancy and increase topic coverage. This method consistently outperforms other graph based text summarization methods such as BenSumm [5], LexRank [10], SASbSC [23] using four datasets on ROUGE metrics [17] as shown in Figure 5 and Table 2. This method performs well in other low resource languages also such as Hindi, Marathi and Turkish due to the language independent nature, as shown in the Table 3.

The main contributions of this paper are: (I) Proposed a new way to calculate similarity between two sentences. (II) Contributes a novel methodology for extractive text summarization for the Bengali language; by improving sentence similarity calculations and enhancing clustering techniques. (III) It offers a generalizable solution for creating less redundant and information rich summaries across languages. (IV) It provides a publicly available high quality dataset of 500 human generated summaries.

The rest of the paper is organized as follows: The Related works and Methodology are described in section 2 and 3 respectively. Section 4 illustrates the result of the performance evaluation for this work. Section 5 discusses the findings of the paper in more depth, and section 6 concludes the paper.

2 Related Work

Text summarization has been an important necessity for textual data consumption for a long time because of its ability to compress a given input text into a shortened version without losing any key information. For this reason, automating the text summarization process has been a research problem for NLP. Thus researchers attempted automatic text summarization for a long time too. At first, indexing-based text summarization methods were attempted such as the work by Baxendale [4]. This method scored sentences based on the presence of indexing terms in the sentence and picked the sentences with the best score. But this type of method failed to capture the topic and essence of the input text as we wouldn't have the topic keywords of an unforeseen input document. To solve this issue, text summarization with statistical methods like TF-IDF became very popular due to its ability to capture the important topic words of a document in an unsupervised manner. Edmundson [8] proposed a method which can focus on the central topic of a document by using the TF-IDF measure. It uses two metrics, Term Frequency (how many times a term appears in the input) and Inverse Document Frequency (inverse of how many documents the term appears in a large text corpus) to calculate the importance of a term in a document. Using TF-IDF helps to identify the words that are common in the input text but not as common in the language in general and thus classifying them as the central topic of the document. But this method is also error-prone due to its consideration of every word as a unique isolated term without any semantic relation with other words. This leads to the method often missing some central topic if it gets divided into too many synonyms.

The problems faced by topic-based summarization methods were alleviated by graph-based extractive text summarization methods which brought new modern breakthroughs. Graph-based methods like LexRank [10] and TextRank [18] were able to capture the relationship between sentences more accurately due to use of sentence similarity graph in their process. LexRank [10] calculates the similarity graph by using cosine similarity of bag-of-words vectors between two sentences from the input. The most important sentences from the graph are classified using the PageRank [22] algorithm on the graph. PageRank ranks those sentences higher who are more similar with other high ranked sentences. Another graph-based method, TextRank [18] also uses a similar approach while building the similarity graph. In the graph for every sentence, TextRank distributed the score of that sentence to its neighbours using a random walk. This process is repeated over and over until the scores converge. Then the method picks the sentences with the best scores as the summary. Although graph-based methods such as LexRank and TextRank models are Ground-breaking compared to their time, they still lacked fundamental understanding of the words involved in a sentence due to not using any vector representation of the semantic relationship between the words involved.

To solve the problem of representing semantic relationship, a mathematical abstraction called Word Vector Embedding was conceptualized by the seminal work of Salton et al. [25]. Word Vector Embedding uses a word vector space as a mathematical abstraction of a lexicon where the closer two words are semantically, the closer they are in the vector space. Using word vector for graph based text summarization has only been started to be attempted recently [16] due to the lack of fully developed word embedding datasets.

Although text summarization have been a forefront of NLP research, text summarization research in Bengali is a more recent development than in other high resource languages. So, a lot of sophisticated approaches from other languages haven't been attempted at Bengali yet. Earlier bengali extractive methods have been focused on some derivative of TF-IDF based text summarization such as the methods developed by Chowdhury et al. [5], Dash et al. [6], Sarkar [26] etc. Sarkar [26] used a simple TF-IDF score of each sentence to rank them and pick the best sentences to generate the output summary. Dash et al. [6] used weighted TF-IDF along with some other sentence features like sentence position to rank the sentences. Chowdhury et al. [5] however, used TF-IDF matrix of a document to build a graph and perform hierarchical clustering to group sentences together and pick one sentence from each group. One shortcoming of this method is that TF-IDF matrix is not semantically equivalent to the actual sentences due to the fundamental issues with TF-IDF. So, the TF-IDF doesn't perfectly represent the semantic relationship between the sentences in the generated graph. Using word vector embedding for Bengali has solved this problem of semantic representation. Word embedding datasets such as FastText [12] dataset[1] with word vector embeddings in 157 languages, including Bengali. Using this dataset, SASbSC [23] proposed a model where they replaced all the words from the input with their respective vector, then averaged the word vectors in a sentence to get a vector representation for the sentence. The sentence average vectors are then used to get the similarity between two sentences using the Gaussian similarity function to build an affinity matrix. This affinity matrix is used to cluster the sentences using spectral clustering to group sentences into distinct topics. The summary is generated after picking one sentence from each cluster to reduce redundancy and increase coverage.

But the sentence average method suffers critically due to its inability in capturing accurate relationship between sentences. This happens due to words in a sentence generally not having similar meaning with each other, instead they express different parts of one whole meaning of a sentence. This makes the words more complementary instead of being similar leading to word vectors being scattered throughout the word vector space. This characteristics makes the sentence average vectors always tending towards the centre and not representing the semantic similarity accurately. An example of this happening is shown in Figure 1 where the distance between the sentence average vectors are being misleading. In the figure, scenario (a) shows two sentence with word vectors very closely corresponding with each other. On the other hand, scenario (b) shows two sentences without

