International Journal of Engineering & Technology, 10 (2) (2021) 95-102

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

BnVec: Towards the Development of Word Embedding for Bangla Language Processing

Md. Kowsher[1*], Md. Jashim Uddin[2], Anik Tahabilder[3], Nusrat Jahan Prottasha[4], Mahid Ahmed[5], K. M. Rashedul Alam[6] and Tamanna Sultana[7]

[1,2,5,6,7] *Noakhali Science and Technology University, Noakhali 3814, Bangladesh*
[3] *Western Carolina University, Cullowhee, NC 28723, USA*
[4] *Daffodil International University, Dhaka 1207, Bangladesh*
[*] *Corresponding author E-mail: ga.kowsher@gmail.com*

Abstract

Progression in machine learning and statistical inference are facilitating the advancement of domains like computer vision, natural language processing (NLP), automation & robotics, and so on. Among the different persuasive improvements in NLP, word embedding is one of the most used and revolutionary techniques. In this paper, we manifest an open-source library for Bangla word extraction systems named BnVec which expects to furnish the Bangla NLP research community by the utilization of some incredible word embedding techniques. The BnVec is splitted up into two parts, the first one is the Bangla suitable defined class to embed words with access to the six most popular word embedding schemes (CountVectorizer, TF-IDF, Hash Vectorizer, Word2vec, fastText, and GloVe). The other one is based on the pre-trained distributed word embedding system of Word2vec, fastText, and GloVe. The pre-trained models have been built by collecting content from the newspaper, social media, and Bangla wiki articles. The total number of tokens used to build the models exceeds 395,289,960. The paper additionally depicts the performance of these models by various hyper-parameter tuning and then analyzes the results.

Keywords: *Word Embedding, Word2vec, fastText, Hash Vectorizer, TF-IDF, Bangla NLP*

1. Introduction

Word embedding refers to the vector representation of linguistic or phonetic information. It is one of the most popular document representation models that is being comprehensively used in multiple domains of Natural language processing (NLP) application including named entity recognition [7], sentiment analysis, part of speech tagging, and so forth [20]. Document classification has become another essential research area where word embedding techniques are applied for semantic relationships between words [15]. Systems like informative dialogue flow [2,21] and chatbot [8] also incorporate word embedding techniques. There are lots of word embedding strategies accessible for popular dialects. Whereas, in Bengali, we notice an alternate picture. Bengali being one of the most communicated dialects on the earth has some unique linguistic traits which make it distinguishable from most of the languages, even from the sub-continental dialects. Therefore, it is exceptionally vital to embrace an alternate methodology in building up the word representation scheme of Bengali semantic frameworks for gaining information from the text. Changing over data into lower-dimensional vectors is the primary objective of this research. Throughout these years various strategies for word clustering or embedding have been presented. A portion of these techniques offers the grouping of English and other popular languages in terms of high precision. Since Bangla is an increasingly entangled language, high accuracy is hard to pick up. We tried three separate techniques to choose the strategy that works best for Bangla word-embedding. These days word vector portrayal has been the most widely recognized strategy for word group creation.

BnVec is a distributed word embedding open-source repository that provides the Bangla NLP research community with powerful and free word embedding models. BnVec is a distributed and state-of-art word representation library that intends to outfit the staggering word embedding models like the CountVectorizer, TF-IDF, Hash Vectorizer, Word2vec (Google) [14], fastText (Facebook) [3], and GloVe (Stanford) [17]. Our paper can be divided into two major parts. In the first segment, we

