Extractive Text Summarization Using Word Similarity-Based Spectral Clustering

Fahim Morshed, Md. Abdur Rahman, Sumon Ahmed

October 23, 2024

Abstract

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 based method that had no understanding of inter-sentence relationship or used similarity measures that didn't express the relation 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 of 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.

1 Introduction

Text Summarization is the process of shortening a larger text without losing any key information. But manually summarizing longer texts is really time-consuming and counter-productive. So developing an Automatic Text Summarization (ATS) system that would process larger documents and summarize them automatically is really necessary [31]. This is a major area of natural language processing (NLP) research which is progressing very rapidly. This task has become more and more important in the current digital age, where the amount of textual data grows exponentially in many fields [17], such as news, legal documents, health reports, research papers, and social media content. ATS techniques allow users to quickly get the essential information without needing to read through large amounts of text [7]. ATS is used in many 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. It can also play a critical role in personal assistants and chatbots, providing condensed information to users quickly and efficiently [29].

There are two main types of automatic text summarization: extractive and abstractive [29]. Extractive summarization, which is the focus of this paper, works by selecting the best sentences or phrases directly from the source document, maintaining the original wording and sentence structure [21]. In contrast, abstractive summarization involves generating new sentences to capture the meaning of the text, Similar to human made summarization [20]. For text

summarization, extractive methods are widely used because of its simplicity and effectiveness, especially for languages with limited NLP resources [12].

There are a lot of different ways to achieve extractive summarization. A commonly used method for extractive text summarization is graph-based summarization [7]. This method represents the sentences of a document as nodes of a graph, and the edges between them are weighted by the similarity between the sentences [7]. Popular algorithms like LexRank [8] and TextRank [18] build graphs based on cosine similarity between sentences and apply ranking algorithms such as PageRank [22] and Random Walk to determine which sentences are the most important. These sentences are then selected to make the summary. Graph-based methods offer a robust way to capture sentence importance and relationship, ensuring that the extracted summary covers the key information while minimizing redundancy [7].

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, 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 that only the most representative sentence is selected. This method is effective in summarization of documents with multiple topics or subtopics, as it allows the summary to touch on each area without being repetitive.

For Bengali, a low-resource language, early attempts at text summarization relied on traditional methods like TF-IDF (Term Frequency-Inverse Document Frequency) scoring [1, 4, 27, 28]. These approaches, while simple, faced challenges in capturing the true meaning of sentences, as they treated words as isolated terms [29]. Graph-based methods improved summarization quality by incorporating sentence similarity, but they were still limited by the quality of the embeddings used for the Bengali language. With the advent of word embedding models like FastText [11], 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 [24], encountered sentence-similarity calculation issues when averaging word vectors to represent sentence meaning. This method failed in most similarity calculation cases cases because words in a sentence are often complementary rather than being similar, leading to inaccurate sentence representations when averaging their vectors. As a result, word-to-word relationships between sentences were lost, reducing the effectiveness of the method.

In this paper, we propose a new approach to address these challenges. Our method improves upon previous attempts [24] by focusing on the individual similarity between words in sentences rather than averaging word vectors. Here, we followed a number of steps to get the similarity between two sentences. At first, for each of a sentence, we picked the closest word vector from the other sentence. Then, we took the Gaussian Similarity for these two vectors. We did this for every word of both sentence. We take the geometric mean of these similarities to get the sentence similarity between the two sentences. By applying Gaussian similarity to the Most Similar Word Distance (D_{msw}) values, we build an affinity matrix that better reflects sentence closeness. Then, we applied spectral clustering on this matrix to group similar sentences together and used TF-IDF to select the most representative sentences from each cluster. This approach reduces redundancy and improves the quality of the summary by selecting sentences that are not only relevant but also diverse. This method works well for Bengali on four diverse datasets (Figure 4). It consistently outperforms other graph-based methods like BenSumm [3], SASbSC [24], LexRank [8]. It also performs similarly well in other low resource languages such as Hindi, Marathi and Turkish (Table 3). These are the only other low resource languages where

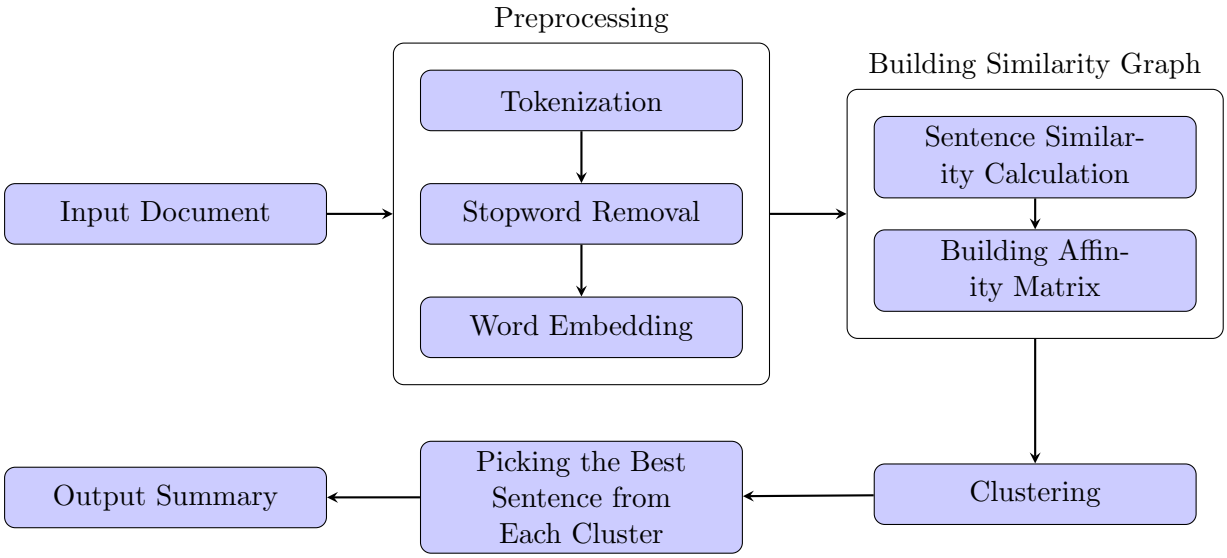


Figure 1: Process Flow Diagram

we found reliable evaluation datasets and tested our model on them. The search process was not exhaustive due to our language barrier. The whole process of summarization is shown in the Process Flow Diagram (Figure 1)

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 Literature review and Methodology are described in section 2 and 3 respectively. The section 4 illustrates the findings of this work. The section 5 discusses the findings of the paper in more depth, and section 6 concludes the paper.

