A Comprehensive Study of Bengali Embedding Models: Insights and Evaluations

Aminul Islam Bulbul[1], Saurav Das[2], Sheikh Abu Haidar Noori[3]
Daffodil International University
Dhaka, Bangladesh
{[1]aminul15-12040, [2]saurav15-11851}@diu.edu.bd, [3]drnoori@daffodilvarsity.edu.bd

Abstract—**Word embeddings in Natural Language Processing (NLP) represent words as vectors, encapsulating both semantic and syntactic meanings. Prominent models such as Word2Vec, FastText, and GloVe play a crucial role in various NLP tasks. This study evaluates these models, trained on a 240 million words corpus derived from nearly 900,000 Bengali newspaper articles, which we scraped using Scrapy. Our evaluation process involved fine-tuning parameters such as vector dimensions, epochs, window sizes, and minimum counts. We assessed the Continuous Bag of Words (CBOW) and SkipGram architectures across Word2Vec and FastText models, measuring their performance. To benchmark these models, we created 133 unique semantic and 103 syntactic Bengali question sets for the first time, assessing accuracy, cosine similarity, training time, memory usage, and a combined evaluation metric. Additionally, we utilized the confusion matrix for concept categorization. Comparing models trained from scratch and using Gensim, we found that models with 25 epochs, a minimum count of 35, and 300 dimensions delivered optimal performance. Specifically, Gensim Word2Vec with SkipGram achieved the highest semantic task accuracy, while FastText from Scratch excelled in syntactic tasks. All the models showed optimal performance for concept categorization in both semantic and syntactic analogy tasks. Additionally, models trained with 100 dimensions consistently showed higher cosine values, indicating better prediction purity, while models trained with 300 dimensions predicted the maximum number of correct answers. Our experiments revealed that models trained from scratch outperformed those trained with Gensim in FastText, with each model exhibiting strengths in different aspects of NLP tasks. Future research will expand on this work by introducing more diverse question sets.**

I. Introduction

Language is one of the best mediums for communication among others. In recent years, natural language processing (NLP) has gained significant attention due to its applications in numerous domains, including machine translation, sentiment analysis, and text summarization. All these modern activities happen through the computer, but it does not understand the raw text, the reason why it is important to convert the text into a numeric format. To do that continuous vector representation comes into place before the word embedding has been introduced [1, 2, 3]. Bunch of other ways to make it, namely, one-hot encoding, count-based representation, neural network language model (NNLM). Many researchers prove that word-embedding or the vector representation of words can demonstrate a better result and also can simplify numerous tasks in the field of NLP [4, 5, 6, 7]. Term word embedding became popular after publishing the word2vec model where the authors introduce the model with two different architectures CBOW and SkipGram with hierarchical SoftMax method[4].In the same year they proposed negative sampling instead of hierarchical SoftMax to speed up the training process. They find that negative sampling takes less time in training and works better than hierarchical SoftMax method[8]. Another approach comes into this field in the next year, which extends the Word2Vec approach to learn distributed representations not only for words but also for larger units of text such as sentences and documents. It captures semantic relationships between words and phrases which allows the model to get accurate natural language processing tasks like sentiment analysis and document classification [9]. In 2016 Grave, E., Joulin, A., Mikolov, T., and Bojanowskiproposed another model, where they introduced it as FastText model, an extension of Word2Vec that incorporates sub-word information to handle out-of-vocabulary words and improve performance on morphologically rich languages. Unlike Word2Vec which represents each word as a single vector, FastText constructs representations by summing the embeddings of character n-grams inside each word. This allows it to capture morphological information about the sub-word units [10]. Another popular embedding model is GloVe (Global Vectors for Word Representation), proposed by Pennington et al. (2014). Unlike the models that only use local context windows, GloVe uses a global log-bilinear regression model to get vector representations by combining both global word-word co-occurrence statistics and local contextual information from the corpus[11]. By integrating LDA and Word2vec, Duan, Zhang, and Jiang, demonstrate that the FNS-LDA2vec model improves topic extraction efficiency and provides better insights into stock investors[12].In recent times, contextualized word representations generated from deep bidirectional language models have achieved great results on several NLP tasks. Instead of considering only words they consider the context to train the model. ELMO and BERT are the most recent embedding models. Bengali is fifth and seventh most pronounced language according to native and general speakers respectively [13, 14]. Because of the complex architecture of the Bengali language, it is more difficult to work with than that of English. Since the language is morphologically rich and the composition is hard, a tiny change or modification can convert the entire meaning into a contrary meaning [15].

Several studies were found for classification problems in the Bengali language where they used only the word2vec model as feature extractor[16, 17, 18]. A. Ahmad and R. Aminuse the wor2vec model for document classification for the first time in Bengali language[16].H. A. Chowdhury, Imon, and Islamuse word embedding for authorship attribution for Bengali literature[19]. Khatun, A. Rahman, S. Islam and Marium-E-Jannat used character level embedding for authorship attribution[17]. Sumit, M. Z. Hossan, Muntasir and Sourov work with sentiment analysis for Bengali language by using word embedding[18]. Some work has been done to compare the embedding models or to find a robust one [20, 21, 22, 23].[20]Conducted intrinsic evaluation for the word2vec model in the Bengali language. Similarly, [23] performed both intrinsic and extrinsic evaluations with the same model to identify the best embedding model. [21, 22] demonstrated the performance of various embedding models through semantic and syntactic tasks.[24]In 2022 an author name Mars, provides a detailed overview of the evolution from word embeddings to pre-trained language models (PLMs), discussing various models' architectures, parameters, and training objectives.

[25]Garrido-Muñoz, Montejo-Ráez, Martínez-Santiago, and Ureña-López explore the challenges posed by biases in deep neural networks for NLP. They highlight how bias can arise from training corpora and discuss methods for detection and mitigation.Some issues like biasness among words can be found from the training time.However, most of the studies still have some shortcomings. In this study we focus on those specific gaps and try to improve them. Here we are providing some notion about the gaps we have tried to fill up. Bangladeshi newspapers often discuss topics like 'Mango' more frequently than 'Avocado,' and they tend to cover geopolitical events involving countries like 'Ukraine,' 'Russia,' 'Israel,' and 'Palestine' rather than focusing on 'Turkmenistan.' Therefore, asking questions about 'avocado' or 'Turkmenistan' may not yield the best results. However, making analogy questions with relevant information can mitigates this issue. Another concern is, use of rare words. For instance, when trying to predict synonyms for word like 'শশাঙ্ক' which is 'চাঁদ' (moon) or 'কুজ্ঝটিকা' which is 'কুয়াশা' (fog) perhaps the model will make incorrect prediction since these words are rarely found in newspaper data. So we decided to ignore uncommon words like 'Turkmenistan', 'Avocado' etc. [24]Also we have to think about the biasness which might arise from training corpora. Therefore, we focus on making a proper analogy question set or choosing words for synonym detection natively. It is undoubtedly a crucial part for judging any embedding models properly. We considered both human correctly predicted answers and cosine mean value to get the robust embedding model. In this paper we tuned every parameter that has a massive impact on model's result. We also focus on performance deviation of the models on the basis of frameworks, they have been trained**.** The Bengali language presents unique challenges and opportunities for NLP research due to its rich linguistic characteristics and limited resources. IN this experimental research we have collected almost 90k news articles where the corpus contains 16 million unique words. To collect the data, we created a web scraper by using scrapy. Our data set contains various types of data. We train word2vec (CBOW and SkipGram), Fasttext (scratch, gensim) and Glove models with our corpus. In our training we tune various parameters to get the best fit for each model. We create two analogy question sets for semantic and syntactic tasks which support the newspapers data. We also create ground truth sets in order to evaluate the predictions. We only take the best prediction as the correct answer (k=1) instead of taking more than one (k=n where k>1)[20]. We compare the same model's training methodology by training the fasttext model from scratch and gensim[26] for the first time. We demonstrate how various embedding models show their linear characteristic when we tune the model. We consider the cosine value of each prediction to identify the pure embedding model. The Bengali language presents unique challenges and opportunities for NLP research due to its rich linguistic characteristics and limited resources. While several studies have investigated various aspects of NLP in Bengali, there remains a need for comprehensive research that addresses gaps in the existing literature.