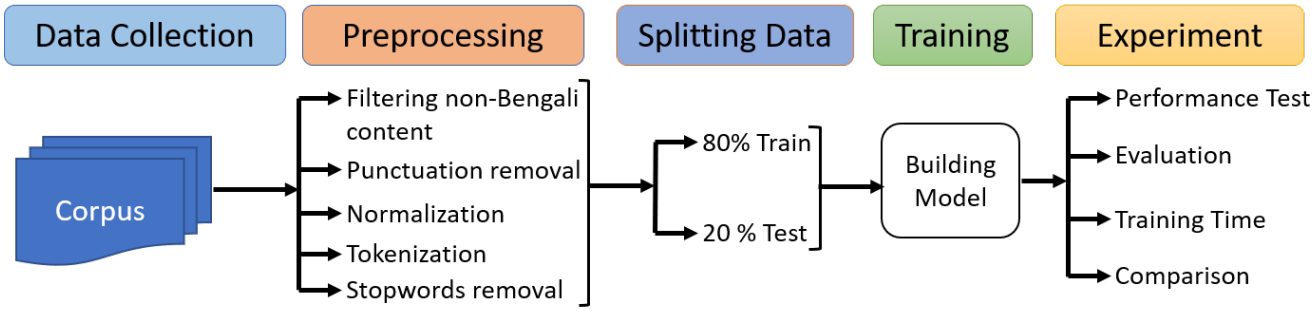

Figure 1: Methodology of BnVec

introduce the data collection, data formatting, data pre-processing, and hyper-parameter tuning. In the second segment we proceed with robust methodological assertions like filtering content, tokenizing, stop words removal experimentations, and so on. The major contributions in this manuscript are stated below:

- We present 5 Word Embedding techniques and figure out the best suitable word embedding system for the Bangla language
- We introduce a powerful python pre-trained module named **BnVec** which will strengthen the research area of Bangla NLP
- Some qualitative and quantitative comparisons between different word embedding techniques with a sufficiently large Bangla dataset are shown

The remaining section of the article is organized as follows: Section-II describes similar work on word vectorizers and embedding; Section-III illustrated the proposed methodology of this work. Section-IV describes the experiment, the pretrained model, and evaluates results. We have concluded the article in Section V by mentioning the future research direction.

2. Related Work

Lund, K., Burgess, C. have presented a matrix factorization method [11] of finding semantic similarities of words. They showed the relationship between words and context words by building a co-occurrence matrix. Their work has been improved in [19] applying some normalization techniques. There have been some works on window-based methods on word embedding. Among them, the Word2vec model [13] is the most used one in NLP areas. Authors in this work have proposed two novel architectures Continuous Bag of Words (CBOW) & Skip-gram (SG). Mikolov et al. extended this work and improved CBOW and SG [14] by increasing the training speed and avoiding overly frequent words. A hybrid model using both matrix factorization and context window-based method named GloVe model has been introduced in [17] which reduced the shortcomings of both methods. But GloVe doesn't take into account the morphological information of words. Bojanowski et al. were the first to consider the morphological information while finding semantic relationships among words [3].For Hindi word embedding there exists fastText and Adaptivemodesls [5]. Word embedding techniques have been implemented in many works to extract information and used as language features. In [6], the authors showed word clustering using the N-gram technique. They found a semantic and contextual sense of words using the N-gram model and concluded that words being used in similar contexts and having similar meanings belongs to the same group. Word sense disambiguation for Hindi words was done in [10] by using SG and CBOW. Ahmad et al. used Word2vec for the Bengali document classification problem [1]. Working with a large language dataset is very challenging for NLP researchers. Zhang et al. showed a way to reduce the dimensionality of large language corpora using Word2vec [12]. Reviewing all the previous works done on word embedding, we can conclude that this is a very rich research area and there are enough opportunities to make contributions in this sector, especially for the Bangla language, since there are not enough resources related to Bangla word embedding.

3. BnVec Methodology

We have complemented five major phases including data collection, preprocessing, training data incorporating word representation algorithms, experiments, and evaluation to complete this BnVec model. We have also ensured the quality of data by making it outlier-free using preprocessing technique. Here, Fig.1 describes the whole methodology of the proposed BnVec method.

3.1. Data description and Corpus preparation

To make a fully functioning word embedding model, a great measure of information is required to validate the speculation and calculation. Word embedding works on linguistic information or corpus, so for that, we had to gather different chunks of the data. Wellspring of our Bangla plain content assortment based on three different web-based methods such as news-related data, social media, and information related to Bangla.
We surfed on various online sources like Bengali blogs, articles, books, news portals, Bangla Wikipedia, and so on. Social media were another big source of our corpus. Another big source of data is the news portals, we collected the last 10 years of news of the popular Bangla newspapers 'The Daily Prothom Alo' and 'The Daily Ittefaq'. Moreover, the Bangla Wikipedia is a significant part of the information collection. In the Table1 we described the dataset we have used.