2 Literature Review

Text Summarization has been an important necessity for textual data consumption for a long time. So automating the Text Summarization process has been a research problem for a long time. Attempts at automatic text summarization started with indexing-based methods [2]. In this attempt Baxendale [2] attempted to summarize text by scoring sentences higher based on a certain word list. But this type of method failed to capture the topic and essence of the input text. To solve this, Text Summarization with statistical methods like TF-IDF became very popular. Edmundson [6] proposed a method which can focus on the central topic of a document. 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 corpus) to calculate the importance of a term in a document. This method identifies the words that are common in the input text but not as common in the language and identifying them as the central topic. But it was too error-prone due to it thinking every word as a unique isolated term and not having any semantic relation with other words. Some words may be a central

topic of a document but not identified as such because they got divided into too many synonyms.

Modern breakthroughs into the extractive text summarization began with the usage of Graph-based Extractive Text Summarization methods like LexRank [8] or TextRank [18]. LexRank [8] calculates the similarity between two sentences using cosine similarity and builds a graph containing similarity between every pair of sentences in the input. The most important sentences are then identified using the PageRank [22] algorithm on the graph. This algorithm ranked the sentences, who are most similar with other high ranked sentences, higher. TextRank [18] also uses a similar approach, but for every sentence, the method distributed its scores to its neighbours using a random walk. The process was done over and over until the scores converge. Although these models are very novel compared to their time, they still lacked fundamental understanding of the words involved in a sentence.

To solve this problem by better expressing the similarity between words, a mathematical abstraction called Word Vector Embedding was conceptualized by the seminal work of Salton et al. [26]. Word Vector Space is a mathematical abstraction of a vocabulary where the closer two words are semantically, the closer they are in the vector space. Using word vector for summarization has only been started to be attempted recently [15].

But Text Summarization attempts in Bengali are a more recent development than in other high resource languages. So, a lot of sophisticated approaches from other languages haven't been attempted yet. Earlier Extractive methods have been focused on some derivative of TF-IDF based text summarization such as Chowdhury et al. [3], Dash et al. [4], Sarkar [27]. Sarkar [27] used simple TF-IDF score of each sentence to rank them and pick the best sentences. Dash et al. [4] used weighted TF-IDF along with some other features like sentence position to rank the sentences. Chowdhury et al. [3] 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 model is that TF-IDF matrix is not semantically equivalent to the actual sentences. So it didn't perfectly represent the sentences' semantic closeness in the graph. Using Word Vector Embedding for Bengali has solved this problem. FastText [11] released a dataset[1] that had word vector embedding in 157 languages, including Bengali. Using this dataset, Roychowdhury et al. [24] proposed a model where they replaced all the words with their respective vector, then averaged the vectors in a sentence to get the vector for a sentence. The Gaussian Similarity between the vectors is used to build the graph. On the graph, spectral clustering was used to group them together and pick one sentence from each cluster using cosine similarity to get the summary.

But this model suffers critically from sentence similarity calculation. Words in a sentence do not have similar meaning, instead they express different parts of one whole meaning of a sentence. Which means they are complementary instead of being similar. So word averages always tend to be in the center and don't represent the semantic similarity anymore because the word vectors get scattered throughout the vector space due to this complementary nature. An example is shown in Figure 2 where the distance between the average word vectors is being misleading. In the figure, each point represents a word vector. The words from the same sentence are grouped together by being colored the same. In Figure 2(a), a scenario is shown in which the words of the two sentences are closer together in a vector space. The average distance between these two sentences can be seen in the Figure 2(c). We can see that averaging the words made both of the average clusters in the center. In Figure 2(b), we can see a different scenario where the word vectors are farther apart meaning wise. But the Figure 2(d) shows the average vector for these two sentences is closer than in the first scenario, thus making this metric misleading. This

