Extractive Text Summarization Using Word Similarity-Based Spectral Clustering

Fahim Morshed, Md. Abdur Rahman, Sumon Ahmed

October 28, 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 to increase the readability and information density of the input text 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 input texts reliably is really necessary to alleviate this manual labour [32]. ATS is a major area of rapidly progressing Natural Language Processing (NLP) research due to the importance of summarization increasing significantly. This increase can be attributed to the information overflow in the digital era due to the trend of exponential growth of textual data in last few years [19]. 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 efficiently get the essential information without needing to read through large amounts of texts by reducing the volume of text significantly [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 [30].

There are two main types of ATS: extractive and abstractive [30]. 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 [23]. In contrast, abstractive summarization involves synthesising new text that reflects on and has information from the input document but does not copy from it, similar to how a human summarizes a text [22]. 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 [14].

The key approach to extractive summarization is implementing a sentence selection method to classify which sentences belong in the summary. For this approach, 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, 29, 28] 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 more effective due to them using a sentence similarity graph in their workflow [9]. Graph based methods represent the sentences from the input document as nodes of a graph, and the semantic similarity between two sentences as the edge between the nodes [23]. Popular graph based algorithms like LexRank [10] and TextRank [20] build graphs based on cosine similarity between the bag-of-word vectors to build this graph.LexRank uses PageRank [24] 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 [21]. 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. 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 [1, 6, 28, 29] to score TF-IDF value for a sentence and use them to select the best scoring sentences to form the summary. These TF-IDF based approaches, while simple, faced challenges in capturing the true meaning of sentences, as they treated words as isolated terms so synonyms would get treated as completely different terms [30]. To solve this problem, graph-based methods were introduced in Bengali. Although graph-based methods improved summarization quality by incorporating sentence similarity, they were still limited by the quality of the embeddings used for the Bengali language. With the advent of word embedding models like FastText [13], 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 [25], encountered issues with sentence-

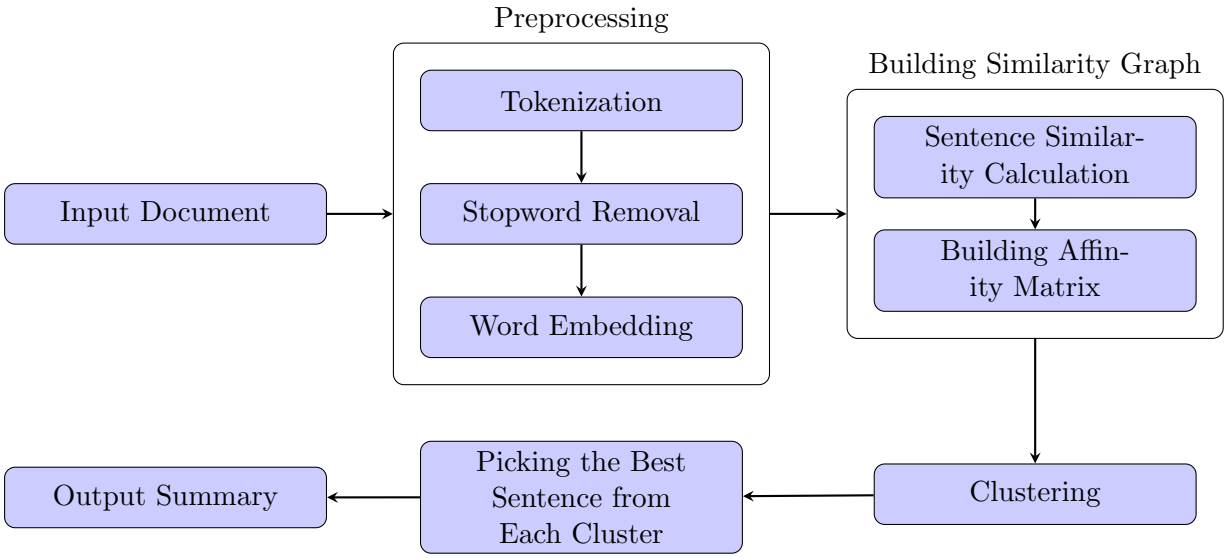


Figure 1: Process Flow Diagram

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 often complementary to each other 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 [25] 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 [5], SASbSC [25], LexRank [10]. 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 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 Related Work

