Iraqi Journal of Science, 2022, Vol. 63, No. 3, pp: 1349-1361
DOI: 10.24996/ijs.2022.63.3.37

ISSN: 0067-2904

Word Embedding Methods for Word Representation in Deep Learning for Natural Language Processing

Md. Anwar Hussen Wadud[1]*, M. F. Mridha[2], Mohammad Motiur Rahman[3]
[1]*Department of Computer Science and Engineering*
[1,2]*Bangladesh University of Business and Technology, Dhaka, Bangladesh*
[1,3]*Mawlana Bhashani Science and Technology University, Tangail, Bangladesh*

Received: 1/2/2021 Accepted: 21/5/2021

Abstract

Natural Language Processing (NLP) deals with analysing, understanding and generating languages likes human. One of the challenges of NLP is training computers to understand the way of learning and using a language as human. Every training session consists of several types of sentences with different context and linguistic structures. Meaning of a sentence depends on actual meaning of main words with their correct positions. Same word can be used as a noun or adjective or others based on their position. In NLP, Word Embedding is a powerful method which is trained on large collection of texts and encoded general semantic and syntactic information of words. Choosing a right word embedding generates more efficient result than others. Most of the papers used pretrained word embedding vector in deep learning for NLP processing. But, the major issue of pretrained word embedding vector is that it can't use for all types of NLP processing. In this paper, a local word embedding vector formation process have been proposed and shown a comparison between pretrained and local word embedding vectors for Bengali language. The Keras framework is used in Python for local word embedding implementation and analysis section of this paper shows proposed model produced 87.84% accuracy result which is better than fastText pretrained word embedding vectors accuracy 86.75%. Using this proposed method NLP researchers of Bengali language can easily build the specific word embedding vectors for word representation in Natural Language Processing.

Keywords: Word embedding; NLP; FastText; Deep Learning, local and pretrained word vector.

1. Introduction

Word embedding is the most important topic in natural language processing, known as distributed word representation to represent words in natural language processing and information retrieval applications [1- 6]. Most of the Machine Learning algorithms also Deep Learning Algorithms cannot process string or plain text in their raw form. Their input requires numbers or list of numbers to perform any type of job such as regression, classification etc. Word Embedding generally map a word to a vector using a dictionary.

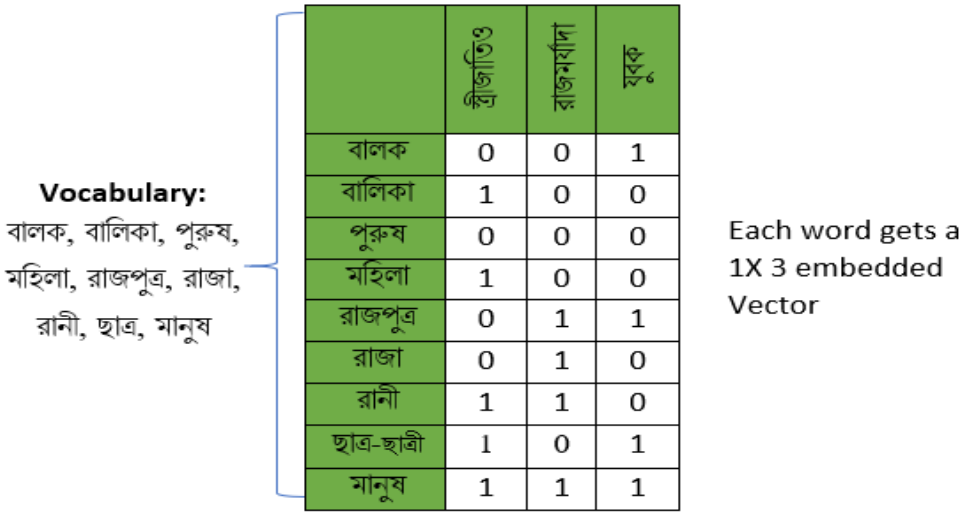
Word Embeddings or Distributional vectors follow distributional hypothesis where similar context occur according to similar meaning of all words. Distributional vectors try to capture the properties of the adjacent word. Generally, word embedding used as the first data processing layer in machine learning or deep learning model [7- 11].

* Email: mahwadud@gmail.com

	1	2	3	4	5	6	7	8	9
বালক	1	0	0	0	0	0	0	0	0
বালিকা	0	1	0	0	0	0	0	0	0
পুরুষ	0	0	1	0	0	0	0	0	0
মহিলা	0	0	0	1	0	0	0	0	0
রাজপুত্র	0	0	0	0	1	0	0	0	0
রাজা	0	0	0	0	0	1	0	0	0
রানী	0	0	0	0	0	0	1	0	0
ছাত্র	0	0	0	0	0	0	0	1	0
মানুষ	0	0	0	0	0	0	0	0	1

Figure 1-Simple Word Embedding Scheme for nine-word vocabulary

Valid word vector can be any set of numbers where vocabularies capture specific meanings, relationship between words etc. In word embedding, every word has a unique number of vector and embeddings are multidimensional vectors typically 50 to 500 in length. Simplest word embedding scheme is one-hot encoding where embedding space has the same size as total number of words in the vocabulary as shown in Figure 1. Main drawback of one-hot embedding is dimension size linearly depends on vocabulary size which consume huge amount of memory. If vocabulary size increase, then linearly increase number of dimensions. To reduce the dimension size, we can use categories for all vocabulary as shown in Figure 2, where similar word has similar embeddings and embedding matrix is less empty space or zeros. Creating N dimensional word embedding vector which contains relationship between similar word and neighbour's word is very difficult for most of researcher. They use pretrained word embedding vectors like as Word2Vec created from Google, Glove word embedding vector created from Stanford or fasttext word embedding vector created form Facebook. This pretrained word embedding vector have their own algorithm to create this vector. During creating word embedding locally, we have to consider similarity between recognize words and relationship between words.