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

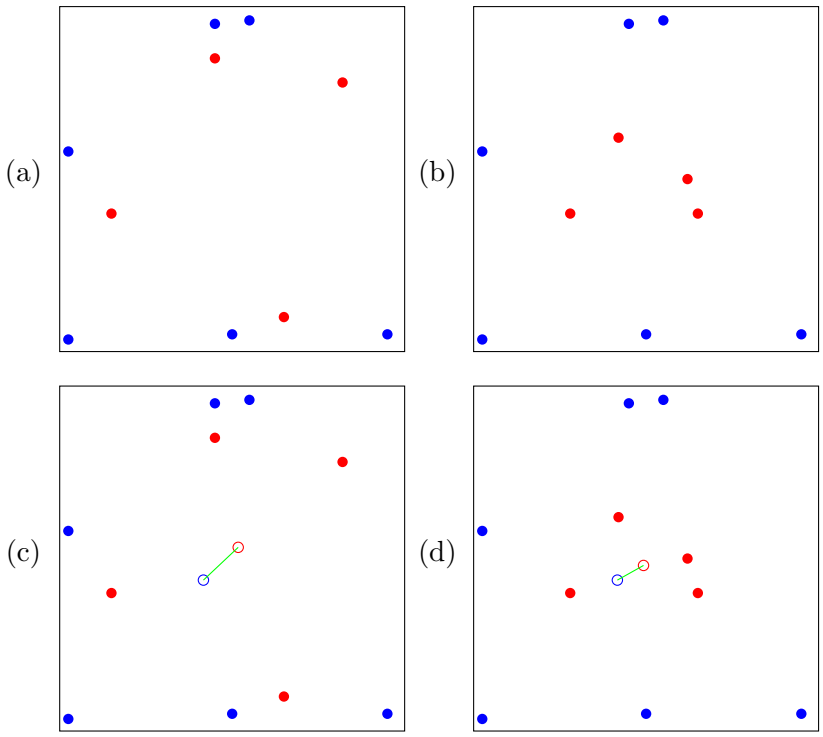


Figure 2: Scenarios where averaging method fails.

shortcoming 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 [24]. 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 of the sentences in a cluster to pick the highest ranked and thus most representative sentence from the cluster [1, 4, 27, 28]. The summarization process followed here involves 4 steps. These are, Pre-processing, Sentence similarity calculation, Clustering and Summary generation. These steps are 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 preprocessing step, the input 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 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. To remove these stop words, we used a list[2] of 363 bengali 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 mathmatical way. We used the FastText dataset[3] [11] with 1.47 million Bengali words and their vector representation to achieve this step. 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 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 relation between the sentences in the input document. This graphical representation is expressed via an affinity matrix to cluster the sentences into groups of 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 to graphically represent the input document. Here, we proposed a novel sentence similarity calculation technique using individual Gaussian similarity between close word pairs to construct the 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 two words from this pair is defined as the Most Similar Word Distance (D_{msw}) to be used in the following steps. 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. In this equation 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 word correspondence between the two sentences. Here, the word similarity is calculated using Gaussian kernel function for the elements of the set D because Gaussian kernel functions provides a smooth, flexible and most representative similarity between two vectors [**?**]. The process of calculating word similarity (W_{sim}) is given in the following 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 [**?**]. The value of sigma was

[2] *https://www.ranks.nl/stopwords/bengali*
[3] *https://fasttext.cc/docs/en/crawl-vectors.html*

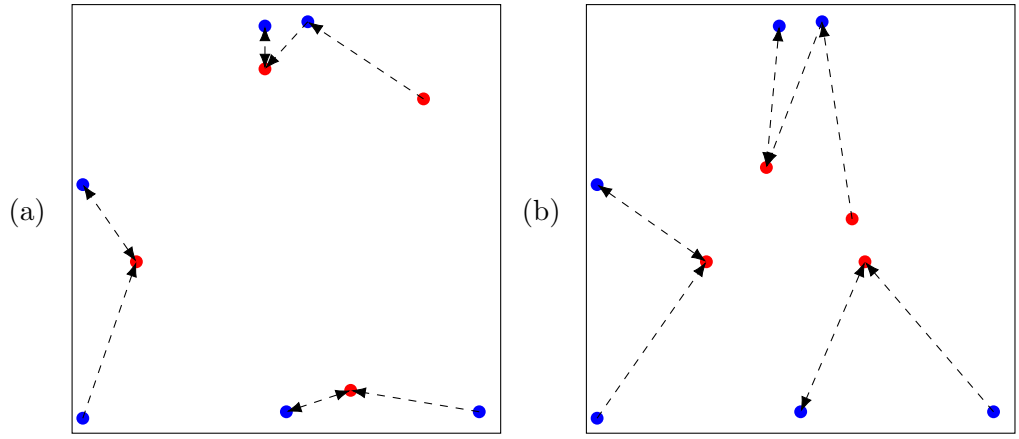


Figure 3: Process of obtaining D_{msw}

fine-tuned to be 5×10^{-11} which gives the similarity measurement. The process of fine-tuning is described in the experimentation section (section 4.5.1).

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

Finally, the sentence similarity between the two sentences $Sim(X, Y)$ is calculated using geometric mean of the word similarities from the above step to construct an affinity matrix. Using geometric mean makes the similarity value less prone to effects of outliers to make the calculation more reliable. This process is shown in the following equation 4. In this equation, 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.

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

By following steps described above, we get a similarity value for two sentence. This value solves the lack of local word correspondence problem faced by the word averaging method based similarity calculation [24] by considering local word to word similarity.
The standard deviation σ in the Equation 4 was fine-tuned to be 5×10^{-11} where it gave the best results (Figure 5). The process of building the affinity matrix, A, is described in the Algorithm 1. Here, line 9–13 and 19–23 are the process of getting D_{msw}. Line 7–26 describes the process of getting $\sum_{i=1}^{n} D_i^2$. Line 4–29 describes the process of getting $Sim(Sentence_i, Sentence_j)$. Line 1–31 describes the process of getting the affinity matrix, A.