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 [4]. In this attempt Baxendale [4] 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 [8] 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 [10] or TextRank [20]. LexRank [10] 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 [24] algorithm on the graph. This algorithm ranked the sentences, who are most similar with other high ranked sentences, higher. TextRank [20] 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. [27]. 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 [17].

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. [5], Dash et al. [6], Sarkar [28]. Sarkar [28] used simple TF-IDF score of each sentence to rank them and pick the best sentences. Dash et al. [6] used weighted TF-IDF along with some other 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 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 [13]

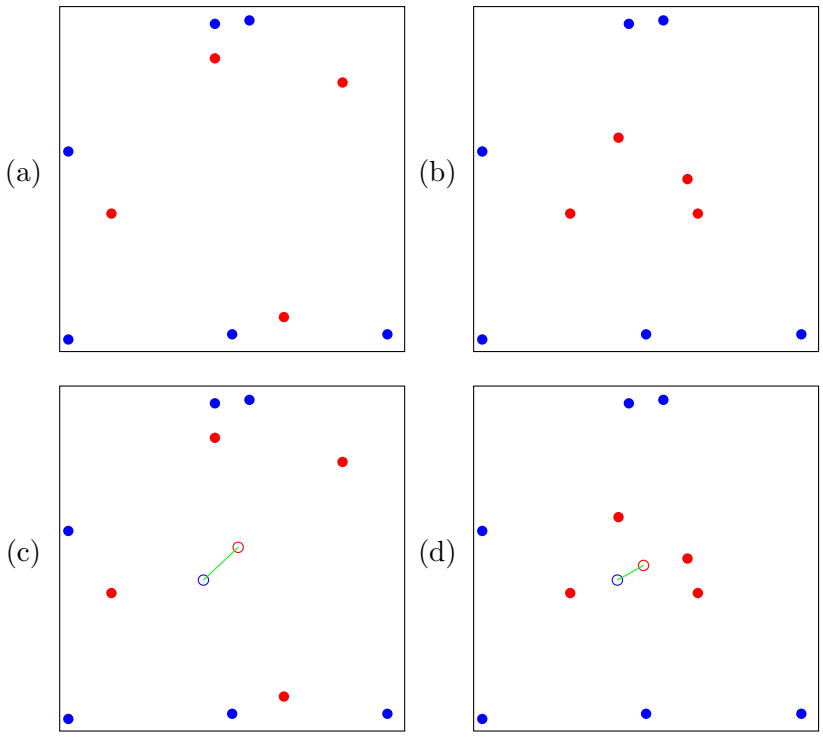


Figure 2: Scenarios where averaging method fails.

released a dataset[1] that had word vector embedding in 157 languages, including Bengali. Using this dataset, Roychowdhury et al. [25] 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 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

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

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 [25]. 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, 6, 28, 29]. 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 mathematical way. We used the FastText dataset[3] [13] 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 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 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

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

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 [3]. 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 [3]. The value of sigma was 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 sentences. This value solves the lack of local word correspondence problem faced by the word averaging based similarity calculation method [25] by considering local word to word similarity. This correspondence is shown in the figure 3 to demonstrate the merit of the method described above. Figure 3 follows up the scenario from the figure 2 to show that the proposed method can solve the local word correspondence problem faced by word averaging method. The figure shows that 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} from equation 3 larger due to the nature of Gaussian kernel function.