II. Literature Review

There are several studies found, where writers work with the Bengali language, but not everyone has used their own corpus to evaluate the models. Ritu, Nowshin, Nahid, Ismail work with different embedding models in order to find the best among them[21]. They compare the performance of word clustering by word2vec (in gensim and TensorFlow) and fasttext. They use the SUMono[27] set and two other Bengali data sets and merge them. After comparing the word cluster by these three models they found fasttext with SkipGram provides the best result.Nafiz, AkibSadmanee and Md. Iftekhar Tanveer did intrinsic evaluation for their models[20]. They used and compared the performance of two different architectures of wor2vec CBOW and SkipGram. They evaluate their models with Mikolov, Sutskever, Chen, Corrado and Dean[8]questions set by various embedding tasks like, synonym, antonym detection, and concept categorization etc. In their experiment they found, SkipGram showed the best performance for almost every task. However, they only work with two architectures CBOW and SkipGram in wor2vec where there are a couple of other dynamic models for embedding like Fasttext, Glove etc. To evaluate their models, they rely on [9] questions which they translate from English language, not a good example of proper experiment. In questions analogy task they considered the first 20 predicted results as correct, which may produce an inefficient model. They only tune the dimension of the vector in order to get the best performance. A. A. A. Rafat and M. Salehin, Khan, Hossain and Abujarworked with various embedding models to see their performance [22]. They create their own Bengali corpus by crawling various Bengali newspapers. They worked with Wor2vec (CBOW, SkipGram), Glove, Fasttext (CBOW, SkipGram) etc. They tuned their model by changing a couple of parameters. They found fasttext with skip Gram architecture provides the best result with 300 dim and 50 epochs. They evaluate their model with semantic and syntactic tasks. They only show the nearest neighbor for semantic evaluation, however there is no clear evidence of their experiment's result whether it is correct or not. They did not speak about any ground truth that has been used to evaluate their predicted answers. So, it remained unclear how they judge their models with their performance. R. Rahman works with the word2vec model by tuning the hyper-parameter values. Extrinsic and intrinsic evaluation has been done[23]. [23] only explored how dimensions and window size can influence the performance matrix. However, they did not provide details about the questions used for evaluating their models. Their focus was solely on CBOW and SkipGram architectures within the word2vec framework to determine a robust embedding model. This approach overlooks other significant models like FastText and GloVe, which remain unexamined. In reviewing the existing literature different key themes arise. Most of the researchers only work with the word2vec model. However, everyone has some limitations on their work. One of them only works with the wor2vec model and trains the model by merging their data with some public data sets **(**e.g.,sumono[28]**).** They did not provide any light on their experimental process. They only checked the performance to find semantic relationships without any analogy task *(a + b - c = d)* and their word selection was so poor which is an unhealthy way to judge the models, studies by [21][9]. Another writer did the same thing but for multiple embedding models, in addition they have created their own corpus by web crawling to train the models. They worked with multiple hyper-parameters by tuning their values. They have mentioned their evaluation through semantic and syntactic tasks. However, there is no identical sign of their syntactic tasks, for semantic tasks they have chosen some random words and find their nearest neighbor without declaring anything about the assumption [22]. Considering the top most prediction (k = 1) is the ideal way to judge an embedding model. Increasing the k's value may show us a better accuracy however it decreases the evaluation quality. Translating foreign language questions for analogy tasks is another lacking, since a data set conveying its regional information, which directly regulates the questions answering tasks (if a = b then c = ?)[20]. Using Bengali wordnet[29] for semantic task or intrinsic evaluation, where the model trained with corpus carrying Bengali newspaper data, unable to demonstrate its actual capability [23].Capability of wor2vec model and its architecture SkipGram and CBOW have been experimented most of the time [20, 21]. Preparing proper analogy questions for semantic or syntactic tasks is crucial for accurately judging and embedding model. To truly evaluate a model's performance, it is essential to understand the origin of the data being used. This ensures that the assessment reflects the model's actual capabilities.

III. Methodology

In this section we will talk about how we collect the data and data preprocessing methodology. We will also provide a description about the models we have used in our experiment. How we evaluate our models and the creation of semantic and syntactic questions sets.

A. Data collection and reprocessing

In order to train the models, we have collected more than 800k newspaper's articles. For collecting the data, we have made a web scraper by using a fancy tool scrapy. We corroborated that our scrapper did not impede the regular response of the website during scraping, enabling auto throttle mode, it is liable for sending the request according to the website's competence. Data collection methodology through scrappy has been shown in Fig 1.

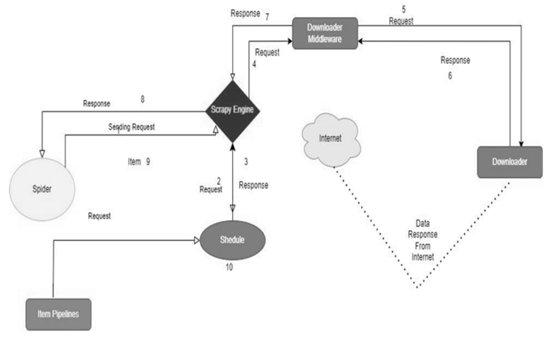

Fig. 1. Dataflow mechanism through scrapy

We have collected every genre data like politics, sports, entertainment, educational etc from the newspapers into JSON format. First, we eliminated the news that had no article at all.

TABLE I. Details of data set

Total Article	891058
Total words	240772592
Total unique words	1604185
Total words after eliminating stop words	177482267
Total unique words without stop words	1603775

Then we tied all the news and prepared a single data set. We remove the noisy data and symbols from our data set, we also erase html tag, unwanted word, multiple spaces, redundant letters, foreign words, letters and numbers to get a monolithic data set. Removing stop-words increases the probability of getting a better embedding relation between words, but we do not lemmatize the data. We wanted to collect the actual relation between words. Lemmatizing can reduce the chances of getting accurate results for syntactic tasks. After truncating the noisy symbol, unwanted data, stop words, we keep rest of them as our data set. Our data set contains 800k articles, and more than 16 lakhs words. We showed more about our data set in Table I We use the python library and regular expressions to clean the data. Before getting the ultimate corpus, we tokenized our data. Instead of providing a tokenized corpus for all models, we provide the raw text file when training the models from scratch, which is more convenient for those. For gensim we used the pickle format to load the tokenized corpus.