3.3 Clustering

The clustering is the most integral part of this summarization technique, aiming to group all the sentences with similar meanings together. Here, spectral clustering is used to cluster the sentences using sentence similarity calculated in the step above. Spectral clustering was chosen

Algorithm 1 Sentence Similarity Calculation

$n \leftarrow$ length(WordVectorList)
$A \leftarrow \{\{0\} \times n\} \times n$
for each sentence$_i$ in WordVectorList **do**
    $D_{\text{Square}} \leftarrow 0$
    count $\leftarrow 0$
    for each sentence$_j$ in WordVectorList **do**
        for each word$_i$ in sentence$_i$ **do**
            $D_{\text{msw}} \leftarrow \infty$
            for each word$_j$ in sentence$_j$ **do**
                if Distance(word$_i$, word$_j$) $< D_{\text{msw}}$ **then**
                $D_{\text{msw}} \leftarrow$ Distance(word$_i$, word$_j$)
                end if
            end for
            $D_{\text{Square}} \leftarrow D_{\text{Square}} + D_{\text{msw}}^2$
            count++
        end for
        for each word$_j$ in sentence$_j$ **do**
            $D_{\text{msw}} \leftarrow \infty$
            for each word$_i$ in sentence$_i$ **do**
                if Distance(word$_i$, word$_j$) $< D_{\text{msw}}$ **then**
                $D_{\text{msw}} \leftarrow$ Distance(word$_i$, word$_j$)
                end if
            end for
            $D_{\text{Square}} \leftarrow D_{\text{Square}} + D_{\text{msw}}^2$
            count++
        end for
        similarity $\leftarrow \exp\left(\frac{-D_{\text{Square}}}{2 \times \text{count} \times \sigma^2}\right)$
        $A[i][j] \leftarrow A[j][i] \leftarrow$ similarity
    end for
end for
Return A

here because Roychowdhury et al. [24] found it to be better performing than DBSCAN method. The spectral clustering steps were followed according to the tutorial given by [30].

To perform spectral clustering on a data, firstly, an affinity matrix is required that shows the weight of edges between the vertexes in the graph. Here the affinity A is prepared using the following Equation 5.

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

Here, S_i, S_j are sentences from the input document. The affinity matrix, A, is used in the spectral clustering which is implemented using SciKit-learn library [23] of python. It is also necessary to provide the number of clusters to achieve. The number of clusters is fixed at $k = ceiling\left(\frac{N}{5}\right)$ due to it being a reasonable size to contain all necessary sentences as well as being short enough to be an effective summary.

3.4 Summary Generation

Summarized text is the collection of selected sentences from different clusters. After clustering, we pick one sentence from each cluster. The sentences inside a cluster are ranked among themselves using TF-IDF techniques. From each cluster, the sentence with the most TF-IDF score is selected. We then rearranged these picked sentences are in their order of appearance to retain the normal flow of information in the input. These sentences are then concatenated together to produce the final output summary. The clustering and summary generation process is shown in Algorithm 2. After clustering in the lines 1 and 2, We ranked the sentences in the lines 3–8. The best sentence indexes are picked in the lines 9–11. The summary is generated in the lines 12–16.

Algorithm 2 Summary Generation

```
1: k ← ⌈ length(A) / 5 ⌉
2: clusters ← spectral_clustering(adjacency = A, k)
3: indexes ← {}
4: for each cluster_i in clusters do
5:     TFIDF ← {}
6:     for each index in cluster_i do
7:         TFIDF.append(tfidf_sum(sentences[index]))
8:     end for
9:     indexes.append(indexof(max(TFIDF)))
10: end for
11: sort(indexes)
12: S ← ""
13: for each i in indexes do
14:     S ← S+ sentences[i]
15: end for
16: Return S
```

4 Result

The text summarization performance of the proposed model is compared against the BenSumm [3], LexRank [8] and SASbSC [24] methods. These methods are the recently published state of the art model for Bengali Extractive Text Summarization. A classic extractive text summarizing method LexRank [8] was also used as a benchmark for comparison.

4.1 Evaluation Datasets

To examine our proposed model, we compared our model with three benchmark models on four different datasets. Multiple datasets are used to examine the effectiveness of the text summarization models to avoid biased result due to any problem with the dataset.

4.1.1 Dataset-1 (Self-curated)

To evaluate the performance of implemented text summarization methods with existing works [3, 8, 24], a curated Bengali extractive text summarization dataset was produced by an expert linguistic team. 250 news documents of various sizes were summarized for this purpose. Each document was summarized twice by two different person to minimize human bias. In total, there is 500 different document-summary pair in this dataset. This dataset is made publicly available[4] for other researchers to use for evaluation purpose in their research.

4.1.2 Dataset-2 (Towhid Ahmed Foysal)[9]

This dataset is a collection of summary article pair from The Daily Prothom Alo. It was published by Towhid Ahmed Foysal in Kaggle[5]. The original dataset was filtered so that all the articles were smaller than 50 characters, and all the summaries that contain something not in the original articles were discarded. After filtering, a total of 10,204 articles remained, each with two summaries.

4.1.3 Dataset-3 (BNLPC)[13]

This dataset is a collection of news article summaries published by Haque et al. [13]. The dataset was collected from GitHub[6]. The dataset contains one hundred articles with three different summaries for each article.

4.1.4 Dataset-4 (Abid Mahdi)

This dataset was published by Abid Mahdi on GitHub[7]. 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 to ensure its quality.

4.2 Text Summarization Models

Four different Bengali extractive text summarization models were implemented to evaluate them by comparing the machine generated summaries against the human generated summaries from the datasets described above.

Model-1: This is the proposed model for this research. The model uses word vector-based Gaussian similarity to perform spectral clustering for grouping similar sentences together and extract one sentence from each group. This is described as Word Similarity-based Spectral Clustering (WSbSC)

Model-2: Model-2 (SASbSC) is the method proposed by Roychowdhury et al. [24]. This method is similar to the proposed method as both methods use word vector embedding and spectral clustering to generate summaries. It uses a sentence center similarity-based graph

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

for spectral clustering. Then it uses cosine similarity to extract sentences from each cluster. SCSbSC averages all the word vectors of a particular sentence to get the Sentence center and calculates sentence similarity based on gaussian similarity of those average vectors. This method was implemented in python as described in their article.

Model-3: BenSumm describes two different summarization methods in the study [3]. But only the extractive method of their paper is implemented and compared with the proposed method because it is also an extractive summarizer. BenSumm implements a TF-IDF based cosine similarity graph between the sentences and then clusters the sentences using Agglomerative Clustering. The implementation codes are publicly available in GitHub[8].