[1] *https://fasttext.cc/docs/en/crawl-vectors.html*

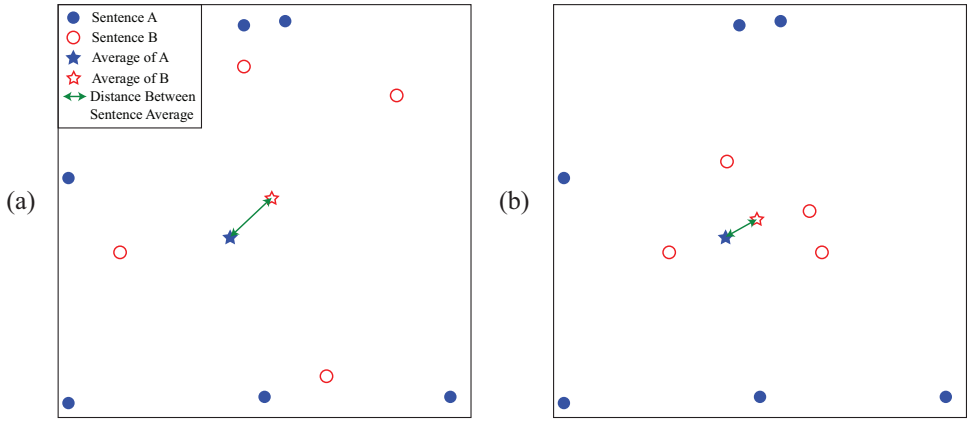


Fig. 1. A scenario where sentence averaging method fails. (a) shows a scenario where the distance between sentence average vectors are larger than (b) despite the word vectors from (a) being more closely related than (b).

any significant word correspondence. But the scenario (a) has a larger distance between sentence average vectors than scenario (b) despite having more word correspondence. This larger distance makes the Gaussian similarity between the sentences lower due to the inverse exponential nature of the function. The lower similarity leads to the graphical representation being less accurate and thus failing to capture the true semantic relationship within the sentences. This shortcoming of the method has been one of the key motivations for this research.

3 Methodology

The summarization process followed here can be boiled down as two basic steps, grouping all the close sentences together based on their meaning to minimize redundancy and picking one sentence from each group to maximize sentence coverage. To group semantically similar sentences into clusters, we build a sentence similarity graph and perform spectral clustering on it [23]. The sentence similarity graph is produced using a novel sentence similarity calculation algorithm that uses geometric mean of Gaussian similarity between individual word pairs from the two sentences. The Gaussian similarity is calculated using the vector embedding representations of the words. On the other hand, we used TF-IDF scores to pick the highest ranked sentences from a cluster [1, 6, 26, 27]. The summarization process followed here involves 4 steps. These are, Pre-processing, Sentence similarity calculation, Clustering and Summary generation. These steps are illustrated in Figure 2 and further discussed in the following subsections.

3.1 Preprocessing

Preprocessing is the standard process of NLP that transforms raw human language inputs into a format that can be used by a computer algorithm. In this step, the document is transformed into a few set of vectors where each word is represented with a vector, each sentence is represented as a set of vectors and the whole document as a list containing those sets. To achieve this representation, the preprocessing follows three steps. These are tokenization, stop word removal, and word embedding. A very common step in preprocessing, word stemming, isn't used here as the word embedding dataset works best for the whole word instead of the root word. These steps are further discussed below.

Tokenization is the step of dividing an input document into sentences and words to transform it into a more usable format. Here, the input document is represented as a list of sentence and the sentences are represented as a

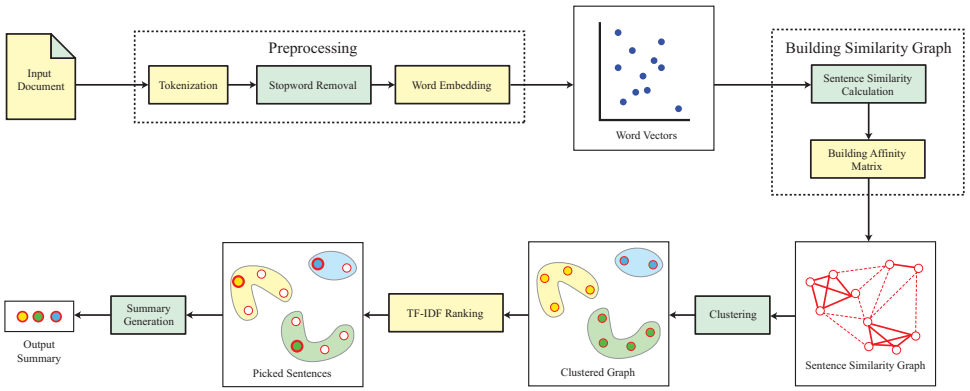


Fig. 2. Process Flow Diagram

list of words. Stop words, such as prepositions and conjunctions, add sentence fluidity but don't carry significant meaning. Removing these words allows the algorithm to focus on more impactful words. Word Embedding is the process of representing words as vector in a vector space. In this vector space, semantically similar words are placed closer together so that the similarity relation between words can be expressed in an abstract mathematical way. Each word from the tokenized and filtered list is replaced with their corresponding vectors. If some words aren't present in the dataset, they are considered too rare and thus ignored. Following these steps, the input document is transformed into a set of vectors to be used in sentence similarity calculation.

3.2 Sentence Similarity Calculation

Sentence similarity is the key criteria to build a graphical representation of the semantic relationship in the input document. This graphical representation is expressed via an affinity matrix to cluster the semantically similar sentences. The nodes in the affinity matrix represents the sentences of the input and the edges of the matrix represents the similarity between two sentence. Here, we proposed a novel sentence similarity calculation technique using individual Gaussian similarity of word-pairs to construct an affinity matrix. To calculate the sentence similarity between two sentences, we adhere to the following steps.