Table 1: The summary of the collected data

Total Articles	1,432,210
Total Sentences	65,881,660
Total Words	395,289,960
Total Words after Removing Punctuation and Extra Stop Words	354,520,143
Unique Words	6,565,187

The Fig.2 represents that the most frequent words are maximum stop words, even they do not make sense as a good sentiment of the word representation. In the pre-processing part, the stop words are removed.

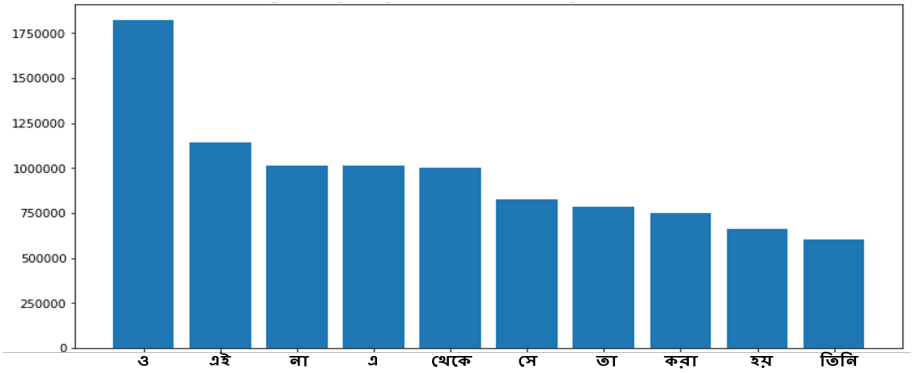

Figure 2: Most Frequent Words before Stopwords Removal

All collected data are stored as a text of Bengali sentences. They can be treated as strings. However, we cannot input this raw data directly to the model. So, we have implemented pre-processing techniques to make the dataset suitable for the model.

3.2. Data Pre-Processing

3.2.1. Non-Bangla Content Filtering

We first filtered the non-Bengali word from the dataset to make it a pure Bangla dataset. That is specifically essential while coping with phonetical essence from the "web" or "Facebook" source of data. Even though Bengali may be easily recognized using its alphabet, there are different languages some of whose alphabet consists of characters that overlap with the Bangla alphabet which includes English and other Indo-European languages. The project turned into finding out one of the best collections of textual contents in Bangladeshi literature and filtering something else. To do this, we used Bangla Regular Expressions(BRE) in Bangla to select languages that use certain alphabets.

3.2.2. Punctuation Removal

This applies to delete unnecessary characters that do not add sentimental or informational details in the expression including colon, semicolon, comma, query mark, exclamation point, etc. For instance: **"মাতৃভূমি প্রতি মানুষের ভালোবাসা, ভালো লাগা, গভীর আবেগ-অনুভূতিকে বলে দেশপ্রেম।" ->"মাতৃভূমি প্রতি মানুষের ভালোবাসা ভালো লাগা গভীর আবেগ অনুভূতিকে বলে দেশপ্রেম"** (Motherland is the love, affection, patriotism of everyone)

3.2.3. Normalization

Bangla characters normalization is an essential preprocessing action that is executed for managing Bengali text. There are heaps of words having various character structures. Hence, changing over the character to a word is extremely significant. For example, the word **' টাকা''** (cash) can be composed as the exceptional character **' ৳'**, comparable **' মিনিট'**(minute) can be composed as **' মিঃ'**. Thus **' কোম্পানি'** (company) as **' কোং'**, **'লিমিটেড'** (limited) as **'লিঃ'** and so on. In this progression, we also additionally normalized the URLs and emoticons. Some of the common ones are normalized by supplanting their content with a solitary non-standard Bangla word.

