Performance Analysis of Different Word Embedding Models on Bangla Language

Zakia Sultana Ritu
Computer Science and Engineering
Shahjalal University of Science and Technology
Sylhet, Bangladesh
zakiaritu.cse@gmail.com

Nafisa Nowshin
Computer Science and Engineering
Shahjalal University of Science and Technology
Sylhet, Bangladesh
nafisanowshin107@gmail.com

Md Mahadi Hasan Nahid
Computer Science and Engineering
Shahjalal University of Science and Technology
Sylhet, Bangladesh
nahid-cse@sust.edu

Sabir Ismail
Computer Science and Engineering
Stony Brook University
New York, United States
sabir.ismail@stonybrook.edu

Abstract—**In this paper we discuss the performance of three-word embedding methods on Bangla corpus. Word embedding is a big part of natural language processing related research works. Many research works have focused on finding appropriate methods of word clustering process. Previously N-gram models were used for this purpose but now with the improvement of deep learning methods, dynamic word clustering models are preferred because they reduce processing time and improve memory efficiency. In this paper we discuss the performance of three word embedding models namely, word2vec in Tensorflow, word2vec from Gensim package and FastText model. We use same dataset on all the model and analyze the outcomes. These three models are applied on a Bangla dataset containing 5,21,391 unique words to produce the clusters and we evaluate their performance in terms of accuracy and efficiency.**

Keywords—**Natural Language Processing(NLP) , machine learning, deep learning, word cluster, word embedding, Bangla word clustering, word2vec, fasttext, skip-gram, CBOW, GloVe.**

I. INTRODUCTION

Bangla is a major world language and it will increase more important in the coming years to come with the digitization of the Bangla language. As the importance of this language grows so does the research works concerning Bangla language. One of the major sectors of Bangla natural language processing involves establishing a method for producing fast and accurate word clusters. Much work is being done in this sector and continuous effort is being given to find an appropriate method for producing word clusterings. In this paper we attempt to apply three dynamic word clustering models to compare their performance in producing Bangla word clusterings.

Word clustering can be referred to as a technique for partitioning sets of words into subsets of semantically similar words. This has a far reaching effect in many NLP related works. It is increasingly becoming a major technique used in a number of NLP tasks ranging from word sense or structural disambiguation to information retrieval and filtering. To reach any decision about the performance of a model in producing word clusterings, we need to evaluate its performance by applying it to a large dataset and compare the results. But working with large dataset means more run time and consequently less efficiency. On the other hand, if it is done with smaller dataset, the clusters won't be accurate. So in choosing a word clustering method our main goal is to reduce run time and increase efficiency in producing accurate word clusters.

Word clusters include semantically similar words, meaning it will group those words that are similar in meaning and tend to occur in similar contexts in natural language. There are many approaches to compute semantic similarity between words based on their distribution in a corpus. Much research work has been done to find an efficient and accurate model for building word clusters. Although at first N-gram models were used to construct word clusters, in recent times with the improvement of deep learning methods dynamic models have become more popular in building word clusters. Dynamic models construct the vector representation of words and build clusters from them. In this paper we discuss the performance of three variations of dynamic word clustering models, which are, word2vec in Tensorflow, word2vec from Gensim package and FastText model in case of constructing Bangla word clusters.

This paper is arranged as follows, in section 2, we have shed light on some of the previous approaches to construct word clustering in Bangla and other languages and their performance. In section 3 we have discussed about the dataset that we have used in evaluating the performance of the models. In section 4 we discuss in brief the full methodology applied in our work. Then in section 5 we present an analysis of the results we got from applying these three models and evaluate their performance in producing accurate Bangla word clusterings. We conclude in section 6.

II. BACKGROUND STUDY

Word clustering is an important aspect of dealing with large datasets in research work. So, much research work has been done with a view to finding appropriate methods for constructing word clustering in Bangla and in various other languages. We will discuss some of these works below.

Previous word clustering techniques mostly involved using N-gram model to construct the clusters. This can be observed in the works of Ismail and Rahman [1], who proposed a Bangla word clustering method based on N-gram Language Model. In this paper they tried to cluster bangla word using their semantic and contextual similarity. In this approach they tried to cluster the words based on the idea that, the words that have similar meaning and are used in similar context in a sentence, belong to the same cluster.