Firstly, for every word of a sentence, we find its closest counterpart from the other sentence to build a word pair. The Euclidean distance between the vector representation of the word-pairs is defined as the Most Similar Word Distance (D_{msw}). The process of calculating the D_{msw} is shown in the Equation 1. In this equation, for every word vector x, in a sentence X, we find the Euclidean distance ($d(x, y_i)$) between the word vectors x and y_i where y_i is a word vector from the sentence Y. The lowest possible distance in this set of Euclidean distance is the D_{msw}.

$$D_{msw}(x, Y) = min(\{d(x, y_i) : y_i \in Y\}) \tag{1}$$

Then, we calculate the D_{msw} for every word of the two sentences X and Y to make the sentence similarity calculation symmetric. This process is shown in the Equation 2 where D is a set containing all the D_{msw} from both X and Y that would be used in the later steps.

$$D = \{D_{msw}(x, Y) : x \in X\} \cup \{D_{msw}(y, X) : y \in Y\} \tag{2}$$

After this, the word similarity between the word pairs is calculated to get the degree of correspondence between the two sentences. Here, the word similarity is calculated using Gaussian kernel function for the elements of the set D; Gaussian kernel functions provides a smooth, flexible and most representative similarity between

two vectors [3]. The process of calculating word similarity (W_{sim}) is given in the Equation 3. In this equation, for every element D_i in set D, we calculate the Gaussian similarity to obtain word similarity. In the formula for Gaussian similarity, σ denotes the standard deviation that can be used as a control variable. The standard deviation represents the blurring effect of the kernel function. A lower value for σ represents a high noise sensitivity of the function [3]. The value of sigma was fine-tuned to be 5×10^{-11} which gives the best similarity measurement. The process of fine-tuning is described in the experimentation section (section 4.4.1).

$$W_{sim} = \{exp\left(\frac{-D_i^2}{2\sigma^2}\right) : D_i \in D\} \tag{3}$$

Finally, the sentence similarity between the two sentences $Sim(X, Y)$ is calculated using geometric mean of the word similarities to construct an affinity matrix. The geometric mean makes the similarity value less prone to effects of outliers thus it makes the calculation more reliable. This process is shown in the Equation 4; the geometric mean of each w_{sim} value for the two sentences is simplified in the Equation 4 to make the calculation process more computation friendly.

$$\begin{aligned} Sim(X, Y) &= \left(\prod_{i=1}^{n} W_{Sim_i}\right)^{1/n} \\ &= \left(exp\left(\frac{-D_1^2}{2\sigma^2}\right) \cdot exp\left(\frac{-D_2^2}{2\sigma^2}\right) \cdot \ldots \cdot exp\left(\frac{-D_n^2}{2\sigma^2}\right)\right)^{1/n} \\ &= exp\left(-\frac{D_1^2 + D_2^2 + \ldots + D_n^2}{2n\sigma^2}\right) \\ &= exp\left(-\frac{\sum_{i=1}^{n} D_i^2}{2n\sigma^2}\right) \end{aligned} \tag{4}$$

By following steps described above, we get a similarity value for two sentences. This value solves the lack of local word correspondence problem faced by the word averaging based similarity calculation method [23]. Figure 3 demonstrates the merit of the method claimed above. Figure 3 shows that the proposed method can solve the local word correspondence problem faced by word averaging method. In the figure, the scenario 3(a) has a set of smaller D_{msw} than the scenario 3(b). Having smaller D_{msw} makes the individual word similarities W_{sim} larger due to the nature of Gaussian kernel function. These values would result in a higher sentence similarity for the sentences in the scenario 3(a) than in the scenario 3(b). This solved the problem showed in the Figure 1 where the scenario 1(a) has a larger sentence average distance than 1(b) resulting in 1(a) having a smaller sentence similarity than 1(b). The whole process of sentence similarity calculation is shown in the Algorithm 1. In this algorithm, we calculate an affinity matrix using the input word vector list. We took the most similar word distance D_{msw} in line 8–13 and 18–23 for each word (line 7 and 17) of a sentence pair (line 3 and 6). Sum of D^2 from Equation 4 is calculated in the lines 14 and 24 to be used in the calculation of sentence similarity (line 27). The similarity is used to construct an affinity matrix A (line 28).

3.3 Clustering

Clustering is a key corner stone of the proposed method where we aim to cluster semantically similar sentences together to divide the input document into multiple topics. Clustering the document minimizes redundancy in the output summary by ignoring multiple sentences from the same topic. For clustering, spectral and DBSCAN methods were considered due to their capability of being able to cluster irregular shapes. However, spectral clustering was found to perform better than DBSCAN because, smaller input documents have lower density which hinders DBSCAN [23].

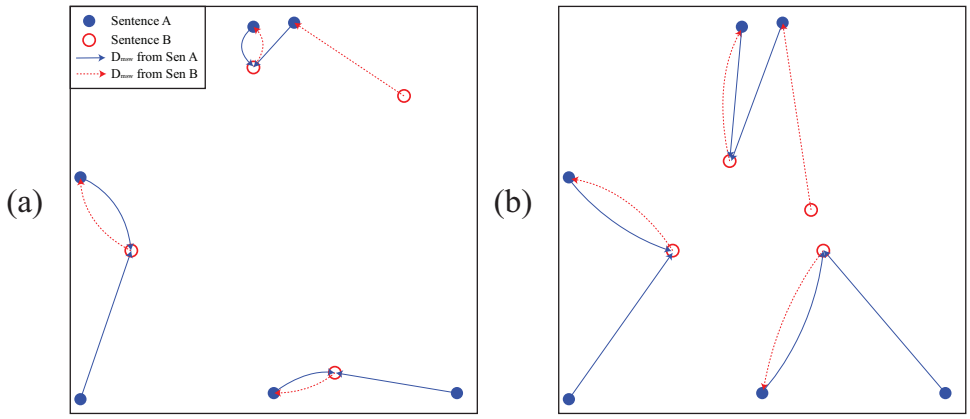


Fig. 3. Emphasis of local word correspondence in D_{msw} method. Here, scenario (a) has a larger similarity value due to having a set of smaller D_{msw} than scenario (b)

On the contrary, spectral clustering takes the affinity matrix of a graph as input and returns the grouping of graph nodes by transforming the graph into its eigenspace [29]. The following Equation 5 shows the process of building an affinity matrix. Here, for every sentence pair S_i and S_j, we calculate their sentence similarity and place it in both A_{ij} and A_{ji} of the affinity matrix A.

$$A_{ij} = A_{ji} = Sim(S_i, S_j) \tag{5}$$

The affinity matrix is clustered into a reasonable, $k = \lceil N/5 \rceil$ groups to achieve an output summary. This summary, is short while the sentence groups resulting from clustering is also not too broad.