	স্ত্রীজাতিত্ব	রাজবাড়ী	যুবক
বালক	0	0	1
বালিকা	1	0	0
পুরুষ	0	0	0
মহিলা	1	0	0
রাজপুত্র	0	1	1
রাজা	0	1	0
রানী	1	1	0
ছাত্র-ছাত্রী	1	0	1
মানুষ	1	1	1

Figure 2-Lower dimensional Word Embedding

The main goal of this research is to discuss how to create local word embedding methods for Bengali text processing. Word Embedding methods in English language processing is very

familiar but in Bengali language usages of this methods is very rare. The key contributions of our research are:

- Identify all stop words in Bengali language
- Perform frequency based local word embedding generation process in Bengali language
- Perform prediction based local word embedding generation process
- Apply Random Forest machine learning classifiers on local word embedding vector and pretrained word embedding vector
- Made a Comparison between local and pretrained word embedding vector.

2. Related Works

Hinton [7], 1986 proposed first linear relational word representation scheme using binary relations between words by leaning distributed representation. Then other researchers improved word embedding vectors by adding different factors. Mikolov et al. (2013c) [2, 12] proposed a word embedding method based on input layer weights which captures syntactic and semantic scheme and relation-specific vector offset used to characterized semantic relationship. Their given example demonstrates male-female relationship learned by representing word vector by vector equation "king – queen = man – woman". Hermann and Blunsom work for multilingual setting from distributional representation of an input sentences that is same sentences in different language [13, 14, 15].

Kiros [16] explained different notation of learning from context sentence using recurrent neural network. Yih [17] proposed a method for text similarity measure where short texts are represented by TF-IDF vectors. Hill [18] also present a liner model where relationship between sentences is not consider. Global vectors for word representing (GloVE) is another vector representation for words based on unsupervised learning algorithm which is crated for English language [19]. Most of the pretrained word embeddings are based on English language. Recently Facebook introduced new word embeddings named fastText which have word vector for 157 languages [20]. FastText also have word vector for Bengali language and each vector have 300 dimensions in size. But the major problem of using fastText is large data size and need huge memory and high configurable machine for processing also have limited number of Bengali words. All of word embedding related research used pretrained word embedding vectors or recurrent neural network to skip word pre-processing or used own word embedding with limited properties. So, in this paper word embedding creating process has been discussed for the Bengali language also discussed how to use pretrained word vector for the Bengali language.

Kumar et. al [21] discussed pretrained word embeddings for 14 languages including Bengali language. They apply different word embedding methods on 14 languages and find out their performance result but they didn't discuss the local embedding generation process for all languages. They use pretrained word embedding methods to generate word vector for specific language.

3. Word Embedding Method

Creating local Word Embedding methods depends on aims of our NLP research. Word Embedding methods can be broadly classified into two categories- Frequency based Embedding and Prediction based Embedding. Frequency based embedding methods are easy to understand and mainly used for text classification, sentiment analysis and many more [22]. There have several frequency-based methods such as Count Vector, TF-IDF Vector, Co-occurrence Matrix etc. On the other hand, prediction-based word embedding methods predicts a target word by mapping words in the vocabulary. Most usable prediction-based methods are Continuous bag of words (CBOW) and Skip-Gram model. Both of these techniques learn weights by applying backpropagation neural network [23]. Following are short discussion of different word embedding methods and their use cases.

3.1 Count Vector

This method learns vocabularies form all of the documents then form a matrix by counting number of times each word occurs in each document. If total unique word number is T and number of documents is D then count matrix size will be T X D as shown in Figure 3. Using count vector methods any one can prepare embedding vector by choosing high frequency word from vocabulary list.

3.2 TF-IDF Vector

Term Frequency-Inverse Document Frequency (TF-IDF) [24] reflect how important a word for a specific document from a collection of documents which used in information retrieval and text mining. Formal equation of TF-IDF is:

$$tf\_idf_{t,d} = tf_{t,d} \times idf_t \quad (1)$$

Where, $tf_{t,d}$= (Number of times term t appears in a document d) / (Number of terms in the document d) and idf_t = log (Number of documents N) / (Number of documents a term t has appeared in)

If a word has appeared in all the document, then value of idf_t will be 0 and probably the word is not relevant to a particular document.