Their work was slightly upgraded later by Urmi, Jammy and Ismail [2]. They proposed a unsupervised learning approach to identify stem or root of a Bangla word from contextual similarity of words. Their object was to build a big corpus of Bangla stems along with their respective inflectional form. They worked with the assumption that if two words are similar in spelling and are used in similar context in many sentences, they have a higher chance of originating from the same root. They implemented 6-gram model for stem detection and achieved an accuracy of 40.18%. They have concluded that with big amount of text data this model will improve further.

Researchers then focused on producing word clustering in dynamic approach and its performance. We get insights about this from the works of Yuan [3], who showed that word clustering technique that is based on word similarities is better than conventional greedy approach in terms of speed and performance. The basic approach of this work was to check for a certain word in the corpus, its co-occurring words for similarity. That is to say, if two words are similar, their co-occurring word pattern will also be similar. Based on this they computed word clusters and when compared with other clustering methods, this approach was found to be more efficient.

The performance of dynamic models in producing Bangla word clusters was shown by Ahmed and Amin [4]. They discussed the effect of Bangla word embedding model in document classification. They worked with a dataset prepared from Bangla newspaper documents. They applied word2vec model to generate vector representation of words for word clustering. Using this they prepared clustering of word embeddings that are found in close proximity to each other in feature space. This information was later used as features to solve Bangla document classification problem.

Altszyler, Sigman and Slezak [5] tried to find out if LSA and word2vec model's capacity to identify relative dimension increases with increase in data. They found out that Word2vec can take advantage of all types of documents while LSA only gives better performance when out-of-domain documents are removed from corpus.

In case of Arabic language Soliman, Eissa and El-Beltagy [6], found that the performance measure of word2vec differs from dataset to dataset but on each dataset it shows good performance in capturing similarity among words.

Upgrading the performance of word2vec in finding vector representation of words in huge datasets like a dataset containing one billion words were attempted by Rengasamy, Fu, Lee and Madduri [7]. They applied word2vec in a multi-core system and found that this approach is 3.53 times faster than original multi-threaded word2vec implementation and 1.28 times faster than recent parallel word2vec implementation.

Ma and Zhang [8] discussed the effect of word2vec in reducing the dimensionality of large datasets. They found out that, in dealing with large scale training data, word2vec helps in clustering similar data. This strategy can reduce data dimension and speed up multi-class classifications.

With the goal of preparing vector representation of words, Naili, Chaibi, Ghezala [9], applied LSA, Word2vec and GloVe on both English and Arabic language. They reached the conclusion that although all three methods performance depend on the language used, among the three, word2vec gives the best vector representation of words.

Robert Bamler and Stephan Mandt [10] tried to find the semantic evolution of individual words over time in time-stamped datasets. They applied Word2vec model to produce the embedding vectors. They showed experimentally that both skip-gram filtering and smoothing lead to smoothly changing embedding vectors that help predict contextual similarities at held out time stamps.

Fasttext model is a relatively new model ventured in producing word clusterings. It is a variation of skip gram model architecture of word2vce model which was proposed by Bojanowski, Grave, Joulin and Mikolov [11]. The method they followed was, each word was represented as a bag of character n-grams and vector representation was constructed from them. This allowed them to construct word clusters for words not present in the training data. They concluded that this method gives state of the art word representations for both similarity and analogy task.

Finally, we can say that there is rich literature growing on word embedding techniques and there is much scope of improving in this sector.

III. DATA COLLECETION

We used three separate corpus, and merged them. First corpus is, SUMono [12] which contains available online and offline Bangla text data. We also used a news corpus, which contains news data from Bangla news websites. We also used Bangla wiki data from wikipedia. The detail is given in table I. Accuracy of any model largely depends on the dataset it is applied on. If a word is used in various kind of sentences, then the trained model can be more accurate as it covers a large area of variety. The more frequent the words are, the more accurate the model will be. This corpus contains Bangla text data on various topics. Because the corpus was built taking contents from various sources like Bangla articles from wikipedia, Bangla news portals and from writings of different renowned Bengali writers. As the text is collected from various sources it covers topics of different kinds and sectors as well as the language structure of Bangla used in day to day life. This is an important aspect of the data collected, because to get better and accurate clusters dynamically we need data that covers vast areas in which a specific word can be used.
Figure I represents some of the most frequent words from the corpus. And detailed information of the corpus is represented in table I.