B. Construction of Analogy Questions

We developed a set of 133 semantic analogy questions and 103 syntactic questions, referred to as *QS236*, to evaluate models on both semantic and syntactic tasks. The semantic questions cover themes such as politics, family relations, and country-capital relationships, while the syntactic questions explore linguistic aspects such as number, tense, Bengali possession, and nationality. The distribution of these topics within the question sets is illustrated in Fig. 3 and Fig. 4.

1) *Question Compilation Process:* Preparing analogy questions was one of the most challenging tasks. Out of 240 million words in the corpus, we selected only the most frequently used words.

Step 1: We identified topics with the highest probability of occurrence in the corpus. In addition to our own experience, we consulted classmates, university students, and faculty members. Initially, we identified 20 common topics, but after consulting our supervisor and senior researchers, we narrowed this list down to 10.

Step 2: As a reference, we used GloVe's analogy question sets (developed by Stanford University). These sets cover a wide range of topics, including geopolitics, country-capital relationships, family, and grammatical categories like adjectives and superlatives. We compared these topics with our findings to finalize our list of core subjects.

Step 3: We conducted nearest-neighbor searches for selected words across various models, identifying patterns that further helped refine the most frequent topics.

Step 4: Based on all these findings, we compiled both semantic and syntactic question sets, ensuring that the topics aligned with the national interests of Bangladesh, as reflected in local newspaper trends.

2) *Rationale*: These questions were specifically designed to evaluate the models' performance on both semantic topics (e.g., geopolitical issues) and syntactic constructs (e.g., numbers, tenses, adjectives), testing their ability to handle diverse linguistic structures. For instance, we examined whether models could accurately predict: *"China is to Beijing as Russia is to?"* or identify grammatical transformations like *"Teacher is to Teachers as Computer is to Computers."*

Furthermore, we aimed to test the models' capacity to capture the underlying meanings of words and categorize them appropriately, despite biases.Bias in the models typically reflects the inherent trends present in the training data [24]. For example, when testing the analogy *"Bumrah is to India as Africa is to ?,"* the model might predict either *"Kagiso Rabada"* or *"AB de Villiers"* due to corpus bias. While both are South African cricket players, only *"Kagiso Rabada"* would be correct since both Rabada and Bumrah are bowlers, whereas de Villiers is a batsman.
We observed that most models were able to successfully identify these nuanced relationships, demonstrating their ability to capture the latent similarities between words.

C. Embedding models

1) Fasttext: Fasttext is the moderate descendant ofwor2vec.It is the extended version of wor2vec.FastText converts the word into vectors by using n-gram technique. It splits a single word into characters depending on the value of n. The representation will be <a, ar,rm,my,y>, <ma, mac, ach, chi, hin, ine, ne> and <emb, embe, mbed, bedd, eddi, ddin, ding, ing> for the words army, machine and embedding while n= 2, 3 and 4 sequentially. The core difference of the FastText model, instead of working with words it works with n-gram of letters. FastText methodology is almost the same but instead of predicting the word-word occurrences probability it predicts the letters co-occurrence probability. We found FastText can generate unknown words which do not even exist in the vocabulary. For instance, we can generate whole new words like 'বাংলা' (Bangla) and 'দেশ' (Desh) if we have the root word 'বাংলাদেশ' (Bangladesh) in our vocabulary but we will not find those words inside other embedding model's vocabulary, though we trained the models with the same data set.

2) CBOW and SkipGram: CBOW follows the feedforward NNLM model[30]. It predicts the target word when context words are given as input. After predicting the word, it will go one word forward and do the same for finding the word co-occurrences probability based on given context. This forward propagation occurred by inputting the context words as one-hot encoded representation, in a shallow neural network. The complexity of the training for this model is $Q = N \times D + D \times log2(V)$.

In their paper, they have mentioned that the weight matrix of input and the projection layer is shared for all words like NNLM. It is the extended version of Bag of Words. CBOW generates a targeted word after getting the input's one hot representation. For instance, if the window size is set to 5, CBOW selects two words preceding and two words succeeding the target word as input to predict the target word itself. Hence, when analyzing the context for the target word "বাংলা", the input sequence becomes "সোনার", "বাংলা", "তোমায়", "ভালবাসি". Throughout the training process, the CBOW model iterates over the entire corpus, dynamically adjusting its context window for each target word encountered. SkipGram In contrast to CBOW, SkipGram predicts surrounding words given a single word as input. It takes the target word as input and predicts the probability of context words based on the defined window size. SkipGram accepts a single word as input and forecasts the likelihood of context words based on the window size. For the input word "আগুনে," it will anticipate "বেইলি," "রোডে," "নিহত," "পয়তাল্লিশ" as nearby words. The time complexity of SkipGram is given by $Q = C \times (D + D \times log2(V))$, where C is the context window size, N is the number of neurons, and D and V have the same definitions as in CBOW.

3) Glove: GloVe, short for Global Vectors for Word Representation, is a word embedding model introduced by Stanford University. Unlike word2vec, which focuses only on local word context, GloVe considers both local and global word co-occurrence probabilities. It achieves this by constructing a word-word co-occurrence matrix based on a corpus. For instance, consider the sentences "I have my own little cat. You have a little cat." In this corpus, the co-occurrence matrix would identify that the words "have," "little," and "cat" co-occur multiple times. However, GloVe reduces this redundancy by considering each word's co-occurrence with others only once. This matrix captures how often words co-occur in the entire corpus, allowing GloVe to understand both local and global word relationships. By leveraging this comprehensive view of word co-occurrence, GloVe generates vector representations that effectively capture word semantics.

D. Evaluation methods

We compare the actual truth with predicted truth to justify the answer. We evaluated the models with various embedding tasks, performing both semantic and syntactic evaluations.We construct two different ground truths for semantic and syntactic evaluation.To assess semantic performance, we created 133 semantic questions, prompting the models to predict analogous relationships. For instance, given the analogy "America" is to "American" as "Russia" is to? the models were tasked with identifying the semantic relationship between a country and nationality. Similarly, we made 103 questions to explore syntactic relationships, focusing on grammatical structures and parts of speech. For example, "If 'book' refers to 'books', then what does 'cow' refer to?" aimed to evaluate the understanding of grammatical number. In addition to accuracy metrics, we calculated the cosine mean value to further evaluate model performance.