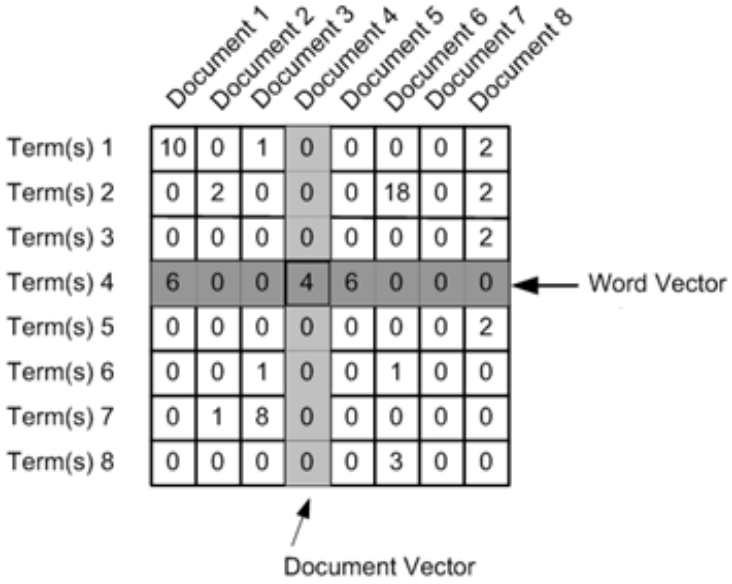


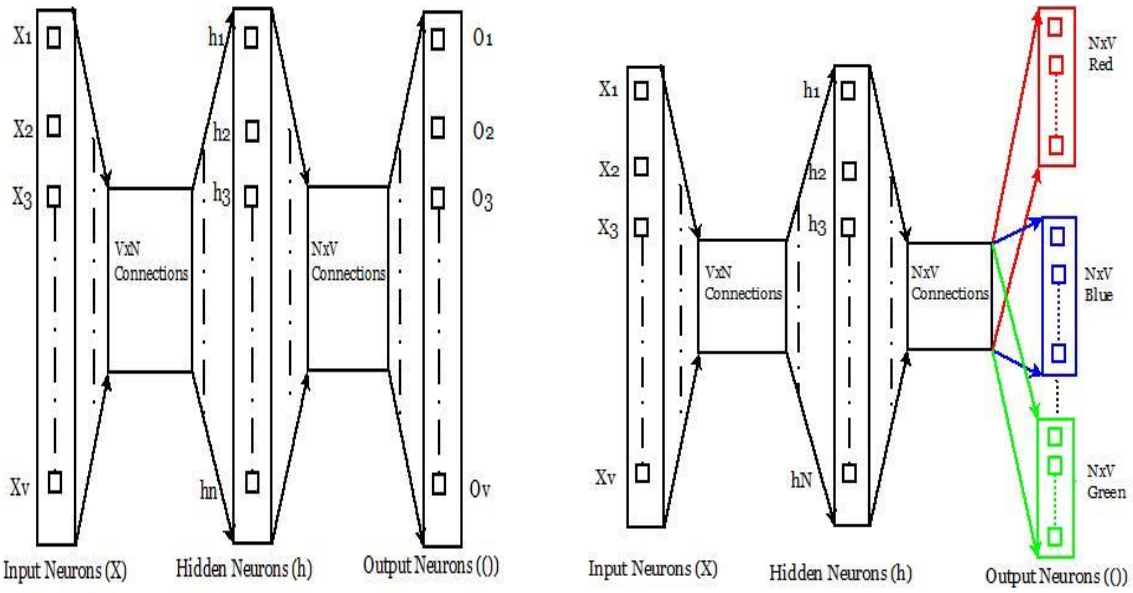
Figure 3-Embedding vector with size T X D using Count Vector

3.3 Co-occurrence Matrix

Used for identify semantic relationship between words by using factorization which is a most common problem for word embedding and can be solved efficiently. This matrix is computed by counting how two or more words occur altogether in a given corpus. $\text{Count}(\text{word}_{(\text{next})} \mid \text{Count}(\text{word}_{(\text{current})})$ is a formula to count neighboring word which represent how many times word ($\text{word}_{(\text{next})}$) follows the current word ($\text{word}_{(\text{current})}$) [25].

3.4 Continuous Bag of Words (CBOW)

Used to predict a word by learning the context which is very effective to find out missing word in a corpus. Context words becomes Neural Network input layer and missing word is predicted at the output layer. Error between the output layer and input layer is used to re-adjust the weights. The architecture of CBOW is shown in Figure 4(a), where we can use several hidden layers between input and output layer in order to maximize the conditional probability of actual output word from input words.



(a) **(b)**

Figure 4-Architecture of (a) CBOW model and (b) Skip Gram Model

3.5 Skip Gram Model

This model is completely reverse of continuous bag of words model used to predict a target context by learning words. Words becomes Neural Network input layer and context of a word predicted at the output layer. Error between the output layer and input layer is used to re-adjust the weights. The architecture of Skip Gram is shown in Figure 4(b).

3.6 FastText

FastText [20] pretrained embedding vectors is a large collection of word vector with four types of dimension size such as 50, 100 and 200 and 300. FastText used Skip Gram and CBOW model for generating the word matrix. In word to vector generation process fastText set minimum frequency count is 2 that means if any important word is appear only once in a whole document it will be automatically discard from this vector generation process.

4. Performance Analysis and Discussion

For experiment purposes, a sports-related Bengali data set collected from the open-source free Bengali dataset corpus [26] is used which contains a total of 12,086 text files and each file contains more than 6500 words. So, there have almost 78,500,000 Bengali words processed to create word embedding vector which work as first input layer in deep learning neural network. Most of the Bengali researcher use count vector to count word frequency for their research but the limitation is that huge memory size and time consuming to process Bengali text processing. we used Python Anaconda 3.6 machine learning platform and Jupyter Notebook tool for this implementation.

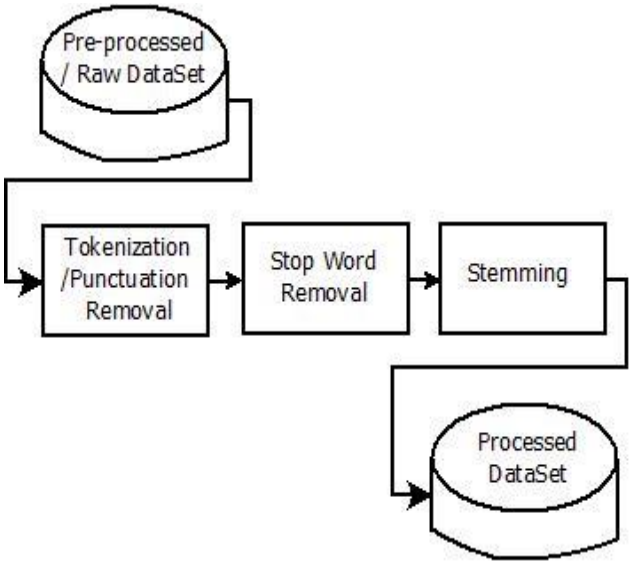


Figure 5-Pre-processing steps

4.1 Pre-processing

Before word embedding method implementation, we need to preprocessing our corpus, because collected data is not suitable for processing. It contains huge amount of punctuation marks, stemming and stop word etc. Preprocessing as shown in Figure 5 is the most important step during implementation. At first, we have to clean data by removing punctuation, stop words and stemming.