Model-4: LexRank [8] 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 a python package in PyPI as LexRank[9]. LexRank is implemented using a large Bengali Wikipedia corpus[10].

4.3 Evaluation Metrics

To evaluate the correctness of the machine generated summaries compared to the human generated summaries, we used the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) method [16]. It compares two text blocks, a human produced reference summary and a machine generated summary. The ROUGE method uses N-gram-based overlapping to find a precision, recall and F-1 score. The Rouge python package[11] is used as the implementation to calculate ROUGE scores. There are three different metrics in the package for comparison of the summaries which are:

1. **ROUGE-1:** It uses unigram matching to find how much similar two summaries are. It calculates total common characters and is a good performance indicator. But it can also be misleading too as many large enough texts will share a very high proportion of uni-grams between them.

2. **ROUGE-2:** It uses bi-gram matching to find how much similar the two summaries are in a word level. Shared bigrams lead to a deeper analysis of syntactic similarities between the two summaries.

3. **ROUGE-LCS:** It finds the longest common sub-sequence between the summaries to calculate the rouge scores. It can calculate the similarity in flow of the sentences between two summaries.

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

4.4 Comparison

Average F-1 scores for the three Rouge metrics (Rouge-1, Rouge-2, Rouge-LCS) of the four models (Proposed (WSbSC), SASbSC, BenSumm, LexRank) on the four datasets are shown in the table 1. We can see that our proposed model performs 11.9%, 24.1% and 16.2% better than the closest method (SASbSC) in Rouge-1, Rouge-2 and Rouge-LCS respectively on our self-curated dataset. It performs 68.9%, 95.4% and 84.6% better than the next closest method (BenSumm) on Dataset-2. It performs a tie in R-1, 3% better in R-2 and 2.6% better than the

[8] *https://github.com/tafseer-nayeem/BengaliSummarization*
[9] *https://pypi.org/project/lexrank/*
[10] *https://www.kaggle.com/datasets/shazol/bangla-wikipedia-corpus*
[11] *https://pypi.org/project/rouge/*

Dataset-1 (SC)			
Model	Rouge-1	Rouge-2	Rouge-LCS
Model-1 (WSbSC)(Proposed)	**0.47**	**0.36**	**0.43**
Model-2 (BenSumm) [3]	0.41	0.29	0.36
Model-3 (SASbSC) [24]	0.42	0.29	0.37
Model-4 (LexRank) [8]	0.22	0.14	0.20
Dataset-2 (TAF)			
Model-1 (WSbSC)(Proposed)	**0.49**	**0.43**	**0.48**
Model-2 (BenSumm) [3]	0.29	0.22	0.26
Model-3 (SASbSC) [24]	0.23	0.12	0.18
Model-4 (LexRank) [8]	0.24	0.16	0.22
Dataset-3 (BNLPC)			
Model-1 (WSbSC)(Proposed)	**0.41**	**0.34**	**0.40**
Model-2 (BenSumm) [3]	0.36	0.28	0.34
Model-3 (SASbSC) [24]	**0.41**	0.33	0.39
Model-4 (LexRank) [8]	0.26	0.19	0.24
Dataset-4 (AM)			
Model-1 (WSbSC)(Proposed)	**0.49**	**0.41**	**0.47**
Model-2 (BenSumm) [3]	0.31	0.22	0.28
Model-3 (SASbSC) [24]	0.30	0.18	0.24
Model-4 (LexRank) [8]	0.22	0.14	0.20

Table 1: Comparison of average Rouge scores between graph based extractive summarization models on 4 different datasets

closest method (SASbSC) on R-LCS using the BNLPC dataset. It performs 58%, 86.4%, and 67.9% better than the closest method (BenSumm) on Dataset-4 in all three metrics.