3.2.4. Tokenization

In order to apply machine learning, we need to split the text documents into small units and this process is called tokenization. Let's consider the following example for tokenization. Example: **দেশকে ভালোবাসার অর্থ দেশের মানুষকে ভালোবাসা** (To love the country means to love the people of the country)=> **'দেশকে', 'ভালোবাসার', 'অর্থ', 'দেশের', 'মানুষকে', 'ভালোবাসা'**.

3.2.5. Stop Words Removal

There are some words that are repeated frequently in a sentence but don't have much significance in the meaning of the sentence. So, we have removed the stop words and the procedure is exemplified here: এবং(and), কোথায় (where),অথবা (or), তে (to), সাথে (with), etc are some common examples. Since the word portrayal has no impact on emotions or relevant investigation. Along these lines, the expulsion of stop words may expand training and preparing information execution. Each stopword in the pre-prepared model is expelled. In any case, in the trained model class, the end of stop words relies upon the client. In most cases, one needs to adopt well-structured functions to discard the stop words building.

3.3. Building the models

We employed the preprocessed data for the training mode. Through using various vector sizes, window sizes, and a number of iterations, we evaluated the performance of each model individually. The models we built were built using the Gensim tool [18], a popular toolkit designed to handle almost every common NLP problem and also includes Word2vec model implementation. As building models, we followed the Word2vec, fastText, and GloVe to developed the pre-trained models. For defined functions, we followed CountVectorizer, TF-IDF, hash vectorizer, Word2vec, fastText, and glove.

3.3.1. CountVectorizer

Count Vector is an exceptionally powerful and straightforward technique used in language processing. An N×M grid is generated, where N represents the number of information components and M is the number of unit components present in the information components. Each data component is represented by the frequency of unit elements presented in the data component.

3.3.2. TF-IDF Vectorizer

TF-IDF, aka "Term-frequency time's inverse document-frequency" is a popular and powerful encoding model for text encoding. It calculates the significance of a word in a given document concerning the overall occurrence of that word in a dataset. TF-IDF rises when a word/term is more frequent in a document but less frequent in a whole dataset of documents. The equation of computing the TF-IDF of any term t in a given document d in any document set is calculated from equation-1 [16]:

$$tf\_idf(t,d) = tf(t,d) * idf(t) \tag{1}$$

where "tf" is the term frequency that is the total number of term t in document d calculated from equation-2:

$$tf(t,d) = t : t \in d \tag{2}$$

and "idf" is inverse document frequency calculated as equation-3:

$$idf(t) = log\left(\frac{n}{df(t)}\right) + 1 \tag{3}$$

where n represents the number of total documents in the dataset and df(t) represents the occurrence frequency of t, which is the number of documents in the dataset containing the term t.

3.3.3. Hashing Vectorizer

Hashing vectorizer is another widely used word embedding method formulated in [16]. It uses some hashing techniques to tokens large test datasets in a memory-efficient way. The main strategy used by it is that this vectorizer does not store the vocabulary dictionary in memory. But this strategy comes with the cost of losing the inverse transformation from feature vectors to words.

3.3.4. Word2Vec

Word2Vec is a novel and widely used word embedding model. It uses neural networks to find the semantic similarity of the context of the words. Two inversely related architectures are used in Word2vec namely Skip-gram and a CBOW. Skip gram is an unsupervised learning model which is used to compute semantic similarity in words based on the context of a given word. Skip gram calculates the maximum average logarithmic probability according to equation-4 [14]

$$-\frac{1}{V}\sum_{v=1}^{V}\sum_{-c\le m\le c,\ m\neq 0} log\left[p\left(w_{v+m}|\ w_v\right)\right] \tag{4}$$

from given training words $w_1, w_2, w_3..w_N$. Here c is the size of the context also called the window size. The probability $p(w_{n+m}|w_n)$ can be calculated using equation-5:

$$p(i|\ o)\ =\ \frac{\exp\left(u^T{}_i.u'_o\right)}{\sum_{v\in V} exp\left(u^T{}_z.u'_o\right)} \tag{5}$$

where V represents the vocabulary list and u, u' represents the 'input' and the 'output' vector representations of i, o respectively. Continuous Bag of Words (CBOW) is another architecture highly used in language processing tasks. Unlike Skip-gram, CBOW predicts a target word based on the context of some given words as input. CBOW uses continuous distributed representations of the context. In CBOW a fixed window is constructed with some sequence of words and the model tries to predict the middle word of the window based on the future and history words using the log-linear classifier. The model tries to maximize the expression-6 [4] to achieve the predicted word wt.