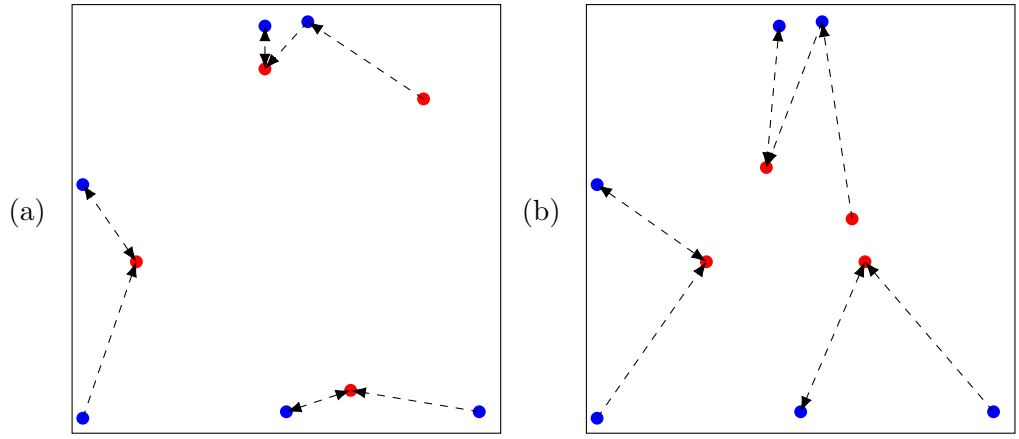


Figure 3: Local word correspondence of D_{msw} method. Dots in the figure represent word vectors and are coloured with the same colour if they are from the same sentence. Black dashed arrows represent the D_{msw} from its destination word vector. Here, scenario (a) will have larger similarity due to it having smaller D_{msw} than scenario (b)

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 2 where the scenario 2(a) has a larger sentence average distance than 2(b) resulting in 2(a) having a smaller sentence similarity than 2(b).

The whole process of sentence similarity calculation is shown in the following algorithm 1. In this algorithm, we calculate an affinity matrix using the input word vector list from the preprocessing step. 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). $\sum_{i=1}^{n} D_i^2$ 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 helps with minimizing redundancy in the output summary by not selecting multiple sentences from the same topic. For clustering, spectral and DBSCAN clustering methods were considered due to their capability of being able to cluster irregular shapes. In clustering sentences, spectral clustering was found to perform better than DBSCAN because smaller input documents have lower density which hinders DBSCAN [25].

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 [31]. The following equation 5 shows the process of building an affinity matrix. In this equation, for every sentence pair S_i and S_j, we calculate their sentence similarity using the equation 4 and use the value in both A_{ij} and A_{ji} place of the affinity matrix A.

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

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

Algorithm 1 Sentence Similarity Calculation

$l \leftarrow$ length(WordVectorList)
$A \leftarrow [[0] * l] * l$
for each sentence$_i$ in WordVectorList **do**
    $D_{Square} \leftarrow 0$
    n $\leftarrow 0$
    for each sentence$_j$ in WordVectorList **do**
        for each word$_i$ in sentence$_i$ **do**
            $D_{msw} \leftarrow \infty$
            for each word$_j$ in sentence$_j$ **do**
                if Distance(word$_i$, word$_j$) $< D_{msw}$ **then**
                $D_{msw} \leftarrow$ distance(word$_i$, word$_j$)
                end if
            end for
            $D_{Square} \leftarrow D_{Square} + D_{msw}^2$
            $n \leftarrow n + 1$
        end for
        for each word$_j$ in sentence$_j$ **do**
            $D_{msw} \leftarrow \infty$
            for each word$_i$ in sentence$_i$ **do**
                if Distance(word$_i$, word$_j$) $< D_{msw}$ **then**
                $D_{msw} \leftarrow$ distance(word$_i$, word$_j$)
                end if
            end for
            $D_{Square} \leftarrow D_{Square} + D_{msw}^2$
            $n \leftarrow n + 1$
        end for
        similarity $\leftarrow \exp\left(\frac{-D_{Square}}{2 \times n \times \sigma^2}\right)$
        $A[i][j] \leftarrow A[j][i] \leftarrow$ similarity
    end for
end for
Return A

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 to 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 following equation 6. In this equation, 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 following 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 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 performance of the proposed method has been compared against three other Bengali text summarization methods to evaluate the correctness of generated summaries by using human written summaries as reference. The three methods, which have been used as a benchmark, are BenSumm [5], LexRank [10] and SASbSC [25]. All of these methods have been evaluated using four Bengali extractive text summarization datasets to test the robustness of the method's performance in various types of input. For evaluation of the methods, the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [18] 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 [25], 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. All four of these methods are further discussed in the following section.