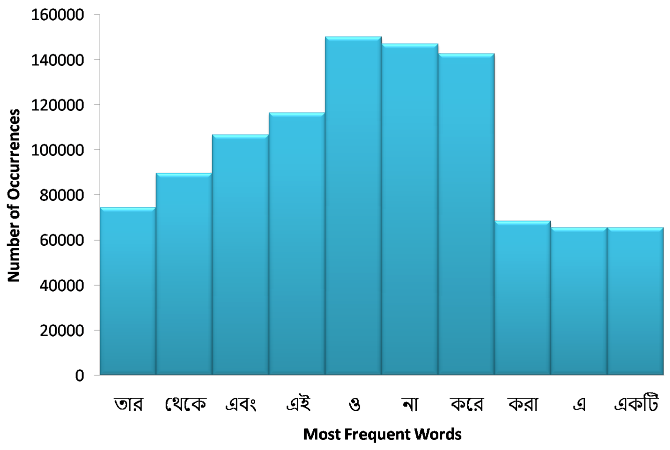

Figure 1. Histogram of Most Frequent Words with Number of Occurrences

Table I
Details of the Corpus

Total sentences	1,593,398
Total words	2,51,89,733
Unique words	5,21,391

IV. METHODOLOGY

Over the years many techniques have been introduced for word clustering. Most of the approaches give high accuracy for clustering English words. As Bangla is a more complicated language, it is hard to gain high accuracy. We attempted three different approaches to determine which approach works better for Bangla word clustering. Nowadays vector representation of words have become the most popular approach for building word clusters. We applied three machine learning approaches which are based on vector representation of Words. We followed some steps to train our datasets. these steps are discussed below.

A. The Basic Steps

- **Corpus:** The corpus is stored as a text file. In the text file the data constitutes of Bangla sentences. They can be treated as strings. But We can not feed the strings directly to this model, so some pre-processing was done on the dataset.
- **Tokenizing:** We had to pre-process the corpus in order to use them as our proper input. Firstly, we couldn't feed a word just as a text string to a model. For this, we tokenized each word. Example:

  আমার সাথে বাংলায় কথা বল => 'আমার', 'সাথে', 'বাংলায়', 'কথা', 'বল'

- **Training:** We used the tokenized dataset for training. We compared the output from each model individually by using different window sizes, vector sizes and iterations over the dataset.

B. Our Experiments

*1) **Experiment I: Word2vec in Tensorflow**:* We used the official version of sentence embedding implementation of tensorflow. The code generates word clusters based on the features of Word2vec using the Skip-Gram model and the Negative Sampling accelerated classification algorithm. We tried different window sizes, vector sizes and iterations.

*2) **Experiment II: FastText Model**:* Facebook's AI Research lab created the FastText library for word embedding and text classification. Both Skip gram and CBOW model can be trained with FastText. For the training of Skip Gram model, we kept the window size at 5 and vector size was 100. we trained the CBOW model as well.

*3) **Experiment III: Word2vec from Gensim package**:* The python library Gensim provides Word2Vec class for working with word embedding. We tuned the parameters and checked results both for Skip gram and CBOW model.

C. Training Time

The size of our corpus is 59856174 bytes. We used a computer with 4GB RAM, core i3-3110M CPU. Training Time for each experiment is as follows-

Table II
Training Time of the Experiments

Experiment	**Training Time**
Word2Vec in Tensorflow	18 minutes
FastText- Skip gram Model	23 minutes
FastText- CBOW Model	24 minutes
Gensim Word2Vec- Skip gram Model	30 minutes
Gensim Word2Vec- CBOW Model	32 minutes

V. RESULT ANALYSIS

There are similarities among the clusters from three different approaches as well as some significant differences. For some words, we got a similar set of words for a set of models. But the variety was also notable. We tried to get the most satisfactory results from each approach. We applied various combination of window and vector sizes in order to tune the parameters to get the most optimum and satisfactory results. Table III shows parameter tuning for each approach.

Table III
Parameter Tuning for Optimum Results

Experiment	**Window size**	**Vector size**	**Iteration**
I- Word2vec in Tensorflow	4	1000	10
II- Skip gram Model	5	100	5
II- CBOW Model	5	100	5
III- Skip gram Model	5	400	5
III- CBOW Model	5	400	5

Some sample results from the experiments are given in table IV, V, VI, VII and VIII.