$$\frac{1}{V}\sum_{v\in V} log\left(p\left(w_v|\ w_{v-c},\ \ldots w_{v-2}, w_{v-1},\ w_{v+1},\ w_{v+2}\ldots w_{v+c}\right)\right) \tag{6}$$

Where V and c denote the same as the Skip-gram model. Fig.3 [13] represents both the model in a nutshell.

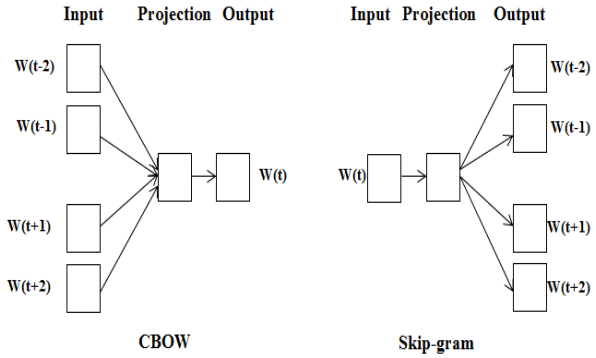

Figure 3: Skip-gram and CBOW architecture

3.3.5. GloVe

Glove or Global Vectors learns the embeddings from word co-occurrences [17]. It formulates a co-occurrence matrix represented by C where the rows and columns represent the word vocabulary and each entry in C, that is C_{ij}, denotes the co-occurrence weight of words i and j. Higher weight results in higher similarity in the resultant vectors.

3.3.6. fastText

fastText is one of the most robust models that is being used for word embedding by using subword information [3]. This model learns the embeddings from character n-grams of the training words. As a result, a non-existing word in the vocabulary during training time can be constructed from its constituent n-grams. This surpasses the limitation of Word2vec and GloVe where a non-vocal word can't be obtained after training.

4. Experiments & Evaluation

In this section, we will illustrate the environmental setup of the project, training performance, installation of the library, analysis of the training time, comparison and evaluation in both qualitative and quantitative approach.

4.1. Experimental setup & training performance

The entire model was executed in the python programming language. The framework had been divided into two major segments. The first one is the per-trained model and the second is the defined train class. Both of the sections were implemented in python 3.8. As the IDE, we used the Google Collab to implement the whole works. We preferred to use TPU (Tensor processing unit) which is an AI accelerator application with 35gb ram and 40 cores which helped to run multi-threads during training. From the Table2, we can figure out that the training time of CBOW & Skip-gram for Word2vec, fasTest, and GloVe are 146.58, 194.43, 109.31, and 276.37, 331.14, 247.34 in minute unit respectively with the same size of the window(5), vector(300), and iteration(25).

Table 2: Parameter and training performance of models

Model	Window Size	Vector Size	Iteration	Training Time(minute)	
				CBOW	SkipGram
Word2Vec	5	300	25	146.58	276.37
fastText	5	300	25	194.43	331.14
GloVe	5	300	25	109.31	247.34

4.2. Installation and Uses

The whole python library for the BenVec is separated into two parts. The first part consists of defined selective vectorizer models with essential defined functions. The first part can be used to train the own dataset and can create a vectorizer according to the dataset. Besides, all the functions were implemented based on Bangla language suitable. The second part is related to three pre-trained functions for three models such as Word2vect, fastText, and GloVe. The Fig.4 represents a