WSbSC is the proposed model for this research. We use local word correspondence-based Gaussian similarity to perform spectral clustering for grouping semantically similar sentences together. We extract one sentence from each group as the output summary to minimize redundancy and maximize coverage.

SASbSC [25] is the first method we considered for comparing against the proposed method due to SASbSC being a recent model with a very similar approach to our model. SASbSC also uses word vector embedding and spectral clustering in their summarization workflow. However, it uses the average of word vectors in a sentence for calculating sentence similarity to be used in the clustering. After clustering, SASbSC uses cosine similarity between the sentence average vectors in a cluster to pick the best sentence from that cluster.

BenSumm [5] is another recent Bengali text summarization 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. BenSumm used TF-IDF based sentence vectors to build a similarity matrix which they used to cluster the sentences using agglomerative clustering. A Github implementation[4] provided by the authors is used in the comparison process.

LexRank [10] uses a TF-IDF based matrix and Googles PageRank algorithm [24] 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[5] which is used in the comparison process using a large Bengali wikipedia corpus[6].

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 using human generated summaries as reference. 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 the summary. In total, 500 different document-summary pairs are present in this dataset. The dataset is publicly available on Github[7] for reproducibility.

The second dataset is a **Kaggle** dataset[8] produced by Towhid Ahmed Foysal [11]. This dataset is a collection of summary article pair from The Daily Prothom Alo. The dataset is vast in size however the summaries are slightly lower in quality. We filtered out all the articles which are smaller than 50 characters, and with unrelated summaries to improve the quality of the dataset. After filtering, total 10,204 articles remained, each with two summaries in the dataset.

[4] *https://github.com/tafseer-nayeem/BengaliSummarization*
[5] *https://pypi.org/project/lexrank/*
[6] *https://www.kaggle.com/datasets/shazol/bangla-wikipedia-corpus*
[7] *dataset link*
[8] *https://www.kaggle.com/datasets/towhidahmedfoysal/bangla-summarization-datasetprothom-alo*

The third dataset we used for evaluation is the **BNLPC** [15] dataset. This dataset is a collection of news article summaries published with a text summarization method [15]. The dataset was collected from GitHub[9] for reproducibility. The dataset contains one hundred articles with three different summaries for each article.

The fourth dataset is a **Github** dataset[10] produced by Abid Mahdi. 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.

4.3 Evaluation Metrics

To evaluate the correctness of the machine generated summaries compared to the human generated summaries, we used the ROUGE method [18]. The method compares a reference and a machine generated summary to evaluate how well machine generated summaries align with the reference. The method uses N-gram-based overlapping to calculate a precision, recall and F-1 score for the summaries. we used the Rouge package[11] as the implementation 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 and generally is a good performance indicator. But using it as the only metric can also 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. Shared bigrams lead to a deeper analysis of syntactic similarities between the two summaries. 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 Comparison

We calculated the avaerage F-1 scores for three Rouge metrics (Rouge-1, Rouge-2, Rouge-LCS) for four extractive text summarization models (WSbSC (proposed), BenSumm [5], SASbSC [25], LexRank [10]) on four datasets (Self-Curated (SC), Kaggle, BNLPC, Github). The result of this evaluation is shown in table 1. In this table, we can see that our proposed model performs 11.9%, 24.1% and 16.2% better than the 2nd best 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 respectively than the 2nd best method (BenSumm) on the Kaggle dataset. It also performs a tie in R-1, 3% better in R-2 and 2.6% better than the 2nd best method (SASbSC) on R-LCS using the BNLPC dataset. It performs 58%, 86.4%, and 67.9% better than the 2nd best method (BenSumm) on the Github dataset in all three metrics.