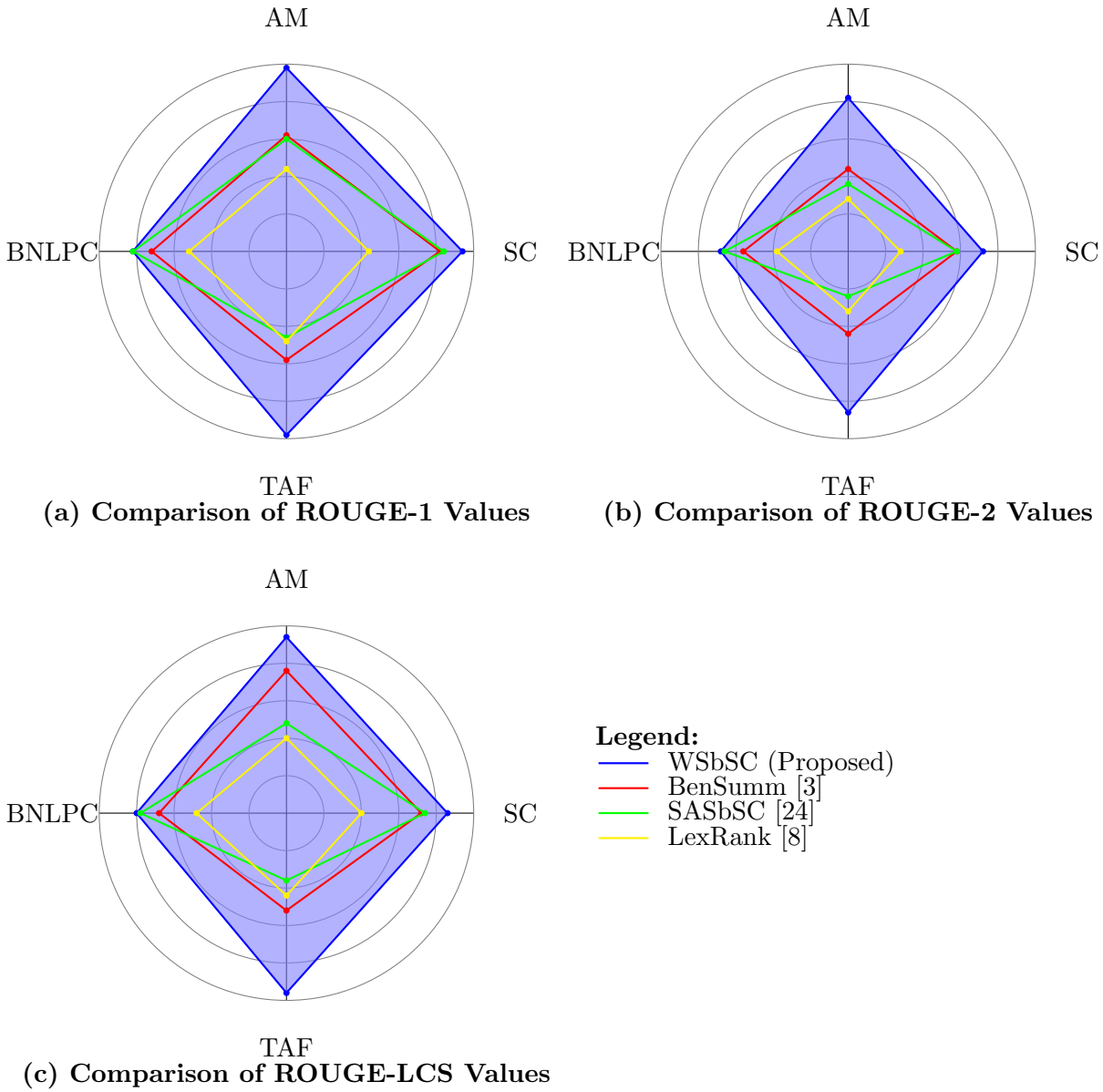
These results are further visualized into three radar charts, so that the performance of each model on the four datasets can be visualized at once.
These charts (Figure 4) show us that the proposed method is much more dataset independent and performs uniformly on every metric across the datasets. Other models, although perform good on certain datasets, fail to show consistency. For example, Both BenSumm and SASbSC perform well on Dataset-1 and Dataset-3, but the performances fall sharply on Dataset-2 and Dataset-4.

4.5 Experimentation

We experimented on our model with different ranking techniques and different values for standard deviation (σ) on Equation 4 to get the best rouge values for a summary. The standard deviation (σ) for the Gaussian Similarity represents a smoothing factor that can be used as a control variable to be fine-tuned for the best result. On the other hand, ranking methods pick the most representative sentence from a cluster after the clustering step. We checked First Rank and TF-IDF Rank methods for ranking. These experimentations are discussed with more detail below.

4.5.1 Fine-tuning Standard Deviation (σ)

We checked for different Standard Deviation (σ) on Equation 4. We checked for sixty-three different values for σ from 10^{-12} to 10 on regular intervals and found that 5×10^{-11} works best as the value for σ on our self-curated dataset (dataset-1). The result for the fine-tuning process is shown in the following line chart (Figure 5).



(a) Comparison of ROUGE-1 Values

(b) Comparison of ROUGE-2 Values

(c) Comparison of ROUGE-LCS Values

Figure 4: The Radar chart of the models of being compared on four datasets at once

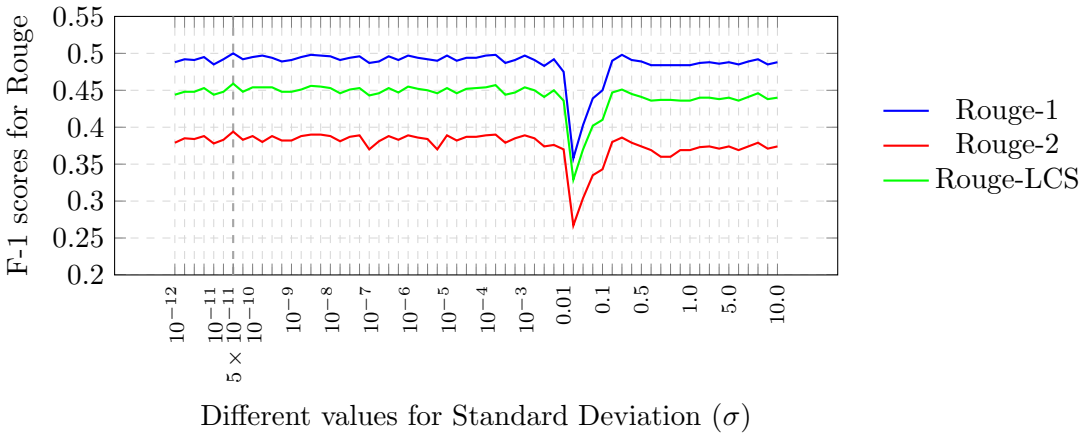


Figure 5: Fine-tuning for different standard deviation (σ) values

Method	Rouge-1	Rouge-2	Rouge-LCS
FirstRank	0.47	0.36	0.43
TF-IDF	**0.50**	**0.40**	**0.46**

Table 2: Comparison of Result of different ranking techniques

Language	Rouge-1	Rouge-2	Rouge-LCS
Bengali (Dataset - 1)	0.47	0.36	0.43
Bengali (Dataset - 2)	0.49	0.43	0.48
Bengali (Dataset - 3)	0.41	0.34	0.40
Bengali (Dataset - 4)	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

4.5.2 Different Ranking Techniques Inside Clusters

We implemented two ranking methods to pick the best sentence from each cluster. The first one is the First Rank method where we just pick the sentence that appears first in the input document. The second one is the TF-IDF ranking, where we ranked the sentences by their TF-IDF scores and pick the best one. We can see in the table 2 that the TF-IDF performs better on a high quality dataset like our Self-curated one.