The cosine mean, defined as the sum of cosine similarities for correct predictions divided by the total number of correct predictions, provides a refined assessment of model accuracy, with higher values indicating more accurate predictions. To gain a clear understanding of the models' proficiency, we introduced a Combined Metric (C. Metric) that considers both Cosine Mean (C.M) and correct predictions (C.P). This combined metric allows us to control the relative importance of accuracy versus cosine similarity by adjusting a weight parameter (w). The range of the weight parameter is from 0 to 1, where a lower value emphasizes accuracy over C.M, and a value of 0.5 maintains neutrality. The formulas for these metrics are as follows:

$$\text{Accuracy (A.C)} = \frac{Number\ of\ Correct\ Predictions}{\text{Number Of questions}} \quad (1)$$

$$Cosine\ Mean\ (C.M) = (CosineSumOfCorrectPrediction)/TotalCorrectPrediction \quad (2)$$

$$Combined\ Metric\ (\text{C.Metric}) = Accuracy + w \times (CosineSumOfC.P)/Number\ of\ Correct.\ Predictions \ (3)$$

Furthermore, we analyzed the models' proficiency in concept categorization. Selecting the best-performing model from each embedding technique, we constructed confusion matrices to classify data into distinct categories Table VI and Table VII.**QS236**is conveying different types of questions given in Fig. 3, 4and Table VIII.

To validate the models' predictions, we established separate ground truths for semantic and syntactic evaluations. By comparing the predicted and actual truths, we carefully assessed the models' ability to discern semantic and syntactic relationships accurately.

IV. EXPERIMENT AND RESULT

In this part we dive deeply into our actual work. We discuss every nuance of experimental changes and their impact on results.

A. Machine Setup

Our experimentation was conducted on a local machine equipped with 16 GB of RAM, an i7 CPU, and a built-in 512 SSD. We operated across two different operating systems, namely Windows and Linux. Specifically, we utilized the Linux OS, leveraging the Ubuntu terminal, to train the Glove model from scratch. For all other model training tasks, we opted Windows OS. We utilized Jupyter Notebook via Anaconda in Python to facilitate various aspects of our experimentation journey. We release our spider to crawl the newspaper through the Pycharm IDE. Data cleaning, Model training, model's evaluation, performance demonstration, everything happened on Jupyter-notebook where we used libraries like Pandas data frame, regular expression, matplotlib, NumPy etc. We compared the prediction with our ground truth by using Microsoft excel and Google sheet, in addition we also counted the summation of correctly predicted pairs and mean of cosine similarity with the same software.

B. Elimination between CBOW and SkipGram

Working with large text datasets demands significant time and computational power, making the identification of a robust embedding model, suitable architecture, and optimal hyper-parameters challenging. Since CBOW and SkipGram architectures are applicable for word2vec and FastText models, training models with the same parameters twice is very time-consuming. To pace our study, we decided to eliminate either CBOW or SkipGram on the basis of their performance. Therefore, we trained the models using identical parameters (window size 5, epoch 5, dimensions 100 and 300, and minimum counts 5 and 10) for both architectures. Our evaluation used custom-made question pairs including 133 semantic analogy questions and 103 syntactic analogy questions. Semantic tasks assess a model's ability to detect semantic relationships like synonyms, while syntactic tasks evaluate parts of speech, which are critical for named entity recognition. We trained the dataset using:

- FastText from scratch,
- FastText with the gensim library,
- Word2vec with the gensim library.

Additionally, we trained the GloVe model. We found that both architectures provided very close results. SkipGram architecture is more computationally expensive, while CBOW is more efficient in terms of training time. However, SkipGram generally provided more accurate results than CBOW in semantic and syntactic analogy tasks for most cases. Therefore, we emphasized correctly predicted pairs to identify a suitable architecture for our subsequent experiments.

We observed that SkipGram provided the best results for every model (word2vec, FastText from scratch, and gensim FastText), except for syntactic tasks using the Facebook FastText implementation. Training time was consistently longer for SkipGram compared to CBOW. Given that SkipGram gave the best results, therefore we chose to eliminate CBOW for further experiments.

C. Hyper-parameters effect on models' performance

1) Epoch: Performance of the embedding models highly dependent on epoch. We examine the model's performance by providing various epoch's size each time. Every time we complete a full training process, we increase the epoch size. We started from epoch = 1 and terminated at 25. We used epochs = 1, 5, 15, 20, 25, 35 etc. We saw that rising epoch size enhancing the accuracy shown in Table III and V. However, the phenomenon continued till a point and after reaching the end point it started decreasing shown in Fig. 2. Each epoch for a large data set model is very costlier. So enhancing epoch means raising the time complexity. Therefore, we need to choose the best fit of the epoch, we found epoch 25 as the best fit for our models. Though the consumption of training time is also dependent on various parameters like minimum count, dimension etc. Enhancing the dimension means it will take more time than a lower dimensional training on the other hand increases of the minimum count is the cause of less training time.

2) Minimum count: The parameter minimum count (min) indicates a word's occurrence inside the corpus should be greater or equal to the minimum count's value (word>= min) for being considered as training data. Higher minimum count reduces the vocabulary size of the model which causes less time complexity. We experiment with the minimum count's value 5,10,15,20,25,35 and 50.

When we increase the minimum-count it produces best results with less vocabulary size but less min value produces moderately good results with higher vocabulary size. We got the best semantic tasks result with higher min value but when we performed syntactic tasks for facebook fasttext we found that most of the time models with less minimum count showed the satisfactory result.

Though increasing the minimum-count parameter enhances the model's accuracy but also reduces the size of the vocabulary Table II showed the different amount of unique words are being used in training time for minimum count 5, 10, 25 and 35 respectively.

Models like FastText may overcome this situation since it can produce new words but wor2vec and GloVe are unable to do so. Therefore, they may not find some words inside the vocabulary at the time of evaluation which decreases their purity. Table III and V tell, how minimum count's values are dominating the prediction ratio.

TABLE II. TRUNCATED VOCABULARY AT MINIMUM COUNT

Truncated vocabulary at min count 5 yields	374,604 words
Truncated vocabulary at min count 10 yield	239117 words
Truncated vocabulary at min count 25 yield	139876 words
Truncated vocabulary at min count 35 yields	115712 words

3) *Dimension:* We trained the models with two different dimensions. The accuracy of the models for 100 and 300 dimensions is slightly different. The training time is higher for 300 dimensions than 100. However, we noticed some interesting facts that no one talked about before. Though most of the time the models with 300 dimensions gave us the best accuracy in terms of correctly prediction. But the sum of cosine values of correctly predicted answers remains greater for 100 dimensions, most of the time.

4) Window size and Negative sampling: We trained wor2vec model with standard parameters we found so far and provide window sizes 5, 7 and 9. The model shows the best performance with 5, therefore, for rest of the training we fixed this parameter. Negative sampling is faster than Hierarchical SoftMax (hs), also provide far better result than that of hs. Which are shown in Fig 6 and Fig 7.

B. Models' performance analysis

1) FastText from Scratch: To train the FastText model we provide .txt file. It does not require any tokenize data since the FastText model has its built-in tokenizer. The model spends a huge time for being trained. The model allocates a portion of memory and CPU in order to train itself. The model demonstrates its adroitness in both semantic and syntactic task. Fasttext model generates some unwanted words if we consider a less minimum count (5) value.