[9] *https://github.com/tafseer-nayeem/BengaliSummarization/tree/main/Dataset/BNLPC/Dataset2*
[10] *https://github.com/Abid-Mahadi/Bangla-Text-summarization-Dataset*
[11] *https://pypi.org/project/rouge/*

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 [25]	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 [25]	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 [25]	**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 [25]	0.30	0.18	0.24
	LexRank [10]	0.22	0.14	0.20

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

These results are further visualized into three radar charts in figure 4 to compare the performance of the models using all four datasets to compare them on individual metrics. In these charts, we can see that proposed method performs more consistently and uniformly across all the datasets regardless of the quality of the dataset. But other models, such as BenSumm performs well in some datasets (SC, GitHub, BNLPC) but also fails in the Kaggle dataset. Similarly, SASbSC performs well in SC and BNLPC datasets but the performance decreases sharply in Kaggle and GitHub datasets. LexRank although performs similarly across all datasets but is far lower on average.

4.5 Experimentation

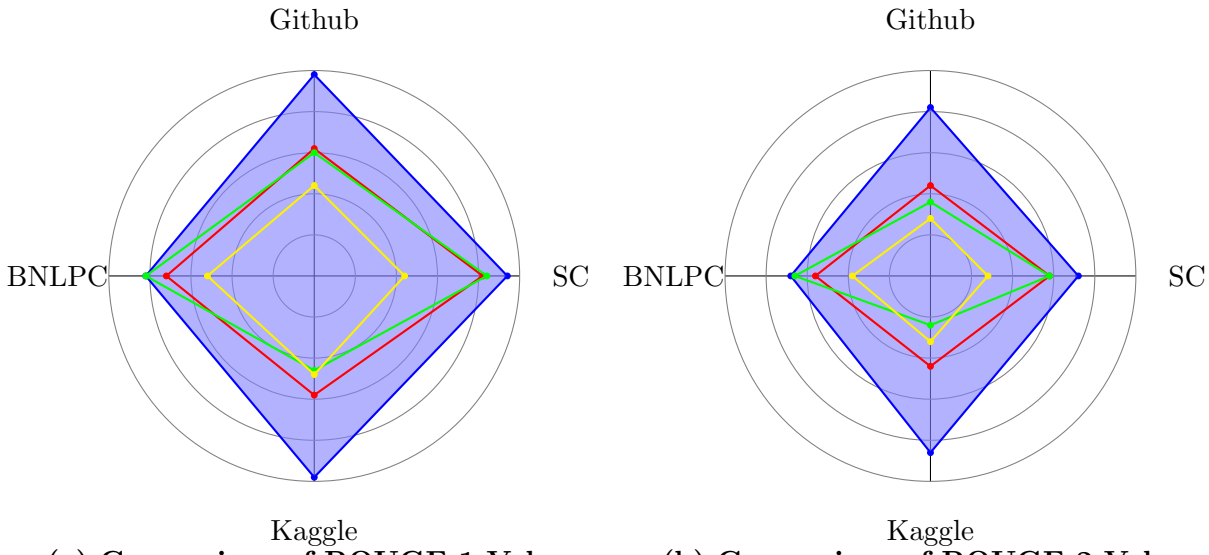
We experimented on our model with different ranking techniques and different values for standard deviation (σ) for the equation 4 to get the best rouge values for a summary. The standard deviation (σ) for the Gaussian Similarity represents a smoothing factor that can be used as a control variable to 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 lead extracting and TF-IDF ranking methods for to pick the a representative sentence from each cluster. 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 to pick the best σ. Sixty-three different values for σ from 10^{-12} to 10 on regular intervals have been experimented on and can be seen that 5×10^{-11} works best as the value for σ on our self-curated dataset. The result for fine-tuning process is shown in the following line chart (Figure 5).