4.1.1 Tokenization and Punctuation Removal

Tokenization means breaking up a given sentence into smaller meaningful units. Each unit is called token which may be number, punctuation marks or words. Words have been identified based on the spaces and remove unnecessary items in words such as punctuation marks, hash tag, emoji, emoticon, etc.

4.1.2 Stop Word Removal

Every text contains some unimportant word which is known as stop word. These words have no importance during processing documents. In English language stop words are "the", "a", "an", "of", "my" etc. Similarly, in Bengali language stop words are "এই", "ও", "তাই ", "অথএব", "অথচ" etc shown in Table 1. These stop words have been identified after analyzing huge Bangla data sets for a long time. For example, consider a sentence "You are still talking riddles, the real work has not started yet" where stop words are 'you', 'are', 'still', 'the', 'has', 'not'. After translating the sentence into Bengali format it is "আপনি কিন্তু এখনও হয়োঁলি করে কথা বলছনে আসল কাজটি এখনো শুরু করনে নাই" where stop words are different from English language. By removing stop word the token list will be [হয়োঁলি, কথা, বলছনে, আসল, কাজটি, শুরু, নাই] which are the most powerful words for this sentence.

Table-1 Some Stop words in Bengali Language

Bengali Stop Words
অবশ্য, গুলি, বিষয়টি, অনকে, গয়িে, ব্যবহার, অনকেে, গয়িছে, ব্যাপারে, অনকেেই, গছে, ভাবে, অন্তত, গলে, ভাবইে, অথবা, গলে, মধ্যে, অথচ, গো, মধ্যইে, অর্থাত, চলে, মধ্যওে, অন্য, ছাড়া, মধ্যভাগে, আজ, ছাড়াও, মাধ্যমে, আছে, ছলিনে, মাত্র, আপনার, ছলি, মতো, আপনি, জন্য, মতোই, আবার, জানা, মোটইে, আমরা, ঠকি, যখন, আমাকে, তনিি, যদি, আমাদরে, তনি, ঐ, যদওি, আমার, তনিওি, যাবে, আমি, তখন, যায়, আরও, তবে, যাকে, আর, তবু, যাওয়া, আগে, তাঁদরে, যাওয়ার, আগইে, তাঁহারা, যত, আই, তাঁরা, যতটা, অতএব, তাঁর, যা, আগামী, তাঁকে, যার, অবধি,

তাই, যারা, অনুযায়ী, তমেন, যাঁর, আদ্যভাগে, তাকে, যাঁরা, এই, তাহা, যাদরে, একই, তাহাতে, যান, একে, তাহার, যাচ্ছে, একটি, তাদরে, যতে, এখন, তারপর, যাতে, এখনও, তারা, যনে, এখানে, তারঁে, যমেন, এখানইে, তার, যখোনে, এটি, তাহলে, যনিি, এটা, তনিি, য, এটাই, তা, রখে , এতটাই, তাও, রাখা, এবং, তাতে, রয়ছে, একবার, তে◌া, রকম, এবার, তত, শুধু, এদরে, তুমি, সঙ্গে, এঁদরে, তে◌ামার, সঙ্গওে, এমন, তথা, সমস্ত, এমনকী, থাকে, সব, এল, থাকা, সবার, এর, থাকায়, সহ, এরা, থকে, সুতরাং, এঁরা, থকেওে, সহতি, এস, থাকব, সইে, এত, থাকনে, সটো, এতে, থাকবনে, সটেি, এস, থকেইে, সটোই, একে, দকি, সটোও, এ, দতি, সম্প্রতি, ঐ, দয়িে, সখোন, ই, দয়িছে, সখোনে, ইহা, দয়িছেনে , স, ইত্যাদি, দলিনে, স্পষ্ট, উনি, দু, স্বয়ং, উপর, দুটি, হইত, উপরে, দুটে◌া, হইব, উচতি, দয়ে, হলে, ও, দওেয়া, হইয়া, ওই, দওেয়ার, হচ্ছে, ওর, দখো, হত, ওরা, দখে, হতে, ওঁর, দখেতে, হতইে, ওঁরা, দ্বারা, হব, ওকে, ধর, হবনে, ওদরে, ধরা, হয়ছেলি, ওঁদরে, নয়, হয়ছে, ওখান, নানা, হয়ছেনে, কত, না, হয়, কব, নাকি, হয়নি, করত, নাগাদ, হয়, কয়কে, নতি, হয়ইে, কয়কেটি, নজি, হয়তে◌া, করব, নজিইে, হল, করলনে, নজিরে, হল, করার, নজিদেরে, হলইে, কারও, নয়িে, হলওে, করা, নওেয়া, হলে◌া, করি, নওেয়ার, হসিাব, করয়িে, নইে, হওয়া, করার, হওয়ার, করাই, পক্ষ, হওয়ায়, করল, পর্যন্ত, হন, করলনে, পাওয়া, হে◌াক, করতি, পারনে, জন, করয়িা, পারি, জনক, করছেলিনে, পার, জনরে, করছ, পর, জানত, করছনে, পরইে, জানায়, করছেনে, পরওে, জানয়িে, করছে, পর, জানানে◌া, করনে, পয়ে, জানয়িছে, করবনে, প্রতি, জন্য, করায়, প্রভৃতি, জন্যওজ, কর, প্রায়, জ, করইে, ফরে, বশে, কাছ, ফল, দনে, কাছ, ফরি, তুল, কাজ, ব্যবহার, ছলিনে, কারণ, বলত, চান, কছিু, বললনে, চায়, কছিুই, বলছেনে, চয়ে, কন্তিু, বলল, মে◌াট, কংিবা, বলা, যথষ্টে, কি, বলনে, টি, কী, বল, কউে, বহু, কউেই, বস, কাউক, বার, কনে, বা, ক, বনিা, কে◌ানও, বরং, কে◌ানে◌া, বদল, কে◌ান, বাদ, কখনও, বার, ক্ষত্রে, বশিষে, খুব, বভিন্নি্ন।