4.6 Implementation Into Other Languages

The proposed model is not language-dependent, thus it can be extended into other languages. To perform this method into a language, we only need a language-specific tokenizer, a list of stop-words and a word vector embedding dataset. We tried to find quality extractive text summarization dataset for evaluating the method, but could only find relevant datasets in three other languages i.e., Hindi, Marathi and Turkish. We adopted this Model into these three low resource languages to check this hypothesis.

The Table-3 shows the result of the proposed word similarity based spectral clustering method for extractive summarization in other low resource languages. For the Hindi language, we used a Kaggle dataset[12] produced by Gaurav Arora. For the Marathi language, we used another Kaggle dataset[13] produced by Ketki Nirantar. For the Turkish language, we used a GitHub dataset[14] produced by the XTINGE [5] team. We can see that the results on Marathi and Turkish are slightly better than the result on Bengali. Although it performs slightly lower on Hindi, The score is still similar to Bengali.

5 Discussion

The results presented in the previous sections highlight the effectiveness of the proposed WSbSC model for extractive text summarization in Bengali, as well as its adaptability to other

[12] *https://www.kaggle.com/datasets/disisbig/hindi-text-short-and-large-summarization-corpus/*

[13] *https://www.kaggle.com/datasets/ketki19/marathi*

[14] *https://github.com/xtinge/turkish-extractive-summarization-dataset/blob/main/dataset/XTINGE-SUM_TR_EXT/xtinge-sum_tr_ext.json*

low-resource languages. This section delves into an analysis of the comparative results, the strengths and limitations of the proposed method, and potential areas for further research.

As evidenced by the results shown in Table 1 and Figure 4, the WSbSC model consistently outperforms the baseline models, namely BenSumm [4], LexRank [8], and SASbSC [24], across multiple datasets. This performance improvement is largely for the novel approach of calculating sentence similarity. Taking the geometric means of individual word similarities overcomes the problems of averaging vector method. The Gaussian similarity-based approach used in WSbSC provides a more novel and precise method for capturing the semantic relationships between sentences.

Our proposed strategy is more suited for the sentence similarity calculation than other strategies that can compare two sets of vectors such as Earth Movers Distance (EMD) [25], Hausdorff Distance [14], Procrustes Analysis [10]. EMD [25] tries to find the lowest amount of "work" needed to transform one set into the other one. It considers adding a new point to a set, removing a point from a set, scaling the whole set, moving a point, rotating the set etc. as "work". This is very computationally expensive as hundreds of separate possibilities have to be checked for each point in each set. And it also focuses on scaling and rotating, which are not relevant in word vector space. Another method, Hausdorff distance [14] takes the worst case scenario and calculates the farthest distance between two points in the two set. It is easily influenced by outliers. It was avoided because words tend to spread out over the whole word space and this would suffer from the same problem as the averaging method. Procrustes Analysis [10] tries to find the lowest amount of misalignment after scaling and rotating the two sets. Both of these processes are irrelevant in the context of word vector.

On the other hand, the proposed method focuses on Local Point Correspondence between two sets which is more important for words. The Gaussian similarity function captures the proximity of points smoothly, providing continuous feedback on how similar two points are in a normalized way. It is also robust against small outliers because of the use of a soft similarity measure (Gaussian) and geometric mean which helps smooth over small differences in word similarities.

One of the key strengths of this proposed method is the reduction of redundancy, which is a common issue in extractive summarization methods. By grouping sentences with similar meanings and selecting a representative sentence from each group, the model ensures that the summary covers a broad range of topics without repeating itself. The use of Spectral clustering is well-suited for the clustering task too because it does not assume a specific cluster shape and can infer the number of clusters using the Eigen gap method. Our proposed model also has an improved sentence similarity calculation technique. Using the geometric mean of individual word similarities offers a more precise measure of how closely two sentences are related. This is a marked improvement over traditional methods that rely on word averaging, which often dilute the semantic meaning of a sentence. Another key strength is that it is found to be scalable across languages. By requiring only a language-specific tokenizer, stop-word list, and word embedding dataset, WSbSC can be easily adapted to other languages, as demonstrated in the experiments with Hindi, Marathi, and Turkish datasets (Table 3). This makes the model highly versatile and valuable for extractive summarization in low-resource languages.

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 [11] dataset used here is heavily reliant on wikipedia for training. Which could introduce some small unforeseen biases. In cases where the word em-

beddings do not fully have some word of a given document, the model's performance could degrade as it leaves those words out. The model also does not take into account the order in which words appear in a sentence or when the form special noun or verb groups. So it can be a little naive in some highly specialized fields.

The WSbSC model has demonstrated its ability to perform well in low-resource languages such as Hindi, Marathi, and Turkish. Despite differences in language structure, the model's core methodology remained effective, yielding results that were consistent with the Bengali dataset evaluations. This underscores the potential of WSbSC as a generalizable approach for extractive summarization across different languages, if appropriate pre-processing tools and word embedding datasets are available.

The proposed Word Similarity-based Spectral Clustering model represents a significant advancement in Bengali extractive text summarization. Its ability to accurately capture sentence similarity, reduce redundancy, and generalize across languages makes it a valuable tool for summarizing text in low-resource 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 and evaluated a Word Similarity-based Spectral Clustering (WSbSC) method for Bengali extractive text summarization. The method uses semantic relationships between words to identify the best sentences from a text, addressing the need for effective summarization techniques in the Bengali language, which remains underrepresented in natural language processing research. By using spectral clustering, we aimed to group sentences based on their semantic similarity, improving the coherence and relevance of the generated summaries.