Table IV
Results from Experiment I

Random Word	Words on cluster
আমরা	আমাদের, আমি, চাই, যখন, তাই, তারা, কি, সেই, কিন্তু, সবাই
তাঁর	তার, সেই, সাথে, তাঁদের, একজন, একই, ওই, তিনি, পরে, বলে
জন্য	প্রয়োজন, সুযোগ, জন্যে, পাশাপাশি, তাই, দরকার, কারণ, কিছু, কিন্তু, তাদের
কোন	কোনো, এমন, অন্য, কারণ, তবে, সেটা, বা, নেই, তাই, এখনো
পারে	হবে, পারবে, পারি, হলে, পারেন, হতো, চাই, চায়, পারেনি, থাকে
হতে	যেতে, থাকতে, করতে, হলে, না, রাখতে, তাই, তাহলে, তবে, দিতে
বড়	সবচেয়ে, অবস্থা, খুবই, খুব, আমাদের, অনেক, কিছু, মতো, মানুষের, আছে
টাকা	হাজার, লাখ, কোটি, টাকার, খরচ, পাঁচ, মাত্র, বিক্রি, তিন, প্রায়
নতুন	মাধ্যমে, তৈরি, কাজ, জন্য, বিভিন্ন, নানা, একটি, এই, সব, একই
দেখা	অদেখা, দেখাইত, দেখাক, দেখায়ই, দেখাসহ, দেখায়া, দেখাএইসবই, দেখাত, দেখাবো, দেখাবো

Table V
Results from Experiment II-Skip Gram

Random Word	Words on cluster
আমরা	আমরা।, কীআমরা, নয়আমরা, আমরাই, আমরাতো, হয়আমরা, কিআমরা, হোকআমরা, আমর, লেআমরা
তাঁর	তাঁর।, তাঁরই, তাঁরও, তাঁহা, তাঁরা, তাঁতীও, তাঁরাই, তাঁ, তাঁকেসহ, ওঁর
জন্য	জন্য।, জন্যেঃ, জন্যও, জন্যে।, সৌজন্য, জন্যে, এজন্য, জন্যই, জন্যি, এরজন্য
কোন	কোনস, কোন্, কোনোা, কোনোও, কোনো, কোনডা, কোন্, কোনোই, কোনও, কোনই
পারে	পারো, পারে।, পারেএ, পারেতখন, পারেঃ, পারেআর, পারেএমন, পারেনই, পারেন।, ঐপারে
হতে	ঝতে, লখনউতে, ধতে, ইইউতে, নড়তে, নতে, পেতে, চড়তে, অইতে, ওতে
বড়	বড়বড়, বড়র, হড়বড়, বড়ও, বড়ইর, বড়ছোট, ছোটবড়, ছোটছোট, ছোট, নড়বড়
টাকা	টাকা।, দশটাকা, টাকায়ও, টাকাসহ, হাজারগুণ, দুইটাকা, হাজারও, দুটাকা, হাজারদিঘী, টাকাকী
নতুন	নতুননতুন, নতুনতর, নতুন।, নতুনই, নত, নতূন, জৈতুন, নিত্যনতুন, নতুনরা, চালু
দেখা	অদেখা, দেখাইত, দেখাক, দেখায়ই, দেখাসহ, দেখায়া, দেখাএইসবই, দেখাত, দেখাবো, দেখাবো

Table VIII
Results from Experiment III- CBOW

Random Word	Words on cluster
আমরা	ইয়ার্কদের, যতই, পৌঁছাতে, হোক, লিডাররা, শেয়ারিঙের, অবশ্যই, কথাও, চিন্তা, দুবারই
তাঁর	তাঁরা, তৈরিতে, কো, সারা, বর্জনের, সৃষ্টিতে, ভাইরাস, অপরিশোধিত, কলফনি, লিপির
জন্য	যুক্তি, চিহ্নিত, জোরদার, পরিবেশ, আশ্বাস, প্রকাশ, সমালোচনা, অন্তর্ভূক্ত, প্রত্যাখ্যান, প্রতিরোধ
কোন	কোনো, প্রত্যয়, কিছুরই, কারণ, বইতে, তেমনভাবে, আধ্যাত্মিকতা, সাবজেক্টে, ঘটেনি, ইন্ডাস্ট্রির
পারে	ঈর্ষায়, দগ্ধ, অভিবাসী, পাঠ্যবই, পারবেন, পারত, পারবে, পেরেছিল, অস্বীকৃতি, নামিজউদ্দিন
হতে	হতেই, থাকতে, দিতে, যাইতে, ফিল্মগুলো, পেতে, নিতে, সাজিয়া, ঘটাতে, জানতে
বড়	ফুটো, শ্লেষ্মা, দরজাটা, আলো, পাথরের, লেজের, চওড়া, রাস্তা, অন্ধকার, খাটো
টাকা	বছরের, 'ছয়, চলাচলের, দশ, সপ্তাহের, পূর্ববঙ্গে, লাখ, জেলার, উপলক্ষে, গান্ধীর
নতুন	গান্ধীকে,পাকিস্তান, বণ্টন, ক্রয়, মন্তব্য, শিক্ষা, পরিকল্পনার, অধিগ্রহণ, মাউন্টব্যাটেন, প্রস্তাবিত
দেখা	পাওয়া, জানা, পাহারায়, বেঁকে, কাঁচুলি, ব্লেজারেই, উর্দুকে, শহরগুলোর, ঝরেতুমি, গেয়ে