4.1.3 Stemming

The process of reducing variation of a word is called stemming. There can be different forms of a word based on the context it is being used. For example, "করা", "করছি", "করছলিাম", "করছলি", "করছে","করছে" etc. for all these words, "কর" is the root word. Python Regulation Expression library was used for reducing variation.

Table 2-Top 5 word with corresponding Weight using TF-IDF methods

Word	Weight
খলো	0.67518
সময়	0.67003
গ্যালারী	0.49186
ফলাফল	0.45575
সমর্থ	0.41561

4.2 TF-IDF word embedding method

TF-IDF word embedding method discussed in section 3 and in this section, only implementation process has been discussed here. Scikit-learn library in Python provide a TfidfVectorizer function to create TF-IDF word embedding vectors. At first import TfidfVectorizer from sklean.feature_extraction.text then call fit and transform to calculate the TF-IDF score for the text. Finally, top 5 words have print as shown in Table 2 with their weight through the given document.

After TF-IDF operation produces following results:

Total unique words are: 5430

Highest word weight is: 0.67518

Lowest word weight is: 0.00451

4.3 Local Word Embedding using Skip Gram

Word2Vec can be generated using either Continuous Bag-Of-Words or Skip-gram model which is discussed in section 3. In this paper, the Skip-gram model with a negative sample have chosen for Local word vector generation and implemented on Python using NumPy then Keras Python framework is used to implement deep learning neural network for NLP processing. After pre-processing a clean dataset has found then set the value of some hyperparameters such as learning rate, epochs, embedding size, window size, etc. and generate training data by building vocabulary also build dictionaries that map word to id or vice versa. Then use Skip-gram model to training vocabulary by forward propagation and backpropagation network. Finally get word vector and their similar words in word embedding. Full process shown in Figure 6.

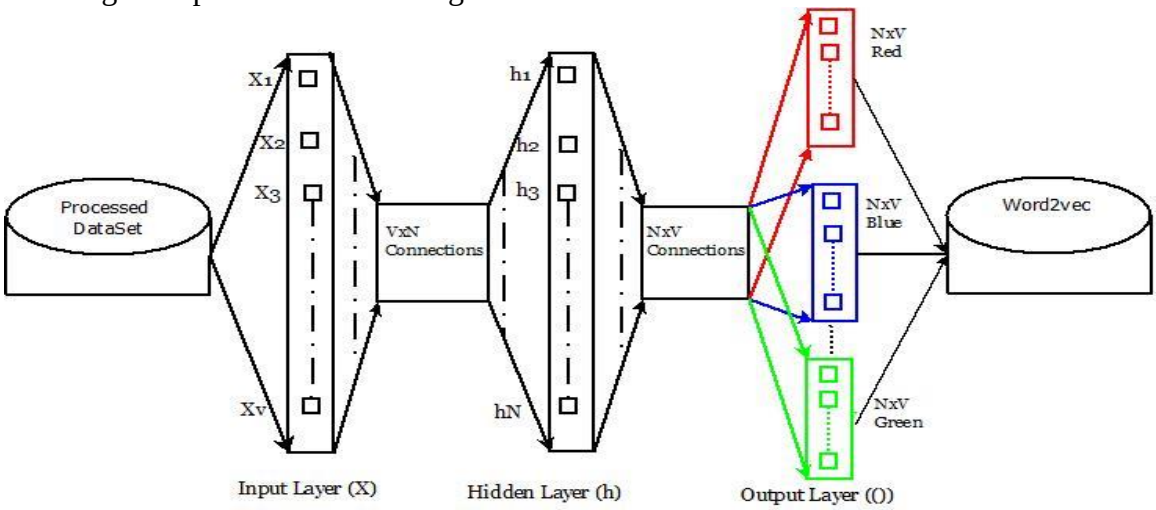


Figure 6-Local Word Embedding for Bengali Language implementation process

4.4 Experimental Evaluation

Table 3 shows the data demography to check the performance of proposed local word embedding compare with pretrained fastText word embedding.

Table 3- Dataset Statistics

Terminology	Values
Total number of Documents	12086
Total Words before pre-processing	78,500,000
Total unique words before preprocessing	12354672
Number of words after preprocessing	25467890
Unique Words after preprocessing Number of	164765

The whole data set has pre-processed and trimmed before sending it to the word vector generation process. The complete datasets have categorized into 5 different features and applied the proposed local word embedding model separately in each proposed feature as shown in Table 4.