It generates 'বাংলাদেশপাকিস্তান' instead of 'বাংলাদেশ' and 'পাকিস্তান'. The model provides 76.69%, 78.95 % accuracy and 80.81, 73.19 cosine value in semantic analogy tasks for 100 and 300 dimensions respectively at minimum count 35. The average cosine mean value was 0.79 and 0.73. In syntactic analogy tasks the model achieves 83.50 and 91.26 percent accuracy, while the cosine and cosine-mean values become 67.60, 64.68 and 0.786, 0.69 for 100 and 300 dimensions respectively.

2) Gensim FastText: Gensim with the FastText library does not support corpus.txt file, instead it requires a tokenized list as a corpus. Training time is faster compared to training from scratch.

The model exhibited accuracies of 74.44% and 77.44% for 100 dimensions, with cosine values of 81.76 and 76.33, and cosine mean values of 0.83 and 0.74 respectively. For 300 dimensions, accuracy increased to 77.44%, although cosine values decreased slightly.

3) Gensim Word2vec: Wor2vec with gensim is the fastest model however it requires tokenized data. Tokenizing is time consuming for a large data set like ours. Wor2vec provide the best result in semantic analogy tasks.

It secure 109, 110 correct prediction and the accuracy becomes 81.95, 82.71 percent for 100 and 300-dimensions models.

TABLE III. SEMANTIC ANALOGY RESULTS

	Semantic											
	dim 100							dim 300				
	Epoch	Minimum count	C.P	AC%	cosine	C.M	C.Metric	C.P	AC%	cosine	C.M	C.Metric
facebook fasttext	**25**	**35**	**102**	**76.69**	**80.81**	**0.7923**	**76.93**	**105**	**78.95**	**73.19**	**0.70**	**79.16**
	20	25	100	75.19	79.69	0.7969	75.43	99	74.44	69.69	0.70	74.65
	15	20	96	72.18	76.55	0.7974	72.42	100	75.19	70.46	0.70	75.4
	5	10	84	63.16	66.96	0.7972	63.40	79	59.40	55.41	0.70	59.61
	5	5	78	58.65	63.17	0.8098	58.89	83	62.41	59.52	0.72	62.626
gensim fasttext	**25**	**35**	**99**	**74.44**	**81.76**	**0.8258**	**74.69**	**103**	**77.44**	**76.33**	**0.74**	**77.662**
	20	25	92	69.17	76.24	0.8287	69.42	102	76.69	75.48	0.74	76.912
	15	20	95	71.43	78.21	0.8233	71.68	101	75.94	74.59	0.74	76.162
	5	10	85	70.54	70.54	0.8299	70.79	90	67.67	66.89	0.74	67.892
	5	5	83	69.22	70.84	0.8535	69.48	93	69.92	69.05	0.74	70.142
gensim word2vec	**25**	**35**	**109**	**81.95**	**87.31**	**0.8010**	**82.19**	**110**	**82.71**	**75.16**	**0.68**	**82.914**
	20	25	108	81.20	87.35	0.8088	81.44	108	81.20	73.63	0.68	81.404
	15	20	109	81.95	86.52	0.7937	82.19	108	81.20	73.00	0.68	81.404
	5	10	104	78.20	82.68	0.7950	78.44	107	80.45	72.32	0.68	80.654
	5	5	98	73.68	78.49	0.8009	73.92	102	76.69	69.52	0.68	76.894
GloVe	**25**	**35**	**99**	**74.44**	**70.78**	**0.7150**	**74.65**	**96**	**72.18**	**56.173**	**0.59**	**72.357**
	20	25	98	73.68	72.66	0.7415	73.90	97	72.93	57.126	0.59	73.107
	5	10	91	68.42	66.8675	0.7348	68.64	98	73.68	56.8969	0.58	73.854
	5	5	98	73.68	72.1529	0.7363	73.90	96	72.18	50.86	0.53	72.339

The cosine mean values are 0.80 and 0.68. In syntactic analogy it provides, accuracy: 56.31%, cosine: 43.25 and cosine-mean: 0.716 for 100 dimensions and for 300 dimensions it becomes 62.14%, cosine 39.03 and cosine mean0.61 which indicating potential for improvement.

4) Gensim Word2vec: differs in several aspects from other models, supporting raw text data for training and incorporating a built-in tokenizer. It exhibited faster training times and competitive accuracies of 74.44% and 72.18% for 100 and 300 dimensions respectively. However, in syntactic tasks, accuracy remain same for both 100 and 300 dimensions but the cosine value remain lower for 300 dimensions. GloVe's performance lagged behind other models, indicating areas for enhancement.

C. Compared Between Gensim and Scratch

Model trained from scratch and using gensim frame work demonstrate different performance. In order to reveal the consequence, we trained the FastText model through both frameworks. We found that the model trained from scratch takes much time than gensim but it burns less memory than gensim. The model trained from scratch provides better results for both semantic and syntactic analogy tasks. However, in almost every case model trained with gensim, produced higher vector values than that of scratch.

TABLE IV. GENSIM VS SCRATCH

memory	Training Time	C.M		C.P	
		semantic	syntactic	semantic	syntactic
<gensim	79346.55	0.7923	0.786	102	86
	194938.2	0.70	0.69	105	94
>scratch	29888.34	0.82	0.8	99	76
	50195.89	0.74	0.71	103	91

D. Result Analysis

When summarizing the results, we observed that increasing epoch size improves the quality of results up to a point. Higher dimensions enhance the number of correctly predicted pairs, but lower dimensions yield better cosine values. Increasing the minimum count value produces better embedding results but reduces vocabulary size, possibly leading to missing words.

TABLE V. SYNTACTIC ANALOGY RESULTS

			syntactic									
			100					300				
	Epoch	Minimum count	correct prediction	AC%	cosine	C.M	C.Metric	C.P	cosine	AC%	C.M	C.Metric
Facebook_ fasttext	**25**	**35**	**86**	**83.50**	**67.60**	**0.786**	**83.73**	**94**	**64.68**	**91.26**	**0.69**	**91.47**
	20	25	77	74.76	61.06	0.793	75.00	91	63.47	88.35	0.70	88.56
	15	20	77	74.76	61.39	0.797	75.00	85	60.83	82.52	0.72	82.74
	5	10	77	74.76	62.28	0.809	75.00	79	57.55	76.70	0.73	76.92
	5	5	66	64.08	53.94	0.817	64.32	80	59.50	77.67	0.74	77.89
Gensim_ fasttext	**25**	**35**	**76**	**73.79**	**61.03**	**0.803**	**74.03**	**91**	**64.99**	**88.35**	**0.71**	**88.56**
	20	25	79	76.70	64.20	0.813	76.94	90	65.23	87.38	0.72	87.60
	15	20	73	70.87	69.59	0.953	71.16	90	65.82	87.38	0.73	87.60
	5	10	70	67.96	58.47	0.835	68.21	75	57.06	72.82	0.76	73.04
	5	5	71	68.93	61.04	0.860	69.19	77	59.12	74.76	0.77	74.99
Gensim_w ord2vec	**25**	**35**	**58**	**56.31**	**43.25**	**0.746**	**56.53**	**64**	**39.03**	**62.14**	**0.61**	**62.32**
	20	25	59	57.28	44.42	0.753	57.51	66	40.29	64.08	0.61	64.26
	15	20	60	58.25	44.76	0.746	58.48	67	40.63	65.05	0.61	65.23
	5	10	61	59.22	46.09	0.756	59.45	59	36.97	57.28	0.63	57.47
	5	5	57	55.34	43.61	0.765	55.57	62	38.64	60.19	0.62	60.38
glove	**25**	**35**	**50**	**48.54**	**33.59**	**0.672**	**48.75**	**50**	**26.61**	**48.54**	**0.53**	**48.70**
	20	25	48	46.60	32.20	0.671	46.80	48	25.21	46.60	0.53	46.76
	5	10	45	43.69	30.55	0.679	43.89	40	21.26	38.83	0.53	38.99
	5	5	37	35.92	25.38	0.686	36.13	36	19.52	34.95	0.54	35.11