3.4 Summary Generation

Output summary is generated by selecting one sentence from each cluster achieved in the previous step to minimize topic redundancy and maximize topic coverage. To select one sentence from a cluster, we perform TF-IDF ranking on the sentences inside a cluster and pick the sentence with the highest TF-IDF score. To get the TF-IDF score of a sentence, we take the sum of all TF-IDF values for the words in that sentence. The TF-IDF value for a word is achieved by multiplying how many time the word appeared in the input document (Term Frequency, TF) and the inverse of how many document does the word appear in a corpus (Inverse Document Frequency, IDF). The process of scoring sentences are shown in the Equation 6 where, for each word W_i in a sentence S and a corpus C, we calculate the TF-IDF score of a sentence.

$$\text{TFIDF}(S) = \sum_{i=1}^{\text{length}(S)} \text{TF}(W_i) \times \text{IDF}(W_i, C) \tag{6}$$

The sentences with the best TF-IDF score from each clusters are then compiled as the output summary in their order of appearance in the input document to preserve the original flow of information. The process of generating output summary is further expanded in the Algorithm 2. After the clustering step (line 2), we took the TF-IDF score (line 7) of each sentence in a cluster (line 6). For each cluster (line 4), we pick the best scoring sentence (line 9). These sentences are then ordered (line 11) and concatenated (line 13–15) to generate the output summary.

Algorithm 1: Sentence Similarity

Function `similarity`($sentence_i$,$sentence_j$)**:**
- $D_{\text{Square}} \leftarrow 0$;
- $n \leftarrow 0$;
- **foreach** $word_i$ ***in*** $sentence_i$ **do**
  - $D_{\text{msw}} \leftarrow \infty$;
  - **foreach** $word_j$ ***in*** $sentence_j$ **do**
    - **if** `Distance`$(word_i, word_j) < D_{msw}$ **then**
      - $D_{\text{msw}} \leftarrow$ `Distance`(word_i, word_j);
    - **end**
  - **end**
  - $D_{\text{Square}} \leftarrow D_{\text{Square}} + D_{\text{msw}}^2$;
  - $n \leftarrow n + 1$;
- **end**
- **foreach** $word_j$ ***in*** $sentence_j$ **do**
  - $D_{\text{msw}} \leftarrow \infty$;
  - **foreach** $word_i$ ***in*** $sentence_i$ **do**
    - **if** `Distance`$(word_i, word_j) < D_{msw}$ **then**
      - $D_{\text{msw}} \leftarrow$ `Distance`(word_i, word_j);
    - **end**
  - **end**
  - $D_{\text{Square}} \leftarrow D_{\text{Square}} + D_{\text{msw}}^2$;
  - $n \leftarrow n + 1$;
- **end**
- similarity $\leftarrow \exp\left(\frac{-D_{\text{Square}}}{2 \times n \times \sigma^2}\right)$;
- **return** similarity;

end

4 Result

The performance of the proposed method has been compared against three Bengali text summarization methods to evaluate the correctness of generated summaries. The three methods, which have been used as a benchmark, are BenSumm [5], LexRank [10] and SASbSC [23]. All of these methods have been evaluated using four datasets to test the robustness of the model for Bengali text summarization for input from different sources. For evaluation, the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [17] metric has been used. Details about the models, datasets and evaluation metrics are provided in the following sections.

4.1 Text Summarization Models

We implemented Bensumm [5] and SASbSC [23], two recent Bengali extractive models, and LexRank [10], a popular benchmarking model for extractive text summarization to evaluate the effectiveness of the proposed WSbSC method. These methods are further discussed in the following section.

WSbSC is the proposed model for this research. We find the Gaussian similarity for word-pairs from two sentences and take their geometric mean to get the similarity between two sentences. We use the similarity

Algorithm 2: Summary Generation

```
k ← ⌈ length(A) / 5 ⌉;
clusters ← spectral_clustering(adjacency = A, k);
indexes ← {};
foreach cluster_i in clusters do
    TFIDF ← {};
    foreach index in cluster_i do
        TFIDF.append(tfidf_sum(sentences[index]));
    end
    indexes.append(indexof(max(TFIDF)));
end
sort(indexes);
S ← "";
foreach i in indexes do
    S ← S+ sentences[i];
end
return S;
```

value to perfom spectral clustering to group sentences into groups and extract a representative sentence using TF-IDF score. The extracted sentences are used to generate the output summary which minimizes redundancy and maximizes coverage.

SASbSC [23] is the first method that introduced the approach of clustering sentences using sentence similarity. However, it uses the average of word vectors in a sentence for calculating similarity. After clustering the sentences based on their similarity, cosine similarity between the average vectors are used to pick the best sentence from a cluster.

BenSumm [5] is another recent research that describes an extractive and an abstractive text summarization technique. We compared the extractive technique with our model to ensure a fair and balanced comparison. Here, similarity matrix is built using TF-IDF which groups the sentences using agglomerative clustering. A Github implementation[2] provided by the authors is used in the comparison process.

LexRank [10] uses a TF-IDF based matrix and Googles PageRank algorithm [22] to rank sentences. Then the top ranked sentences are selected and arranged into summary. An implemented version of this method is available as lexrank[3] which is used in the comparison process using a large Bengali wikipedia corpus[4].

4.2 Evaluation Datasets

We used four evaluation dataset with varying quality, size and source to examine the robustness of the methods being tested. The first dataset is a **self-curated** extractive dataset that we developed to evaluate the performance of our proposed method. An expert linguistic team of ten members summarized 250 news articles of varying sizes to diversify the dataset. Each article is summarized twice by two different experts to minimize human bias in

[2] *https://github.com/tafseer-nayeem/BengaliSummarization*
[3] *https://pypi.org/project/lexrank/*
[4] *https://www.kaggle.com/datasets/shazol/bangla-wikipedia-corpus*

the summary. In total, 500 different document-summary pairs are present in this dataset. The dataset is publicly available on Github[5] for reproducibility.