Table 4-Measurement Features

Features	Window Size	Negative sample	Vector Dimension size
F1	5	10	300
F2	10	15	300
F3	15	20	300
F4	20	25	300
F5	25	30	300

For training and testing purpose 80% datasets have used for train the model and rest of 20% dataset is used to evaluate the model using Random Forest [27] machine learning classifiers.

The model has been trained in five separate features F1 to F5 as shown in Table 4 and tests the model in each feature. The size of epochs is 100 and verbose size is 2 and train the model using fit function. Google Colab open-source platform has used to execute the experiment entirely in the Jupyter notebook. Python 3.7, TensorFlow 2.2.1, Panda 1.0.3, Nvidia K80s GPUs have used to implement the research. Panda data frame has used for dataset handling and scikit-learn 0.22.2 for training and testing the model. First, the proposed local word embedding model has tested for all features and trained and evaluated the model. Then use the fastText pretrained word embedding model to train and test based on different features. After the implementation of skip-gram based local word embedding following results have been produced as shown in Figure 7 for the semantic relationship of different words.

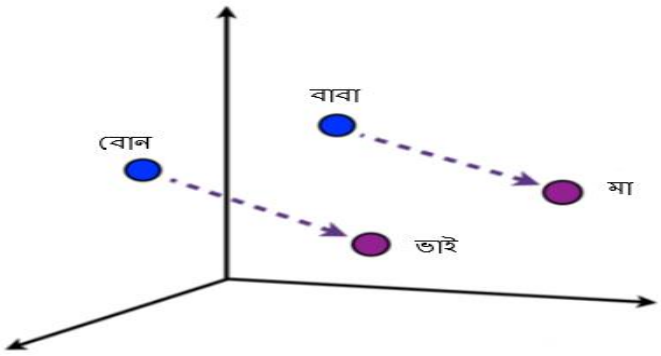


Figure 7-Semantic relationship between Male-Female

If we want to find out relationship based on equation like:

$$\text{বাবা} + \text{ময়ে} - \text{ছলে} = ? \quad (2)$$

Probable result is shown in Table 5:

Table 5-Output of বাবা + ময়ে – ছলে = ?

Probable Output Word	Proposed/ Local Word Embedding	fastText Word Embeddings
মা	0.9142267107 (91.42%)	0.8956247434 (89.56%)
সন্তান	0.7938171625 (79.38%)	0.7865468544 (78.65%)
বাবামা	0.7937164306 (79.37%)	0.8037171625 (80.37%)
স্ত্রী	0.7889907360 (78.89%)	0.7967107976 (79.67%)
স্বামী	0.7878202795 (78.78%)	0.7937536805 (79.37%)

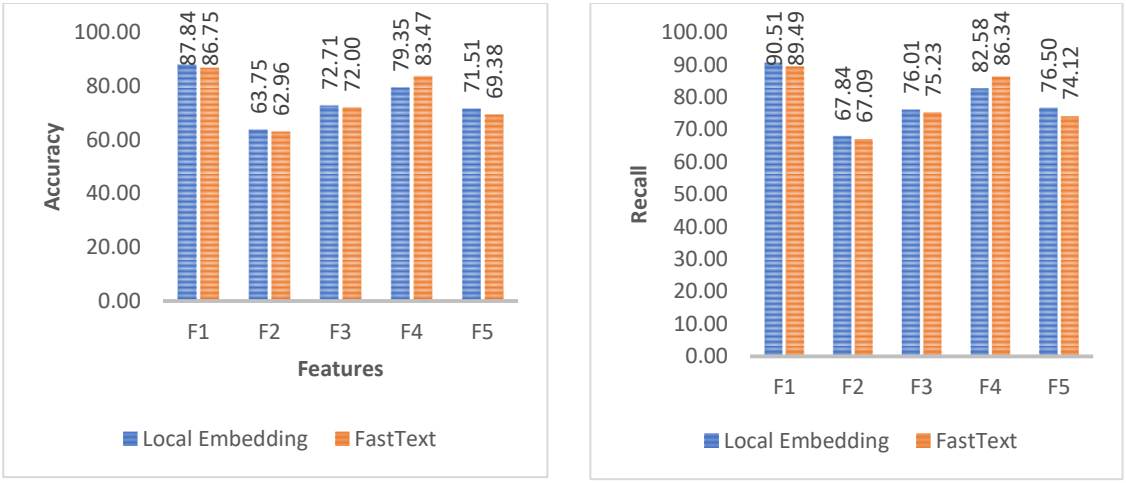
Proposed word embedding and fastText word embedding produce the closest neighbors to this equation. The top 5 closest neighbors are shown in Table 5 with their probability scores. Local word embedding model produce probability that is semantic related to other words which and accuracy is better than pretrained word vector fastText.

Table 6-performance result of Proposed & fastText word embedding

Method	Features	Accuracy (%)	Precision (%)	Recall (%)
Proposed word Embedding	F1	87.84	87.16	90.51
	F2	63.75	63.59	67.84
	F3	72.71	73.09	76.01
	F4	79.35	79.31	82.58
	F5	71.51	71.58	76.50
fastText pre-trained word embedding	F1	86.75	86.17	89.49
	F2	62.96	62.86	67.09
	F3	72.00	72.48	75.23
	F4	83.47	83.13	86.34

	F5	69.38	69.83	74.12

Performance result of proposed & fastText word embedding shown in Table 6. The local word embedding model produces different scores for different features whereas the F1 feature produces better performance than the other four features. The proposed model achieves a maximum accuracy score of 87.84% for feature F1. FastText pretrained word embedding model produces 86.75% accuracy for feature F1 which is minimum than proposed word embedding model. Figure 8 shows the graphical representation of proposed and fastText model.