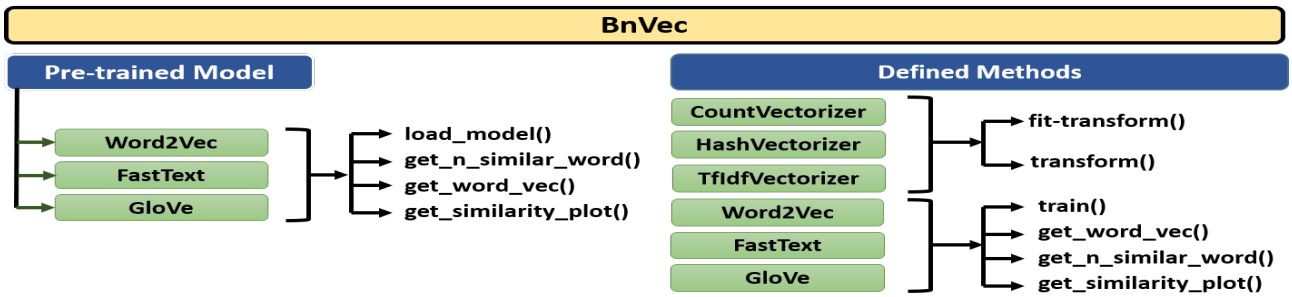

Figure 4: Structure of BnVec

brief structure of the library. Here we used a wide range of sentences to train the models and transferred them to trained models. The main benefit of these models to use easily without training datasets, it also helps to save time, and suitable for the low-level configured machine. Fig.5 & 6 Represent some of BenVec's functions for both classes. To install the BnVec, need to follow the command "pip install BnVec"

```
from BnVec import HashVectorizer
corpus = ['আমাদের দেশ বাংলাদেশ', 'আমার বাংলা']
Vectorizer = HashVectorizer()
X_train = Vectorizer.fit_transform(corpus, n_features = 8)
corpus_t = ["আমাদের দেশ অনেক সুন্দর"]
X_test = Vectorizer.transform(corpus_t)
```

Figure 5: The Hashvectorizer in BnVec

```
from BnVec import BN_Word2Vec
#Training Against Sentences
w2v = BN_Word2Vec(sentences=[['আমার', 'প্রিয়', 'জন্মভূমি'], ['বাংলা', 'আমার',
'মাতৃভাষা'],['আমার', 'প্রিয়', 'জন্মভূমি'], ['বাংলা', 'আমার', 'মাতৃভাষা'],['আমার', 'প্রিয়',
'জন্মভূমি'], ['বাংলা', 'আমার', 'মাতৃভাষা']])
w2v.train()

#Training Against one Text Corpus
w2v_one = BN_Word2Vec(corpus_file="path_to_corpus.txt")
w2v_one.train()

#Training Against Multiple corpuses
w2v_mul = BN_Word2Vec(corpus_path="path/corpus")
w2v_mul.train(epochs=25)

#Training Against a Dataframe Column
w2v_frame = BN_Word2Vec(df= news_data['text_content'])
w2v_frame.train(epochs=25)

#Get word vectors
w2v.get_wordVector('আমার')

#Get similarity
w2v.get_similarity('ঢাকা', 'রাজধানী')

#Get 10 similarity
w2v.get_n_similarWord(['পদ্মা'], n=10)
```

Figure 6: Some of the Functions of Word2vec in BnVec

4.3. Evaluation

To evaluate the proposed models, we have used qualitative as well as quantitative strategies, every one of which is introduced in the accompanying subsection.

4.3.1. Qualitative Evaluation

The motivation behind carrying through the qualitative assessment of our models was to look at how well they identify similitude's among the words. We have used word vectors for a small subset of estimation words and implemented a clustering algorithm to see whether expressions of a similar polarity group together or not. We have conducted similar experiments with different randomly chosen named elements of known kinds.