Models trained with the gensim library took less time than those trained from scratch. SkipGram models slightly outperformed CBOW in embedding tasks but with higher time complexity. Each model excelled in different tasks, with no single model consistently best across all evaluation methods. There were no fixed parameters that provided the best results every time.

However, after extensive experimentation, we found that 300 dimensions, 25 epochs, and a minimum count of 35 gave the best results for most models shown in Fig 5. For syntactic analogy tasks, the Facebook FastText model with SkipGram architecture provided the best results, predicting the maximum correct answers. Gensim FastText, gensim word2vec, and GloVe followed as the second, third, and fourth best, respectively. For semantic analogy tasks, gensim word2vec performed the best, followed by FastText (scratch) and gensim FastText with the same architecture and parameters. Notably, GloVe provided better results with 100 dimensions instead of 300(Table III and V) However if we look at the model's purity; we would see a different picture. While models with 300 dimensions gave the best predictions for both semantic and syntactic tasks, models with 100 dimensions consistently showed higher cosine mean values. Therefore, a combined metric has been created in order to consider both cosine mean and accuracy. Since we want to focus on correct predictions rather than prediction quality in our case study, **we set the weight value (w) to 0.3.** According to the equation mentioned earlier, a lower weight value emphasizes accuracy over cosine value, while a higher weight value emphasizes cosine value over accuracy.

Although this combined metric caused some minor changes in results, the top scorer for each model remained unchanged. In the results analysis, it is evident that FastText from Scratch, Gensim FastText, Gensim Word2Vec, and GloVe exhibited varying degrees of performance across semantic and syntactic analogy tasks. Across semantic analogy tasks, the models showcased competitive accuracies ranging from 74.44% to 81.95% for 100 dimensions and from 72.18% to 82.71%for 300 dimensions.

However, in syntactic analogy tasks, the performance varied significantly among the models, with accuracies ranging from 48.54% to 83.50% for 100 dimensions and from 48.54% to 91.26% for 300 dimensions. Despite differences in training methodologies and implementation details, each model demonstrated its ability to capture semantic relationships effectively. Gensim Word2Vec achieved the best semantic accuracy of 82.71% for 300 dimensions, while FastText from Scratch secured the highest syntactic accuracy of 91.26% for 300 dimensions. Conversely, GloVe yielded the lowestaccuracy of 48.54% in syntactic tasks with 100 dimensions. The models consistently performed best when trained with 25epochs and a minimum count of 35. Further optimization may be required to enhance performance in capturing syntactic relationships.

TABLE VI. CONFUSION MATRIX OF CONCEPT CATEGORIZATION

models	Country capital	political	family	sports	object	nationality	Tense
country capital	36	0	0	0	0	0	0
politics	0	45	0	0	0	0	0
family	0	0	25	0	0	0	0
sports	0	0	0	24	0	0	0
object	0	0	0	0	34	0	0
nationality	0	0	0	0	0	37	0
tense	0	0	0	0	0	0	30

TABLE VII. CONFUSION MATRIX OF CONCEPT CATEGORIZATION GLOVE

glove	country capital	politics	family	sports	object	nationality	tense
country capital	36	0	0	0	0	0	0
politics	0	45	0	0	0	0	0
family	0	0	25	0	0	0	0
sports	0	0	0	24	0	0	0
object	0	0	0	0	20	0	0
nationality	0	0	0	0	0	37	0
tense	0	0	0	0	0	0	27

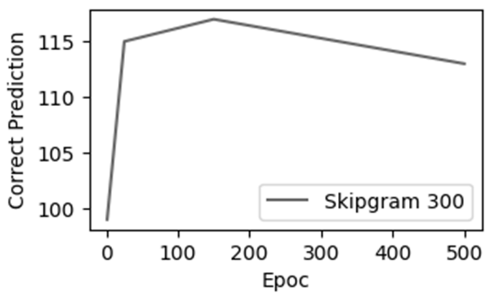

Fig. 2. Effect of increasing epochs on model accuracy

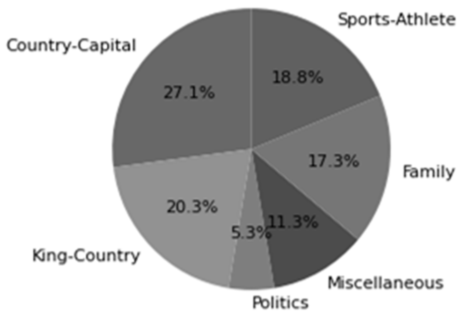

Fig. 3. Types of questions in semantic analogy tasks

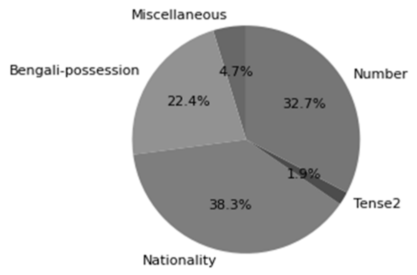

Fig. 4. Types of questions in syntactic analogy tasks

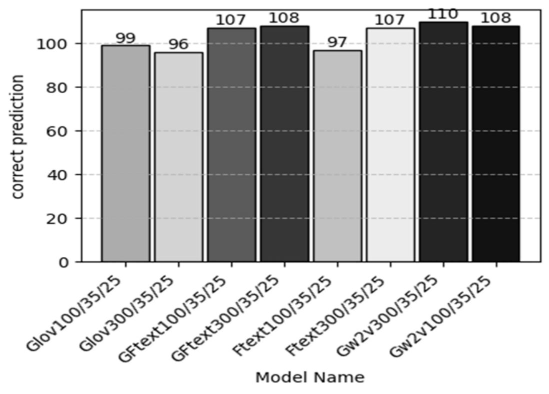

Fig. 5. Prediction comparison across models for semantic tasks

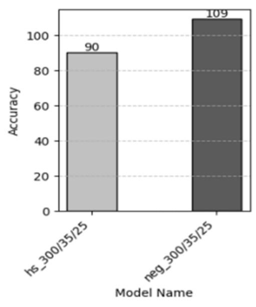

Fig. 6. Performance comparison of hierarchal-softmax and negative sampling using identical parameters