(a) **(b)**

Figure 8-Graphical representation of local embedding and fastText embedding (a) Accuracy score (b) Recall score

Figure 9 shows the F1 score of the local word embedding model and the fastText pretrained word embedding model. The first and third features for both models produce a more advanced F1 score where the proposed model F1 score is higher than the fastText model.

Table 7-Confusion matrix

Confusion matrix	Local word embedding		fastText pre-trained word embedding	
	Predicted Positive	**Predicted negative**	**Predicted Positive**	**Predicted negative**
Actually positive	48.21%	05.05%	47.66%	05.61%
Actually negative	07.11%	39.63%	07.65%	39.08%

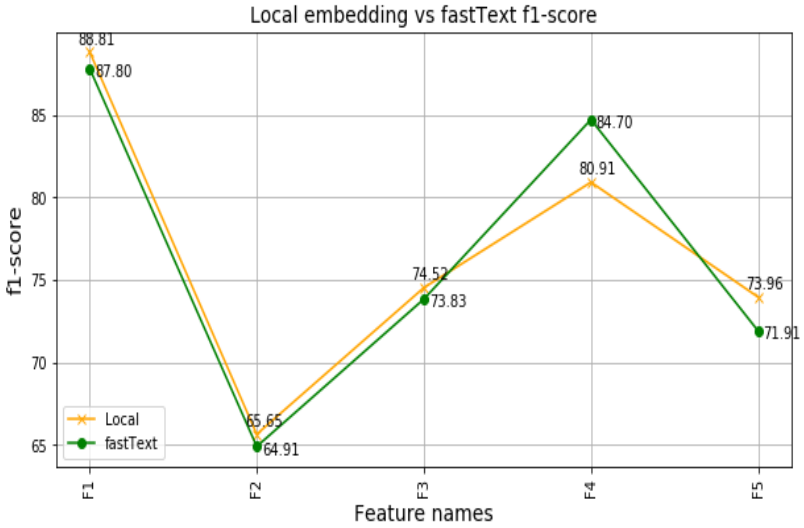
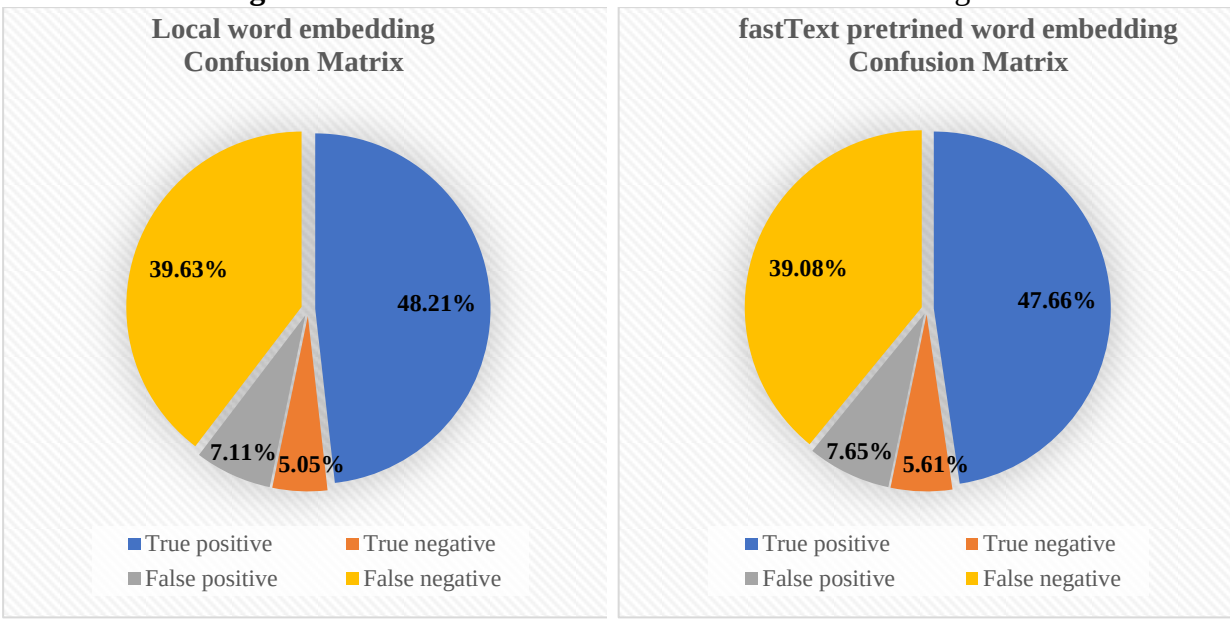


Figure 9-F1-Score of Local and fastText word embedding



(a) **(b)**

Figure 10-Graphical representation of confusion matrix (a) Local word embedding (b) fastText pretrained word embedding

Confusion matrix for local word embedding model and fastText word embedding model shown in Table 7 and graphical representation shown in Figure 10. The positive prediction of the proposed model is 48.21% of the total test dataset that means it can correctly identify the 48.21% dataset as a positive class among the 53% dataset. Only 05.05% dataset was actually positive but the proposed model predicts this as a negative class. For negative classes, the proposed model predicts the negative class of 39.33% of the 47% of total negative datasets. Compared to fastText, our proposed model actually has higher scores of positive and negative probabilities that the fastText model. For false positive and false negative score fastText model probabilities is higher than proposed local word embedding model. The fast text word vector is unique and produces good results in most cases but in Bengali language processing local word vector output can be made based on requirements and shows better performance than other pre-trained word vectors.