Table VI
Results from Experiment II- CBOW

Random Word	Words on cluster
আমরা	আমরাযদিও, আমরাই, আপনাকেও, আমরাও, বলেছিআমরা, আপনা, কীআমরা, কীটও, আপনাকেই, কীটস
তাঁর	পুনঃআলোচনার, সুলোচনার, আলাপআলোচনার, কেস্তার, তাঁরই, কাইয়ুমআলোচনার, সৃষ্টিশীলতার, ধিক্কার, মনিকার, বিঘার
জন্য	জন্যও, জন্য।, জন্স, সৌজন্য, এজন্য, জন্যেঃ, সৌজন্যঃ, জন্যই, জন্যে।, তজ্জন্য
কোন	কোনোও, কোনোা, কোনোই, কোনো, লুকোনো, থোন, কোনস, কোন্, কোন্, কোনোটিই
পারে	পারেআর, পারেতখন, পারে।, পারেএ, পারো, পারডন, পারেঃ, পারেও, পারব্ধর, পারদ
হতে	কইরতে, ঝরতে, ভরতে, কসরতে, খোরতে, শরতে, মরতে, ধরতে, ঠকতে, কুদরতে
বড়	বড়র, বড়ও, হড়বড়, বড়ইর, বড়ই, বড়সড়, গড়বড়, বড়সড়ো, নড়বড়, বড়বাড়ী
টাকা	দশটাকা, ওসাকা, শলাকা, পোঁদপাকা, গাঢাকা, টাকা।, জলধাকা, ইয়াকা, হাজারী, এলাকা
নতুন	নতুননতুন, নতুন।, 'নতুনই, ফোরামটি, অনুষ্ঠিত, পুনর্গঠিত, তিনচারটি, উৎকণ্ঠিত, পরিচালকমণ্ডলীর, ভূখণ্ডটি
দেখা	অদেখা, দেখায়ই, দেখাসহ, দেখাইত, দেখাএইসবই, দেখাত, দেখাক, দেখাবো, দেখায়া, দেখাও

Table VII
Results from Experiment III- Skip Gram

Random Word	Words on cluster
আমরা	আমি, তোমরা, সেটা, তাহলে, পারি, এখনো, হয়তো, এখানে, করেছি, তোমাকে
তাঁর	তাঁদের, স্বাধীনতার, যিনি, নেন, নিজ, মেডিকেল, দেন, করছিলেন, পরিবারের, যুদ্ধ
জন্য	জন্যে, সুযোগ, চেষ্টা, কাজে, উদ্দেশ্যে, মাধ্যমে, ব্যাপারে, ব্যবস্থা, করলে, পর্যায়ে
কোন	কোনো, থাকার, ছাড়া, তাতে, বসানোর, আপত্তি, প্রয়োজন, তেমন, উপায়, এমন
পারে	পারবে, চায়, পারেনি, পারেন, পারত, হতো, পারব, বাধ্য, চাই, পারবেন
হতে	পেতে, যেতে, থাকতে, রাখতে, আনতে, দোকানেও, নিতে, ঘটতে, বেশিও, লাগতে
বড়	ছোট, সবচেয়ে, শিরোনামায়, মেয়ে, গায়কেরা, সুন্দর, জোরটা, জিনিস, মধ্যবিত্তদের, ছেলে
টাকা	কোটি, হাজার, পাঁচ, টাকার, লক্ষ, বরাদ্দ, লাখ, বছর, প্রায়, গত
নতুন	নির্যাতন, প্রক্রিয়া, জাতীয়, পূর্বে, মামলা, এলাকায়, সেনাবাহিনী, অর্থনৈতিক, বিচারের, জোট
দেখা	কমে, রয়ে, ফুসফুসে, গবেষণায়, জানা, বেড়ে, পাওয়া, রোগী, রূপ, জীবদ্দশা