Table 3: Top 10 Similar Words of CBOW and Skip-gram Architecture

Input Word	CBOW Output of 10 similar words	Skip-gram Output of 10 similar words
পরিবার	বাবামা, পরিবারপরিজন, আত্মীয়পরিজন, স্বজন, পারিবারিকভাবে, স্বামীসন্তান , স্ত্রীসন্তান, সামাজিক, নিকটাত্মীয়রা আত্মীয়স্বজনেরা	পরিবারই, বাবামাসহ, আত্মীয়পরিজন আত্মীয়, স্ত্রীসন্তানদের, স্বামীসন্তান , স্বজন, নিকটাত্মীয়রা, স্ত্রীসন্তান, সপরিবার
বিনোদন	চিত্তবিনোদন, বিনোদনমূলক, খেলাধুলা আনন্দময়, নাচগান, আনন্দফুর্তি, লোকসংস্কৃতি, হাসিআনন্দ, পর্যটনকেন্দ্রিক উপভোগে	খেলাধুলা, আনন্দময়, নাচগান, সংস্কৃতি বিনোদনমূলক, নান্দনিক, পর্যটনকেন্দ্রিক উপভোগে,চিত্তবিনোদন, লোকসংস্কৃতি
দুর্ঘটনা	অগ্নিদুর্ঘটনা, প্রাণহানি, বিস্ফোরণ , অঙ্গহানি, মৃত্যু,মর্মান্তিক, ট্রাকচাপায়, পাহাড়ধসের, ভূমিকম্প, জাহাজডুবি	অগ্নিদুর্ঘটনা,প্রাণহানি,লঞ্চডুবি স্বাস্থ্যঝুঁকি,অসতর্কতার,ভূমিকম্প, বিস্ফোরণ, অঙ্গহানি, মৃত্যু,গাড়িচাপায়

Every one of these undertakings is clarified in further subtleties in the accompanying subsections. The subset utilized can be found in Table3. To picture and analyze the outcomes in a 2D chart. The outcomes appear in Fig.7. Inspecting these outcomes, it is anything but difficult to see that much of the time, expressions of a similar polarity have been bunched together.

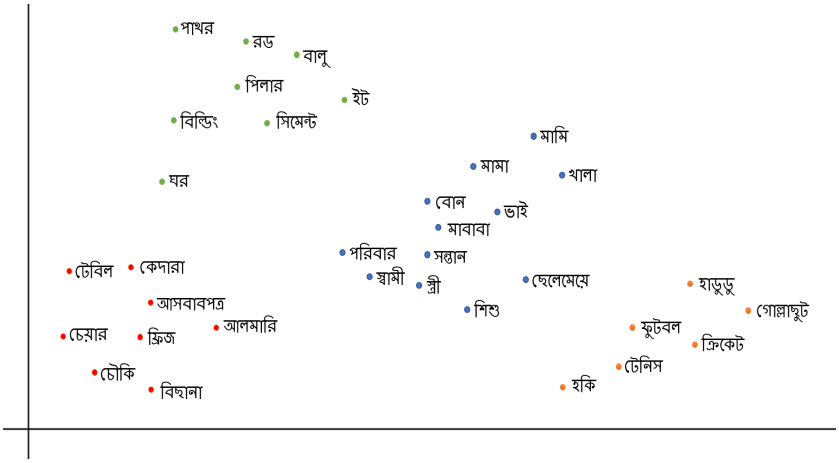

Figure 7: Word Clustering Using fastText

4.3.2. Quantitative Evaluation

To evaluate this model in a quantitative measure, we utilized the Bangla film review corpus (12.5k data) as a form of text classification. Our point was not to completely address this assignment, but instead to exhibit that just by utilizing a decent word embedding model, we could get sensible baseline scores for such undertaking. To do as such, we've splinted the dataset into two sects (training and testing) at that point utilized TF–IDF values as a word embedding method with our Skip-gram based three pre-trained models. At that point, we prepared the training set with Logistic Regression and Impact Learning [9], Random Forest Tree, SVM then the test dataset is utilized to assess the presentation as appeared in Table4. The outcome shows this very guileless methodology gives results that are practically identical to the best of the other accuracy scores; this is accomplished with any element designing or the use of any intricate machine learning models. Word2vec with SVM showed the highest accuracy and F1 score among the other word embedding techniques and classification algorithms. We got an accuracy of 85.921% and an F1 score of 84.091%.