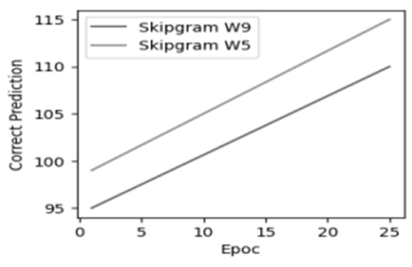

Fig. 7. Impact of window size effect on wor2vec model performance

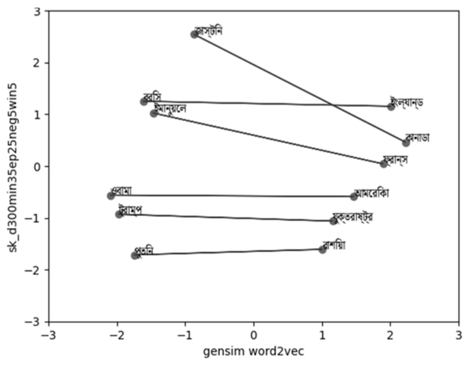

Fig. 8. Vector representation of semantic words in 2d embedding space for gensim word2vec

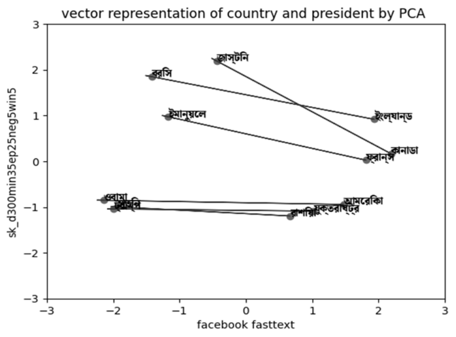

Fig. 9. Vector representation of semantic words in 2d embedding space for fasttext from scratch

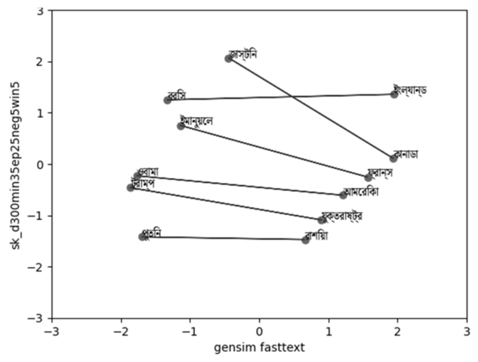

Fig. 10. Vector representation of semantic words in 2d embedding space for gensim-fasttext

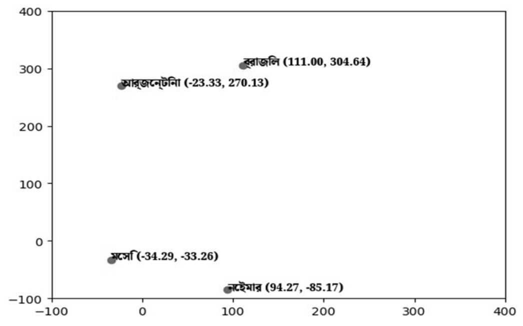

Fig. 11. Correctly predicted pairs in 2d embedding space

TABLE VIII. EXAMPLE OF ANALOGY QUESTIONS

Country – Capital			**Number**	
Country - Capital	চিন - বেইজিং (China - Beijing)	রাশিয়া - মস্কো (Russia - Mosko)	**Singular - Plural**	স্কুল-স্কুলগুলো (School- Schools)
Politics			**Bengali Possession**	
Political parties	আওয়ামীলীগ–ছাত্রলীগ Awamilegue – S.Legue	বিএনপি–ছাত্রদল BNP – Student	Adjective	শিক্ষক–শিক্ষকের (Teacher – Teacher’s)
Country - president	রাশিয়া–পুতিন Russia - Putin	যুক্তরাষ্ট্র–ট্রাম্প U.S - Trump	Adjective	বিদ্যালয়ের – বিদ্যালয় (School’s - School)
Game	**Football** -	**Cricket**	**Tense**	
Athlete - Country	মেসি - আর্জেন্টিনা (Messi - Argentina)	নেইমার - ব্রাজিল (Neymar - Brazil)	Past continuous -Simple Past	যাচ্ছিলো – গিয়েছিল(was going- went)
Country - Bowler	ভারত– বুমরা (India - Bumrah)	আফ্রিকা - রাবাদা (Africa - Rabada)	Past continuous -Simple Past	খাচ্ছিলো–খেয়েছিল(was eating - ate)
Family			**Country(n)-Nationality(adj)**	
Father - Mother	মেয়ে– মা (Daughter - Mother)	ছেলে– বাবা (Son - Father)	Country-Nationality	জাপান–জাপানিজ (Japan-Japanese)
Gender - Power	মেয়ে - রাজকুমারী (Lady - Princes)	ছেলে - রাজকুমার/প্রিন্স (Boy - Prince)	Nationality- Country	কানাডিয়ান–কানাডা (Canadian - Canada)
Power - Gender	রাজা - পুরুষ (King - Man)	রানি - নারী (Queen - Woman)	Nationality- Country	রাশিয়ান–রাশিয়া (Russian-Russia)

V. Conclusions

Textual data is essential in the era of AI, but working with textual data can be challenging, especially for foreign languages like Bengali. Word embedding is a technique that converts texts into vectors, which contain the semantic and syntactic relations of the words. It says the similar words should stay together in the embedding space which is demonstrated in Fig 6, Fig 7, Fig 8 and Fig 9. In this study, we aimed to share the experimental results of different embedding models and how they perform for different tasks by selecting the model architecture and tuning parameters like the epoch, dimension, and minimum count. We trained the models with our own corpus, containing almost 900k newspaper articles. To evaluate our models, we created 133 semantic and 103 syntactic analogy questions, and we considered only the top prediction as the correct answer. Our results showed that increasing the epoch size enhances the pure prediction pairs up to a certain point. Increasing the minimum count value produces good results but decreases the vocabulary size. Higher dimensions (300) provide more accurately predicted pairs than lower dimensions (100), except for the Glove model. However, if we consider the cosine value, lower dimensions (100) produce more pure words than higher dimensions (300). Models with the SkipGram architecture gave the best results, but CBOW spent less time in training. Hierarchical SoftMax is more costly than negative sampling and gives poorer results. FastText models can introduce new words from a complex word, which may be useful for higher minimum count values, and it maximizes vocabulary size. Glove and word2ve have faced vocabulary shortage with a higher minimum count value.

GLOVE demonstrates better results for lower dimensions. Overall, there is no specific set of parameters that can produce the best result for every case. Instead, the accuracy depends on the dataset's size, quality, source, and other factors. However, we can find the best results by tuning the parameters since the accuracy increases linearly across the models. In our study, we found that the model with the parameters minimum count 35, epoch 25, dimension 300 (Glove 100), window size 5, and negative sampling 5 provides the best results for each model except Glove. Among all models, Gensim word2vec with SkipGram predicts 92% correct answers for semantic and FastText from scratch predicts 90% accurate answers for syntactic tasks and for conceptual category tasks we achieve 100% of accuracy. Gensim FastText gave the best cosine mean value for both semantic (0.74) and syntactic (0.71) tasks. Creating question sets for evaluation is one of the most crucial parts of getting better embedding results.