The second dataset is collected from **Kaggle** dataset[6] which is a collection of summary-article pair from "The Daily Prothom Alo" newspaper. The dataset is vast in size however the quality of the summaries are poor. All the articles smaller than 50 characters were discarded from the dataset. The articles with unrelated summaries were also removed from the dataset to improve the quality. After filtering, total 10,204 articles remained, each with two summaries in the dataset.

The third dataset we used for evaluation is **BNLPC** which is a collection of news article summaries [14]. This was collected from GitHub[7] for experimentation that contains one hundred articles with three different summaries for each article.

The fourth dataset is collected from **Github**[8]. The dataset contains 200 documents each with two human generated summaries. These documents were collected from several different Bengali news portals. The summaries were generated by linguistic experts which ensures the high quality of the dataset.

4.3 Evaluation Metrics

To evaluate the correctness of generated summaries against human written summaries, ROUGE metric [17] is used. The method compares a reference summary with a machine generated summary to evaluate alignment between the two. It uses N-gram-based overlapping to calculate a precision, recall and F-1 score of each summary. The Rouge package[9] is used to evaluate the proposed models against human generated summaries. The package has three different metrics for comparison of summaries. These are are:

(1) **ROUGE-1** uses unigram matching to find how similar two summaries are. It calculates total common characters between the summaries to evaluate the performance. But using this metric also can be misleading as very large texts will share a very high proportion of uni-grams between them.
(2) **ROUGE-2** uses bi-gram matching to find how much similar the two summaries are in a word level. Having more common bigrams between two summaries indicates a deeper syntactic similarity between them. Using this in combination with the ROUGE-1 is a standard practice to evaluate machine generated summaries [9].
(3) **ROUGE-LCS** finds the longest common sub-sequence between two summaries to calculate the rouge scores. It focuses on finding similarity in the flow of information in the sentence level between two summaries.

In this study, we compared the F-1 scores from each of these metrics for the four models.

4.4 Experimentation

The proposed model contains three main steps, similarity calculation, clustering and sentence extraction. Each of these steps have different techniques or values that can be experimented on to find the most suited strategy. For sentence similarity calculation, different values for standard deviation (σ) can be fixed to get the most representative semantic similarity value. Spectral clustering was found to work best for the clustering step through experimentations done by other researchers [23]. To extract the best sentence from a cluster, lead extraction and TF-IDF ranking strategies was considered. Thus, experimentations on finding the most suited standard deviation value and sentence extraction was conducted. These experimentations are described in the following sections.

[5] *dataset link*
[6] *https://www.kaggle.com/datasets/towhidahmedfoysal/bangla-summarization-datasetprothom-alo*
[7] *https://github.com/tafseer-nayeem/BengaliSummarization/tree/main/Dataset/BNLPC/Dataset2*
[8] *https://github.com/Abid-Mahadi/Bangla-Text-summarization-Dataset*
[9] *https://pypi.org/project/rouge/*

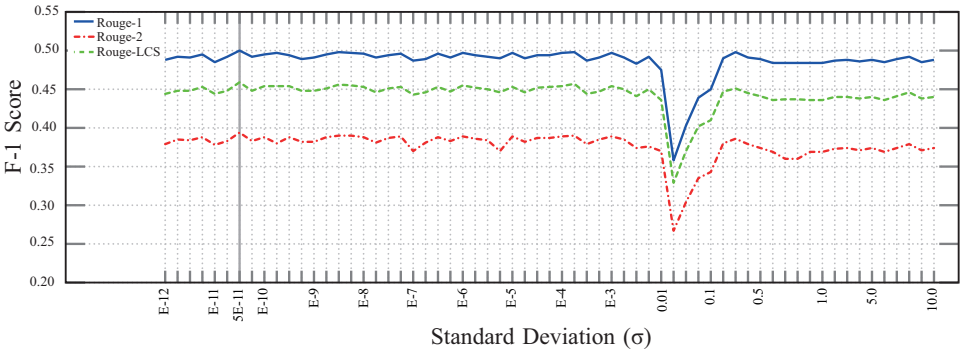


Fig. 4. Fine-tuning for different standard deviation (σ) values

Method	Rouge-1	Rouge-2	Rouge-LCS
Lead extraction	0.47	0.36	0.43
TF-IDF ranking	**0.50**	**0.40**	**0.46**

Table 1. Comparison of Result of different ranking techniques

4.4.1 Fine-tuning Standard Deviation (σ): Standard deviation (σ) plays a crucial role for sentence similarity calculation (Equation 4). Therefore, to fix the most suited σ value sixty-three different values were experimented on. These values ranged from 10^{-12} to 10 on regular intervals. After experimentation, 5×10^{-11} was fixed as the value for σ that gives the most representative semantic relation between sentences. The result for fine-tuning process is shown in Figure 4.

4.4.2 Different Sentence Extraction Techniques From Clusters: We implemented two sentence extraction methods to pick the most representative sentence from each cluster. Firstly, the lead extraction method is used to select the sentence that appears first in the input document. Because, generally the earlier sentences in an input contain more information on the context of the input document. Secondly, extracting sentences based on their TF-IDF score was also experimented on. In Table 1, the TF-IDF ranking is shown to performs better than the lead extraction method.