5. Conclusion and Future work

This research explores different word embedding methods than can be used to create local word embedding vector for Bengali language processing which is very important for deep learning neural network. We perform the most useful two word embedding methods for generating embedding vector. IF-IDF method used to count frequency of word which is used for classification, clustering of NLP sentiment analysis and Word2Vec method used to predict word which is used for regression or prediction analysis. After generating word embedding vector, we fed this vector to deep learning neural network as input-to-input layer. Then we used several hidden layers also CNN layers to successfully train the machine using this word embedded data set. Finally, we compared proposed local word embedding model with fastText pretrained word embedding model. Experimental result shows that proposed model accuracy is 87.84% whereas fastText model accuracy is 86.75% for Bengali language processing. In the future, we have a plan to identify all stop words during pre-processing and to consider misspelling or vocabulary out of words to calculate their word vectors for Bangla language processing.

References

[1] Ronan Collobert and Jason Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," In Proceedings of ICML. 160–167, 2008.

[2] Tomas Mikolov, Wen-tau Yih, and Georey Zweig, "Linguistic Regularities in Continuous Space Word Representations," In Proceedings of NAACL. 746–751, 2013.

[3] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. , "Glove: Global Vectors for Word Representation," In Proceedings of EMNLP. 1532–1543, 2014.

[4] Md. Anwar Hussen Wadud and Md. Rashadul Hasan Rakib, "Text Coherence Analysis based on Misspelling Oblivious Word Embeddings and Deep Neural Network," *International Journal of Advanced Computer Science and Applications*(IJACSA), vol. 12, no. 1, 2021.

[5] Peter D Turney, Patrick Pantel, and others, "From frequency to meaning: Vector space models of semantics," *Journal of Artificial Intelligence Research*, vol. 37, no. 1(2010), pp. 141–188, 2010.

[6] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, Christian Jauvin, "A Neural Probabilistic Language Model," *Journal of Machine Learning Research*, vol. 3, no. (2003), pp. 1137–115, 2010.

[7] G.E. Hinton, "Learning distributed representations of concepts," In Proceedings of the eighth annual conference of the cognitive science society, pages 1–12, 1986, Amherst, MA.

[8] Tom Kenter and Maarten de Rijke, "Short text similarity with word embeddings," In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM 2015), pages 1411–1420, 2015.

[9] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen, "Convolutional neural network architectures for matching natural language sentences," *In Advances in Neural Information Processing Systems* (NIPS 2014), pp.2042–2050, 2014.

[10] Aliaksei Severyn and Alessandro Moschitti, "Learning to rank short text pairs with convolutional deep neural networks,: In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2015), pages 373–382, 2015.

[11] Wenpeng Yin and Hinrich Sch ¨ utze, "Convolutional neural network for paraphrase identification," In Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL 2015), pages 901–911, 2015.

[12] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu, "Towards universal paraphrastic sentence embeddings," Proceedings of the International Conference on Learning Representations (ICLR 2016).

[13] Karl Moritz Hermann and Phil Blunsom, "Multilingual distributed representations without word alignment," In Proceedings of the International Conference on Learning Representations (ICLR 2014).

[14] Karl Moritz Hermann and Phil Blunsom, "Multilingual models for compositional distributed semantics," In Proceeedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014), pages 58–68.

[15] Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha, "An autoencoder approach to learning bilingual word representations*," In Advances in Neural Information Processing Systems* (NIPS 2014), pp. 1853–1861, 2014.

[16] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler, "Skipthought vectors," *In Advances in Neural Information Processing Systems 28* (NIPS 2015), pp. 3294– 3302. Curran Associates, Inc.

[17] Wentau Yih, Kristina Toutanova, John C. Platt, and Christopher Meek, "Learning discriminative projections for text similarity measures," In Proceedings of the Fifteenth Conference on Computational Natural Language Learning, pages 247–256, 2016.

[18] Felix Hill, Kyunghyun Cho, and Anna Korhonen, "Learning distributed representations of sentences from unlabelled data," In Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL 2016).

[19] Pennington, Jeffrey & Socher, Richard & Manning, Christoper. (2014). Glove: Global Vectors for Word Representation. EMNLP. 14. 1532-1543. 10.3115/v1/D14-1162.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[20] Joulin, Armand & Grave, Edouard & Bojanowski, Piotr & Douze, Matthijs & Jégou, Hervé & Mikolov, Tomas. (2016). FastText.zip: Compressing text classification models.

[21] Kumar, S., Kumar, S., Kanojia, D., & Bhattacharyya, P, "A Passage to India: Pre-trained Word Embeddings for Indian Languages," In Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL) (pp. 352-357), (2020, May).

[22] B. Jiang, Z. Li, H. Chen and A. G. Cohn, "Latent Topic Text Representation Learning on Statistical Manifolds," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 11, pp. 5643-5654, Nov. 2018.

[23] W. Liang, R. Feng, X. Liu, Y. Li and X. Zhang, "GLTM: A Global and Local Word Embedding-Based Topic Model for Short Texts," in *IEEE Access*, vol. 6, pp. 43612-43621, 2018.

[24] H. Wu and R. Luk and K. Wong and K. Kwok, "Interpreting TF-IDF term weights as making relevance decisions," *ACM Transactions on Information Systems*, vol. 26, no. 3, 2008.

[25] Cochez, M., Garofalo, M., Lenßen, J., & Pellegrino, M. A. (2018). A first experiment on including text literals in KGloVe. arXiv preprint arXiv:1807.11761.

[26] www.scdnlab.com/corpus/

[27] Xu, B., Guo, X., Ye, Y., & Cheng, J., "An Improved Random Forest Classifier for Text Categorization," *JCP*, vol. 7, no. 12, pp. 2913-2920, 2012.