In the future, we plan to enlarge our corpus and questions by adding diverse data to perform training and analogy tasks again with the best two embedding models we have found. We also plan to compare contextual (word2vec) and non contextual (Bert, Elmo) models to investigate their performance in various NLP tasks.

Acknowledgement

First, we express our heartiest thanks and gratefulness to almighty God for His divine blessing makes it possible to complete the research successfully. We are grateful and wish our profound indebtedness to Dr. Sheak Rashed Haider Noori, Professor, Department of CSE Daffodil International University, Dhaka. Deep Knowledge & keen interest of our supervisor in the field of “Deep Natural Language Processing” to carry out this research project. His endless patience, scholarly guidance, continual encouragement, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project. We would like to express our heartiest gratitude to Dr. Sayed Akhter Hossain, Professor, and Head**,** Department of CSE, for his kind help to finish our project and to other faculty members and the staff of CSE department of Daffodil International University. We would like to thank our entire course mate in Daffodil International University, who took part in this discussion while completing the course work. Finally, we must acknowledge with due respect the constant support and patients of our parents.

VI. Reference

[1] G. E. H. &. R. J. W. David E. Rumelhart, "Learning representations by back-propagating errors," *Nature* , p. pages533–536, 09 October 1986.

[2] J. L. Elman, "Finding Structure in Time," *cognitive science a multidisciplinary journal,* vol. 14, no. 2, pp. 179-211, March 1990.

[3] D. R. J. A. F. James McClelland, Parallel distributed processing: Explorations in the microstructure of cognition., vol. 1, MIT Press, Cambridge, Massachusetts, 1986.

[4] K. C. G. C. J. D. Tomas Mikolov, "Efficient Estimation of Word Representations in Vector Space," in *International Conference on Learning Representations*, January 2013.

[5] L.-A. R. Y. B. Joseph Turian, "Word Representations: A Simple and General Method for Semi-Supervised Learning," in *Association for Computational Linguistics*, Uppsala, Sweden, 2010.

[6] J. W. I. &. C. Ronan Collobert, "A unified architecture for natural language processing: deep neural networks with multitask learning," in *25th international conference on Machine learning*, 05 July 2008.

[7] J. W. L. B. M. K. K. K. P. K. Ronan Collobert, "Natural Language Processing (Almost) from Scratch," *The Journal of Machine Learning Research,* vol. 12, pp. 2493 - 2537, 01 November 2011.

[8] I. S. K. C. G. S. C. J. D. Tomas Mikolov, "Distributed Representations of Words and Phrases and their Compositionality," in *26th International Conference on Neural Information Processing Systems*, October 2013.

[9] T. M. Quoc Le, "Distributed Representations of Sentences and Documents," in *31st International Conference on Machine Learning*, 2014.

[10] E. G. A. J. a. T. M. Piotr Bojanowski, "Enriching Word Vectors with Subword Information," *Transactions of the Association for Computational Linguistics,* vol. 5, p. 135–146, 2017.

[11] R. S. a. C. M. Jeffrey Pennington, "GloVe: Global Vectors for Word Representation," in *2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014.

[12] G.-L. D. a. M.-Y. J. JIN-KAI ZHANG, "Stock Investors' Preferences on Stock Forum Topics Based on FNS-LDA2vec," 04 September 2023.

[13] M.-W. C. K. L. K. T. Jacob Devlin, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *arXiv:1810.04805* , October 2018.

[14] M. N. M. I. M. G. C. C. K. L. L. Z. Matthew E. Peters, "Deep Contextualized Word Representations," in *2018 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies, New Orleans, Louisiana, 2018.

[15] S. A.-M. J. D. A. M. N. Md. Nawab Yousuf Ali, "Morphological analysis of Bangla words for Universal Networking Language," in *Third International Conference on Digital Information Management(ICDIM)*, 2008.

[16] M. R. A. Adnan Ahmad, "Bengali word embeddings and it's application in solving document classification problem," in *2016 19th International Conference on Computer and Information Technology (ICCIT)*, Dhaka, Bangladesh, 2016.

[17] A. Khatun, A. Rahman, M. S. Islam and Marium-E-Jannat, "Authorship Attribution in Bangla literature using Character-level CNN," in *2019 22nd International Conference on Computer and Information Technology (ICCIT)*, Dhaka, Bangladesh, 2019.

[18] S. H. Sumit, M. Z. Hossan, T. A. Muntasir and T. Sourov, "Exploring Word Embedding for Bangla Sentiment Analysis," in *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, Sylhet, Bangladesh, 2018.

[19] M. A. H. I. M. S. I. Hemayet Ahmed Chowdhury, "A Comparative Analysis of Word Embedding Representations in Authorship Attribution of Bengali Literature," in *International Conference on Communications and Information Technology*, 1 December 2018.

[20] A. S. M. I. T. M. A. A. Nafiz Sadman, "Intrinsic Evaluation of Bangla Word Embeddings," in *International Conference on Bangla Speech and Language Processing (ICBSLP)*, Sylhet, Bangladesh, 28 September 2019.

[21] N. N. M. M. H. N. S. I. Zakia Sultana Ritu, "Performance Analysis of Different Word Embedding Models on Bangla Language," in *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, September 2018.

[22] A. A. A. Rafat, M. Salehin, F. R. Khan, S. A. Hossain and S. Abujar, "Vector Representation of Bengali Word Using Various Word Embedding Model," in *2019 8th International Conference System Modeling and Advancement in Research Trends (SMART)*, Moradabad, India, 16 June 2020.

[23] R. Rahman, "Robust and Consistent Estimation of Word Embedding for Bangla Language by fine-tuning Word2Vec Model," in *2020 23rd International Conference on Computer and Information Technology (ICCIT)*, DHAKA, Bangladesh, 21 December 2020.

[24] M. Mars, "From Word Embeddings to Pre-Trained Language Models: A State-of-the-Art Walkthrough," *Applied Sciences ,* vol. 12, no. 17, p. 8805, 1 September 2022.

[25] A. M.-R. F. M.-S. a. L. A. U.-L. Ismael Garrido-Muñoz, "A Survey on Bias in Deep NLP," *Applied Science,* 02 04 2021.

[26] P. S. Radim Řehůřek, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, May 2010.

[27] A. A. M. S. M. R. S. a. M. Z. I. Md. Abdullah Al Mumin, "SUMono: A Representative Modern Bengali Corpus," *SUST Journal of Science and Technology,* vol. 21, pp. 78-86, 2014.

[28] M. A. A. Mumin, "SUMono: A Representative Modern Bangla Corpus," 24 4 2018. [Online]. Available: https://github.com/maamumin/SUMono.

[29] soumenganguly, "Bangla-Wordnet," 30 4 2016. [Online]. Available: https://github.com/soumenganguly/Bangla-Wordnet.

[30] T. M. Quoc Le, "Distributed Representations of Sentences and Documents," in *31st International Conference on Machine Learning, PMLR*, 2014.