4.5 Comparison

The performance of the proposed method (WSbSC) is compared with BenSumm [5], SASbSC [23] and LexRank [10] on four datasets (Self-Curated (SC), Kaggle, BNLPC, Github) using the average F-1 score for three ROUGE metrics (Rouge-1, Rouge-2, Rouge-LCS). The comparative results of this evaluation are shown in Table 2 where our proposed model performs 11.9%, 24.1% and 16.2% better than SASbSC in Rouge-1, Rouge-2 and Rouge-LCS respectively on the self-curated dataset. It performs 68.9%, 95.4% and 84.6% better in Rouge-1, Rouge-2 and Rouge-LCS respectively than BenSumm on the Kaggle dataset. It also performs 3% and 2.6% better in Rouge-2 and Rouge-LCS respectively and ties in Rouge-1 with SASbSC using the BNLPC dataset. It performs 58%, 86.4%, and 67.9% better in Rouge-1, Rouge-2 and Rouge-LCS respectively than BenSumm on the Github dataset. These results are further visualized into three radar charts in Figure 5 to compare the performance of the models on different Rouge metrics. As stated in the charts, the proposed method performed consistently and uniformly across all the datasets regardless of the quality of the dataset. But other models, such as BenSumm performs well in three datasets (SC, GitHub, BNLPC) but fails in the Kaggle dataset. Similarly, SASbSC performs well in SC

Dataset	Model	Rouge-1	Rouge-2	Rouge-LCS
Self-curated	WSbSC (Proposed)	**0.47**	**0.36**	**0.43**
	BenSumm [5]	0.41	0.29	0.36
	SASbSC [23]	0.42	0.29	0.37
	LexRank [10]	0.22	0.14	0.20
Kaggle	WSbSC (Proposed)	**0.49**	**0.43**	**0.48**
	BenSumm [5]	0.29	0.22	0.26
	SASbSC [23]	0.23	0.12	0.18
	LexRank [10]	0.24	0.16	0.22
BNLPC	WSbSC (Proposed)	**0.41**	**0.34**	**0.40**
	BenSumm [5]	0.36	0.28	0.34
	SASbSC [23]	**0.41**	0.33	0.39
	LexRank [10]	0.26	0.19	0.24
Github	WSbSC (Proposed)	**0.49**	**0.41**	**0.47**
	BenSumm [5]	0.31	0.22	0.28
	SASbSC [23]	0.30	0.18	0.24
	LexRank [10]	0.22	0.14	0.20

Table 2. Comparison of average Rouge scores between graph based extractive summarization models on 4 datasets

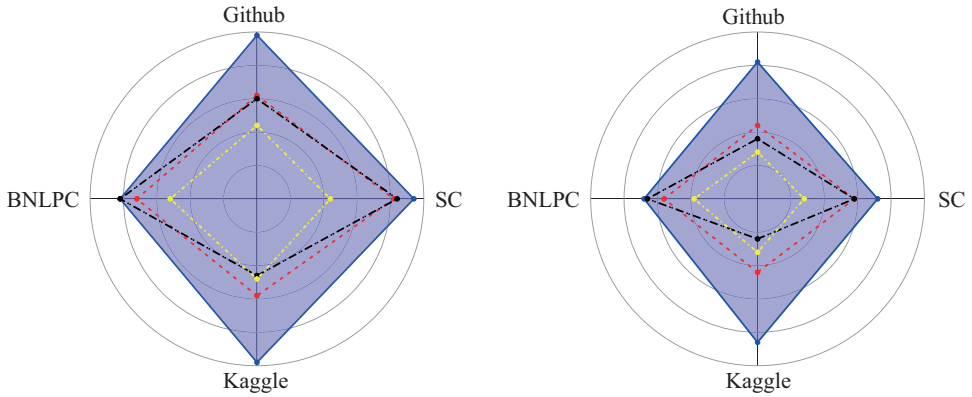
Language	Rouge-1	Rouge-2	Rouge-LCS
Bengali (Self-curated)	0.47	0.36	0.43
Bengali (Kaggle)	0.49	0.43	0.48
Bengali (BNLPC)	0.41	0.34	0.40
Bengali (Github)	0.49	0.41	0.47
Bengali (Average)	0.47	0.38	0.44
Hindi	0.40	0.26	0.36
Marathi	0.50	0.42	0.50
Turkish	0.48	0.39	0.47

Table 3. Comparison of Result of proposed summarization method in other low-resource languages

and BNLPC datasets but its performance decreases sharply in Kaggle and GitHub datasets. LexRank although performs consistently across all datasets but is far lower on average. According to the result analysis in Table 2 and Figure 5, WSbSC is the most accurate and reliable Bengali extractive text summarization model.

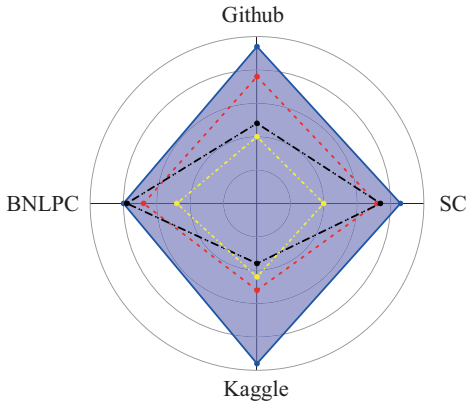
4.6 Implementation Into Other Languages

The proposed model is language-independent thus, it can be extended into other languages too. For this, only a language-specific tokenizer, a stop-word list and a word embedding dataset is required. We implemented this model on three non-bengali datasets to show the language independent nature of the model. To evaluate the quality of the sentence extraction, we tried to find evaluation datasets for summarization on other low resource languages. But could only find relevant datasets in three other languages i.e. Hindi, Marathi and Turkish. We adopted the proposed model into these three low resource languages to check how well it performs. The Table 3 shows the result of the proposed WSbSC method for extractive summarization in other low resource languages. In this table, we can see that the results on Marathi and Turkish are slightly better than the result on



(a) Comparison of ROUGE-1 Values

(b) Comparison of ROUGE-2 Values



(c) Comparison of ROUGE-LCS Values

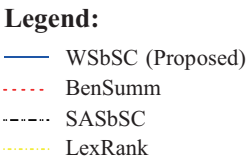


Fig. 5. The Radar chart of the models of being compared on four datasets at once

Bengali. Although it performs slightly lower on Hindi, the score is still similar to Bengali. To evaluate the models performance on Hindi, we used a Kaggle dataset[10] produced by Gaurav Arora. For the Marathi, we used another Kaggle dataset[11] produced by Ketki Nirantar. For the Turkish language, we used a GitHub dataset[12] produced by the XTINGE [7] team for evaluation.

5 Discussion

The results presented at Table 2, 3 and Figure 5 highlights the effectiveness of proposed WSbSC model for extractive text summarization in Bengali, as well as its adaptability to other low-resource languages. This section analyses the comparative results, the strengths and limitations of the proposed method, and potential areas for further research.