From the given examples of the clusters produced by the three different models we can come to some decisions. If we consider clusters containing similar and synonymous words Word2Vec implementation of Tensorflow gives good results. Comparing the two variations of FastText model, the FastText-Skip Gram model is the best because it gives all the inflections of a specific word. The FastText-CBOW model do not produce such accurate results. Gensim library based skip-gram model gives contextually similar words but fails to give inflection of words. The CBOW architecture of this model does not produce good clusters rather it gives noisy output. So from evaluating the results of these models, we can come to the conclusion that FastText-Skip Gram model is the more accurate and efficient model for building Bangla word clusters.

FastText uses n-grams of a word and create vectors for the sum of all the n-grams of the word. As a result it can produce output even if the word is not in the corpus. But the other approaches can not generate results for an unknown word. Though if we want to get cluster for a unknown word from FastText model, there was no satisfactory results but it did gave something. If the dataset can be prepared properly for the FastText skip gram model, we think it will produce really amazing and much more accurate word clusters.

VI. Conclusions

Bangla is a complex language with a wide range of vocabulary containing many rare words. Language structure, use of complex words, multiple meanings in different context all of these reasons makes it really difficult to choose one model as the best model for Bangla word clustering. The contents of the dataset also plays a big role in deciding this. We have tried to give some perspective on some of the dynamic approaches that have been used for Bangla word clustering. Among the models applied, we have reached the conclusion that FastText-Skip Gram model produces the best result on the given dataset. We can get more accurate results by increasing the size of the dataset.

References

[1] S. Ismail and M. S. Rahman, "Bangla word clustering based on n-gram language model," in Electrical Engineering and Information & Communication Technology (ICEEICT), 2014 International Conference on. IEEE, 2014, pp. 1–5.

[2] T. T. Urmi, J. J. Jammy, and S. Ismail, "A corpus based unsupervised bangla word stemming using n-gram language model," in Informatics, Electronics and Vision (ICIEV), 2016 5th International Conference on. IEEE, 2016, pp. 824–828.

[3] L. Yuan, "Word clustering algorithms based on word similarity," in Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2015 7th International Conference on, vol. 1. IEEE, 2015, pp. 21–24.

[4] A. Ahmad and M. R. Amin, "Bengali word embeddings and it's application in solving document classification problem," in Computer and Information Technology (ICCIT), 2016 19th International Conference on. IEEE, 2016, pp. 425–430.

[5] E. Altszyler, M. Sigman, and D. F. Slezak, "Corpus specificity in lsa and word2vec: the role of out-of-domain documents," arXiv preprint arXiv:1712.10054, 2017.

[6] A. B. Soliman, K. Eissa, and S. R. El-Beltagy, "Aravec: A set of arabic word embedding models for use in arabic nlp," Procedia Computer Science, vol. 117, pp. 256–265, 2017.

[7] V. Rengasamy, T.-Y. Fu, W.-C. Lee, and K. Madduri, "Optimizing word2vec performance on multicore systems," in Proceedings of the Seventh Workshop on Irregular Applications: Architectures and Algorithms. ACM, 2017, p. 3.

[8] L. Ma and Y. Zhang, "Using word2vec to process big text data," in Big Data (Big Data), 2015 IEEE International Conference on. IEEE, 2015, pp. 2895–2897.

[9] M. Naili, A. H. Chaibi, and H. H. B. Ghezala, "Comparative study of word embedding methods in topic segmentation," Procedia Computer Science, vol. 112, pp. 340–349, 2017.

[10] R. Bamler and S. Mandt, "Dynamic word embeddings," arXiv preprint arXiv:1702.08359, 2017.

[11] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," Transactions of the Association for Computational Linguistics, vol. 5, pp. 135–146, 2017.

[12] M. A. Al Mumin, A. A. M. Shoeb, M. R. Selim, and M. Z. Iqbal, "Sumono: A representative modern bengali corpus."