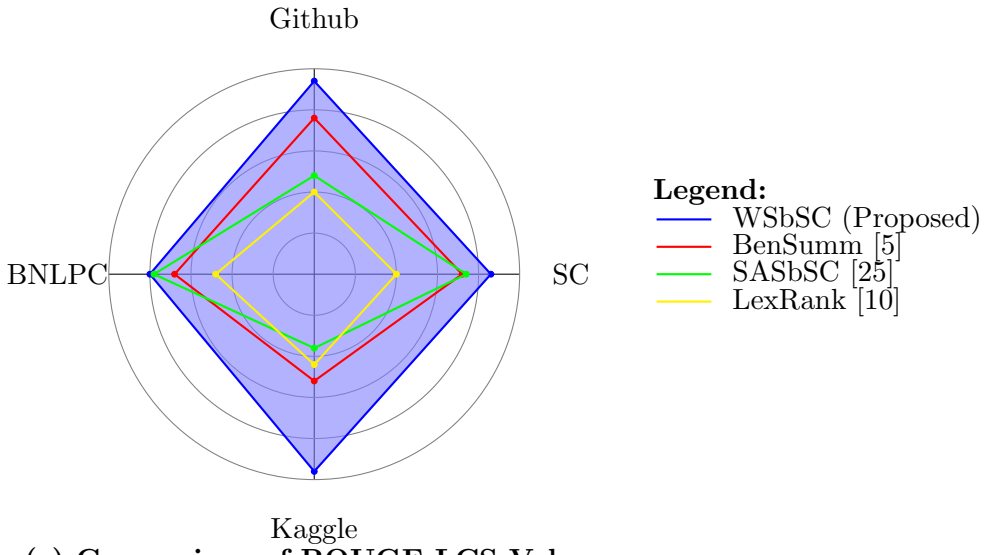
4.5.2 Different Sentence Extraction Techniques Inside Clusters

We implemented two sentence extraction methods to pick the most representative sentence from each cluster. Firstly, the lead extraction method is used where we just select the sentence that appears first in the input document because generally the earlier sentences in an input has



(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

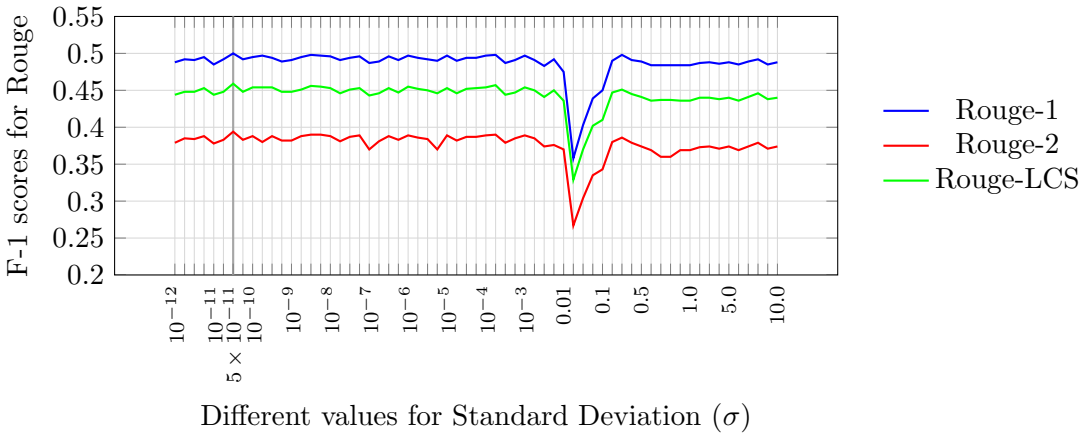


Figure 5: Fine-tuning for different standard deviation (σ) values

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

Table 2: Comparison of Result of different ranking techniques

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

more information on the context of the input document. The second extraction method is the TF-IDF ranking method, where we ranked the sentences by their TF-IDF scores and picked the sentence with the most TF-IDF score. We can see in the table 2 that the TF-IDF performs better on a high quality dataset like the self-curated dataset.