[10] *https://www.kaggle.com/datasets/disisbig/hindi-text-short-and-large-summarization-corpus/*
[11] *https://www.kaggle.com/datasets/ketki19/marathi*
[12] *https://github.com/xtinge/turkish-extractive-summarization-dataset/blob/main/dataset/XTINGE-SUM_TR_EXT/xtinge-sum_tr_ext.json*

As evidenced by the results shown in Table 2 and Figure 5, the WSbSC model consistently outperforms other graph-based extractive text summarization models, namely BenSumm [6], LexRank [10], and SASbSC [23], across four datasets. The proposed model shows better performance compared to other three methods on Rouge-1, Rouge-2, Rouge-LCS metrics. This performance improvement can largely be attributed to the novel approach of calculating sentence similarity. While calculating sentence similarity, taking the geometric mean of individual similarity between word pairs overcomes the lack of local word correspondence faced by the averaging vector method [23]. The Gaussian kernel-based word similarity provides a precise semantic relationships between sentences which further contribute in the performance improvement. Another reason for performance improvement is the usage of spectral clustering which is very effective in capturing irregular cluster shapes.

WSbSC includes a novel strategy for calculating sentence similarity despite the existence of other popular methods for comparing two sets of vectors. Our proposed strategy is more suited for similarity calculation in the context of language than other strategies such as Earth Movers Distance (EMD) [24], Hausdorff Distance [15], Procrustes Analysis [11]. EMD [24] and Procrustes Analysis [11] are two very computationally expensive method who also involve scaling or rotating vectors; irrelevant for word vectors due to not holding any semantic meaning. Another method, Hausdorff distance [15] calculates the highest possible distance between vectors from two set. Although similarly expensive as WSbSC, it is easily influenced by outlier words due to only considering the worst case scenario.

On the other hand, the proposed method focuses on local vector similarity between two sets which is more important for words. The Gaussian similarity function captures the proximity of points smoothly, providing a continuous value for similarity between two words in a normalized way. Gaussian similarity is also robust against small outliers due to being a soft similarity measure. Taking geometric mean also helps smooth over similarity values for outlier words.

One of the key strengths of this proposed method is the reduction of redundancy, a common challenge in extractive summarization methods, by grouping semantically similar sentences together. The use of spectral clustering for the grouping task improves the performance by not assuming a specific cluster shape. Another key strength of WSbSC is the improved sentence similarity calculation technique over word averaging method [23], which dilutes the semantic meaning of a sentence. Another key strength is the scalability of the method across languages by requiring very few language-specific resources. This scalability is demonstrated in the experiments with Hindi, Marathi, and Turkish languages (Table 3).

Despite its advantages, the WSbSC model does face some challenges. The model heavily relies on pre-trained word embeddings, which may not always capture the full details of certain domains or newly coined terms. The FastText [12] dataset used here is heavily reliant on wikipedia for training which could introduce some small unforeseen biases. Where the word embeddings do not have some words of a given document, the model's performance could degrade as it leaves those words out of the calculation process. The model also does not take into account the order in which words appear in a sentence or when they form special noun or verb groups. So it can be a little naive in some highly specialized fields.

The proposed WSbSC model represents a significant advancement in Bengali extractive text summarization due to its ability to accurately capture semantic similarity, reduce redundancy, maximize coverage and generalize across languages. While there are still challenges to be addressed, the results of this study demonstrate the robustness and adaptability of the WSbSC model, offering a promising direction for future research in multilingual extractive summarization.

6 Conclusion

In this study, we proposed a Word Similarity-based Spectral Clustering (WSbSC) method for Bengali extractive text summarization which can also be used in other low resource languages. the proposed method used geometric mean of Gaussian similarities between individual word pairs to identify deeper semantic relationship within two sentences. This method for calculating sentence similarity helps WSbSC to significantly outperform other recent graph based extractive text summarization methods on four varying datasets with different sizes and sources. This improvement in performance is also helped by the use of spectral clustering which helps to improve coherence and relevance of the generated summaries by minimizing redundancy and maximizing topic coverage. High performance across three ROUGE metrics on four datasets prove the versatility and robustness of the proposed method. WSbSC can also be extended into other languages as shown through our experimentations on Hindi, Marathi and Turkish languages proving the generalizability of the method. This method addresses the need for an effective summarization technique in Bengali language as Bengali remains under-represented in NLP research despite it being the 7th largest language in the world.

Through results of extensive experimentations, we showed the strengths of the proposed WSbSC method as it outperforms several baseline techniques using a better approach to grouping sentences into key topics. Despite these promising results, there are areas with limitation that requires further improvement. WSbSC may face limitations on highly specialized or domain-specific texts, where deeper linguistic features beyond word similarity could be considered. The lack of consideration for word order in a sentence is also a key limitation which could be explored in the future. Future works could also explore hybrid models that integrate modern post-processing techniques to improve the flow of the output.

In conclusion, this work contributes to the growing body of computational linguistics research focused on low-resource languages like Bengali. The WSbSC method offers a novel approach for extractive summarization by using a new algorithm for calculating similarity between two sentences and sets the stage for further advancements in both Bengali text processing and multilingual summarization techniques.