Through extensive experimentation on different Bengali summarization datasets, our results showed that the WSbSC method outperforms several baseline techniques, particularly in grouping the sentences into key topics of documents. Despite these promising results, there are areas of further improvement. One limitation observed is that the method may struggle with highly specialized or domain-specific texts, where deeper linguistic features beyond word similarity could be considered. Future work could explore hybrid models that integrate other post-processing techniques to improve 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 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. 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)*, 2017. doi: 10.1109/ICIVPR.2017.7890883.

[2] P. B. Baxendale. Machine-made index for technical literature—an experiment. *IBM Journal of Research and Development*, 2(4):354–361, 1958. doi: 10.1147/rd.24.0354.

[3] Radia Rayan Chowdhury, Mir Tafseer Nayeem, Tahsin Tasnim Mim, Md. Saifur Rahman Chowdhury, and Taufiqul Jannat. 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*, pages 2612–2619. Association for Computational Linguistics, April 2021. doi: 10.18653/v1/2021.eacl-main.224.

[4] Satya Ranjan Dash, Pubali Guha, Debasish Kumar Mallick, and Shantipriya Parida. Summarizing bengali text: An extractive approach. In *Intelligent Data Engineering and Analytics*, pages 133–140. Springer Nature Singapore, 2022. ISBN 978-981-16-6624-7.

[5] İrem Demir, Emel Küpçü, and Alptekin Küpçü. Extractive summarization data sets generated with measurable analyses. In *Proceedings of the 32nd IEEE Conference on Signal Processing and Communications Applications*, 2024.

[6] H. P. Edmundson. New methods in automatic extracting. *J. ACM*, 16(2):264–285, apr 1969. ISSN 0004-5411. doi: 10.1145/321510.321519.

[7] Wafaa S. El-Kassas, Cherif R. Salama, Ahmed A. Rafea, and Hoda K. Mohamed. Automatic text summarization: A comprehensive survey. *Expert Systems with Applications*, 165:113679, 2021. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2020.113679.

[8] Günes Erkan and Dragomir R. Radev. Lexrank: graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479, dec 2004. ISSN 1076-9757.

[9] Towhid Ahmed Foysal. Bangla summarization dataset (prothom alo), 2023. URL `https://www.kaggle.com/datasets/towhidahmedfoysal/bangla-summarization-datasetprothom-alo`. Accessed: 2024-10-13.

[10] J. C. Gower. Generalized procrustes analysis. *Psychometrika*, 40(1):33–51, Mar 1975. ISSN 1860-0980. doi: 10.1007/BF02291478. URL `https://doi.org/10.1007/BF02291478`.

[11] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 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), May 2018. URL `https://aclanthology.org/L18-1550`.

[12] Vishal Gupta and Gurpreet Lehal. A survey of text summarization extractive techniques. *Journal of Emerging Technologies in Web Intelligence*, 2, 08 2010. doi: 10.4304/jetwi.2.3.258-268.

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

[14] Felix Hausdorff. *Grundzüge der mengenlehre*, volume 7. von Veit, 1914.

[15] Aditya Jain, Divij Bhatia, and Manish K Thakur. Extractive text summarization using word vector embedding. In *2017 International Conference on Machine Learning and Data Science (MLDS)*, pages 51–55, 2017. doi: 10.1109/MLDS.2017.12.

[16] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81. Association for Computational Linguistics, July 2004. URL `https://aclanthology.org/W04-1013`.

[17] Bernard Marr. Big data: 20 mind-boggling facts everyone must read. https://www.forbes.com/sites/bernardmarr/2015/09/30/big-data-20-mind-boggling-facts-everyone-must-read/, 2015. Accessed: 2024-10-12.

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

[19] G. Bharathi Mohan and R. Prasanna Kumar. A comprehensive survey on topic modeling in text summarization. In *Micro-Electronics and Telecommunication Engineering*, pages 231–240. Springer Nature Singapore, 2022. ISBN 978-981-16-8721-1.

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

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

[22] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking : Bringing order to the web. In *The Web Conference*, 1999. URL https://api.semanticscholar.org/CorpusID:1508503.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[24] Sohini Roychowdhury, Kamal Sarkar, and Arka Maji. Unsupervised Bengali text summarization using sentence embedding and spectral clustering. In *Proceedings of the 19th International Conference on Natural Language Processing (ICON)*, pages 337–346. Association for Computational Linguistics, December 2022. URL https://aclanthology.org/2022.icon-main.40.

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

[26] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, nov 1975. ISSN 0001-0782. doi: 10.1145/361219.361220. URL https://doi.org/10.1145/361219.361220.

[27] Kamal Sarkar. An approach to summarizing bengali news documents. In *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, ICACCI '12, page 857–862. Association for Computing Machinery, 2012. ISBN 9781450311960. doi: 10.1145/2345396.2345535.

[28] Kamal Sarkar. Bengali text summarization by sentence extraction. *CoRR*, abs/1201.2240, 2012. URL http://arxiv.org/abs/1201.2240.

[29] Oguzhan Tas and Farzad Kiyani. A survey automatic text summarization. *PressAcademia Procedia*, 5(1):205–213, 2017. doi: 10.17261/Pressacademia.2017.591.

[30] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, Dec 2007. ISSN 1573-1375. doi: 10.1007/s11222-007-9033-z. URL https://doi.org/10.1007/s11222-007-9033-z.

[31] Adhika Pramita Widyassari, Supriadi Rustad, Guruh Fajar Shidik, Edi Noersasongko, Abdul Syukur, Affandy Affandy, and De Rosal Ignatius Moses Setiadi. Review of automatic text summarization techniques & methods. *Journal of King Saud University - Computer and Information Sciences*, 34(4):1029–1046, 2022. ISSN 1319-1578. doi: https://doi.org/10.1016/j.jksuci.2020.05.006.