4.6 Implementation Into Other Languages

The proposed model is not language-dependent because it does not rely on any language specific features to summarize the input documents, thus it can be extended into other languages too. To implement this method into a language, we only need a language-specific tokenizer, a list of stop-words and a word vector embedding dataset. We implemented this model on three more languages 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. These languages are 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 Bengali. Although it performs slightly lower on Hindi, The score is still similar to Bengali. To evaluate the result on 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 [7] team for evaluation.

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 analyses 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 other graph-based extractive text summarization models, namely BenSumm [6], LexRank [10], and SASbSC [25], across multiple datasets (Self-curated, Kaggle, BNLPC, GitHub) of varying sizes and source. The proposed model shows performance improvement in three ROUGE metrics (Rouge-1, Rouge-2, Rouge-LCS). This performance improvement can largely be attributed to the novel approach of calculating sentence similarity due to being one of the key change from the SASbSC method [25]. 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. The Gaussian similarity-based approach used to calculate the word similarities provides a novel and precise method for capturing the semantic relationships between sentences to 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.

To calculate the similarity, we developed a novel method that can calculate the similarity between two set of vectors and is relevant in the context of languages. Our proposed strategy is more suited for this task than other strategies that can compare two sets of vectors such as Earth Movers Distance (EMD) [26], Hausdorff Distance [16], Procrustes Analysis [12] due to it focusing on the context of language more. EMD [26] tries to find the lowest amount of "work" needed to transform one set into the other one to calculate similarity. It considers adding a new vector to a set, removing a vector from a set, scaling a set, moving a vector in the set, rotating the set etc. as "work". This process is very computationally expensive as hundreds of thousands separate possibilities have to be checked for each vector in each set. EMD also focuses on scaling and rotating, which are not relevant in word vector space as rotating a sentence does not hold any semantic meaning. Another method, Hausdorff distance [16] takes the worst case scenario and calculates the highest distance between two vectors each from a set. This method needs the same amount of calculation as the proposed method but it is easily influenced by outliers due to only considering the worst case scenario. It also does not have any local vector correspondence and wasn't used because words tend to spread out over the whole word space and it would suffer from the same problem as the averaging method. Procrustes Analysis [12] tries to find the lowest amount of misalignment after scaling and rotating the two sets. But both scaling and rotating are irrelevant in the context of word vectors.

On the other hand, the proposed method focuses on local vector correspondence 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, 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 in the model is also well-suited for the clustering task because spectral method 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 significant improvement over other traditional methods that rely on word averaging, which often dilute the semantic meaning of a sentence. Another key strength is the scalability across languages. WSbSC can be easily adapted to other languages due to it requiring very few language-specific resources. This scalability is demonstrated in the experiments with Hindi, Marathi, and Turkish languages (Table 3). This generalizability makes the model highly versatile and valuable for extractive summarization in low-resource languages. 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.

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 [13] dataset used here is heavily reliant on wikipedia for training which could introduce some small unforeseen biases. In cases 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. Its ability to accurately capture sentence similarity, reduce redundancy, maximize coverage 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 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. 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] Rasim M. Alguliyev, Ramiz M. Aliguliyev, Nijat R. Isazade, Asad Abdi, and Norisma Idris. Cosum: Text summarization based on clustering and optimization. *Expert Systems*, 36(1):e12340, 2019. doi: https://doi.org/10.1111/exsy.12340. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/exsy.12340`.

[3] Jean Babaud, Andrew P. Witkin, Michel Baudin, and Richard O. Duda. Uniqueness of the gaussian kernel for scale-space filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(1):26–33, 1986. doi: 10.1109/TPAMI.1986.4767749.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

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

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

[31] 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.

[32] 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.