References

[1] Sumya Akter, Aysa Siddika Asa, Md. Palash Uddin, Md. Delowar Hossain, Shikhor Kumer Roy, and Masud Ibn Afjal. 2017. An extractive text summarization technique for Bengali document(s) using K-means clustering algorithm. In *2017 IEEE International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*. https://doi.org/10.1109/ICIVPR.2017.7890883

[2] Rasim M. Alguliyev, Ramiz M. Aliguliyev, Nijat R. Isazade, Asad Abdi, and Norisma Idris. 2019. COSUM: Text summarization based on clustering and optimization. *Expert Systems* 36, 1 (2019), e12340. https://doi.org/10.1111/exsy.12340

[3] Jean Babaud, Andrew P. Witkin, Michel Baudin, and Richard O. Duda. 1986. Uniqueness of the Gaussian Kernel for Scale-Space Filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8, 1 (1986), 26–33. https://doi.org/10.1109/TPAMI.1986.4767749

[4] P. B. Baxendale. 1958. Machine-Made Index for Technical Literature—An Experiment. *IBM Journal of Research and Development* 2, 4 (1958), 354–361. https://doi.org/10.1147/rd.24.0354

[5] Radia Rayan Chowdhury, Mir Tafseer Nayeem, Tahsin Tasnim Mim, Md. Saifur Rahman Chowdhury, and Taufiqul Jannat. 2021. Unsupervised Abstractive Summarization of Bengali Text Documents. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Association for Computational Linguistics, 2612–2619. https://doi.org/10.18653/v1/2021.eacl-main.224

[6] Satya Ranjan Dash, Pubali Guha, Debasish Kumar Mallick, and Shantipriya Parida. 2022. Summarizing Bengali Text: An Extractive Approach. In *Intelligent Data Engineering and Analytics*. Springer Nature Singapore, 133–140.

[7] İrem Demir, Emel Küpçü, and Alptekin Küpçü. 2024. Extractive Summarization Data Sets Generated with Measurable Analyses. In *Proceedings of the 32nd IEEE Conference on Signal Processing and Communications Applications*.

[8] H. P. Edmundson. 1969. New Methods in Automatic Extracting. *J. ACM* 16, 2 (apr 1969), 264–285. https://doi.org/10.1145/321510.321519

[9] Wafaa S. El-Kassas, Cherif R. Salama, Ahmed A. Rafea, and Hoda K. Mohamed. 2021. Automatic text summarization: A comprehensive survey. *Expert Systems with Applications* 165 (2021), 113679. https://doi.org/10.1016/j.eswa.2020.113679

[10] Günes Erkan and Dragomir R. Radev. 2004. LexRank: graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.* 22, 1 (dec 2004), 457–479.

[11] J. C. Gower. 1975. Generalized procrustes analysis. *Psychometrika* 40, 1 (01 Mar 1975), 33–51. https://doi.org/10.1007/BF02291478

[12] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning Word Vectors for 157 Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association (ELRA). https://aclanthology.org/L18-1550

[13] Vishal Gupta and Gurpreet Lehal. 2010. A Survey of Text Summarization Extractive Techniques. *Journal of Emerging Technologies in Web Intelligence* 2 (08 2010). https://doi.org/10.4304/jetwi.2.3.258-268

[14] Md. Majharul Haque, Suraiya Pervin, and Zerina Begum. 2015. Automatic Bengali news documents summarization by introducing sentence frequency and clustering. In *2015 18th International Conference on Computer and Information Technology (ICCIT)*. 156–160. https://doi.org/10.1109/ICCITechn.2015.7488060

[15] Felix Hausdorff. 1914. *Grundzüge der mengenlehre*. Vol. 7. von Veit.

[16] Aditya Jain, Divij Bhatia, and Manish K Thakur. 2017. Extractive Text Summarization Using Word Vector Embedding. In *2017 International Conference on Machine Learning and Data Science (MLDS)*. 51–55. https://doi.org/10.1109/MLDS.2017.12

[17] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*. Association for Computational Linguistics, 74–81. https://aclanthology.org/W04-1013

[18] Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 404–411. https://aclanthology.org/W04-3252

[19] G. Bharathi Mohan and R. Prasanna Kumar. 2022. A Comprehensive Survey on Topic Modeling in Text Summarization. In *Micro-Electronics and Telecommunication Engineering*. Springer Nature Singapore, 231–240.

[20] N. Moratanch and S. Chitrakala. 2016. A survey on abstractive text summarization. In *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*. 1–7. https://doi.org/10.1109/ICCPCT.2016.7530193

[21] N. Moratanch and S. Chitrakala. 2017. A survey on extractive text summarization. In *2017 International Conference on Computer, Communication and Signal Processing (ICCCSP)*. 1–6. https://doi.org/10.1109/ICCCSP.2017.7944061

[22] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank Citation Ranking : Bringing Order to the Web. In *The Web Conference*. https://api.semanticscholar.org/CorpusID:1508503

[23] Sohini Roychowdhury, Kamal Sarkar, and Arka Maji. 2022. Unsupervised Bengali Text Summarization Using Sentence Embedding and Spectral Clustering. In *Proceedings of the 19th International Conference on Natural Language Processing (ICON)*. Association for Computational Linguistics, 337–346. https://aclanthology.org/2022.icon-main.40

[24] Y. Rubner, C. Tomasi, and L.J. Guibas. 1998. A metric for distributions with applications to image databases. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*. 59–66. https://doi.org/10.1109/ICCV.1998.710701

[25] G. Salton, A. Wong, and C. S. Yang. 1975. A vector space model for automatic indexing. *Commun. ACM* 18, 11 (nov 1975), 613–620. https://doi.org/10.1145/361219.361220

[26] Kamal Sarkar. 2012. An approach to summarizing Bengali news documents. In *Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI '12)*. Association for Computing Machinery, 857–862. https://doi.org/10.1145/2345396.2345535

[27] Kamal Sarkar. 2012. Bengali text summarization by sentence extraction. *CoRR* abs/1201.2240 (2012). http://arxiv.org/abs/1201.2240

[28] Oguzhan Tas and Farzad Kiyani. 2017. A SURVEY AUTOMATIC TEXT SUMMARIZATION. *PressAcademia Procedia* 5, 1 (2017), 205–213. https://doi.org/10.17261/Pressacademia.2017.591

[29] Ulrike von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and Computing* 17, 4 (01 Dec 2007), 395–416. https://doi.org/10.1007/s11222-007-9033-z

[30] Adhika Pramita Widyassari, Supriadi Rustad, Guruh Fajar Shidik, Edi Noersasongko, Abdul Syukur, Affandy Affandy, and De Rosal Ignatius Moses Setiadi. 2022. Review of automatic text summarization techniques & methods. *Journal of King Saud University - Computer and Information Sciences* 34, 4 (2022), 1029–1046. https://doi.org/10.1016/j.jksuci.2020.05.006

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009