Table 4: Evaluation Metrics on Different ML Algorithms

Algorithm	TF-IDF		Word2Vec		fastText		Glove	
	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score
Logistic Regression	74.521	71.513	82.153	79.151	82.991	80.012	80.151	80.001
Impact Learning	74.981	72.003	84.231	83.124	84.097	83.409	81.892	80.989
Random Forest Tree	73.124	69.983	83.812	83.001	82.974	83.193	80.008	78.921
SVM	75.913	74.081	**85.921**	**84.091**	85.173	83.917	83.821	82.759

5. Conclusion and Future Work

In this research, we have presented two methodologies for Bengali word embedding. The first one includes the well-known functionality of six popular word embedding techniques (CountVectorizer, TF-IDF, Hash vectorizer, Word2vec, fastText, and GloVe) incorporating numerous Bangla text resources like newspaper, Wikipedia, and social media. Different qualitative and quantitative measures were executed on a few tasks to demonstrate their capacity to infer proximity among words and furthermore to look at the exhibition among all word embedding methodologies. We firmly believe that these pre-trained models can easily be utilized by different analysts in the NLP domain to improve the effectiveness of different Bangla NLP tasks. As a future direction of improvement, we want to develop unique and powerful character level embeddings model to solve a considerable amount of recently trending issues like Bangla sentiment analysis, named substance recognition and so on.

References

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

[2] Azevedo, P., Leite, B., Cardoso, H.L., Silva, D.C., Reis, L.P.: Exploring nlp and information extraction to jointly address question generation and answering. In: IFIP International Conference on Artificial Intelligence Applications and Innovations. pp. 396–407. Springer (2020)

[3] Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics **5**, 135–146 (2017)

[4] El Mahdaouy, A., Gaussier, E., El Alaoui, S.O.: Arabic text classification based on word and document embeddings. In: International Conference on Advanced Intelligent Systems and Informatics. pp. 32–41. Springer (2016)

[5] Gaikwad, V., Haribhakta, Y.: Adaptive glove and fasttext model for hindi word embeddings. In: Proceedings of the 7th ACM IKDD CoDS and 25th COMAD, pp. 175–179 (2020)

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

[7] Karim, R., Islam, M., Simanto, S.R., Chowdhury, S.A., Roy, K., Al Neon, A., Hasan, M., Firoze, A., Rahman, R.M., et al.: A step towards information extraction: Named entity recognition in bangla using deep learning. Journal of Intelligent & Fuzzy Systems (Preprint), 1–13 (2019)

[8] Kowsher, M., Tahabilder, A., Islam Sanjid, M.Z., Prottasha, N.J., Hossain Sarker, M.M.: Knowledge-base optimization to reduce the response time of bangla chatbot. In: 2020 Joint 9th International Conference on Informatics, Electronics Vision (ICIEV) and 2020 4th International Conference on Imaging, Vision Pattern Recognition (icIVPR). pp. 1–6 (2020). https://doi.org/10.1109/ICIEVicIVPR48672.2020.9306602

[9] Kowsher, M., Tahabilder, A., Murad, S.A.: Impact-learning: a robust machine learning algorithm. In: Proceedings of the 8th International Conference on Computer and Communications Management. pp. 9–13 (2020)

[10] Kumari, A., Lobiyal, D.: Word2vec's distributed word representation for hindi word sense disambiguation. In: International Conference on Distributed Computing and Internet Technology. pp. 325–335. Springer (2020)

[11] Lund, K., Burgess, C.: Producing high-dimensional semantic spaces from lexical co-occurrence. Behavior research methods, instruments, & computers **28**(2), 203–208 (1996)

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

[13] Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)

[14] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)

[15] Mojumder, P., Hasan, M., Hossain, M.F., Hasan, K.A.: A study of fasttext word embedding effects in document classification in bangla language. In: International Conference on Cyber Security and Computer Science. pp. 441–453. Springer (2020)

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

[17] Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)

[18] Řehůřek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. pp. 45–50. ELRA, Valletta, Malta (May 2010)

[19] Rohde, D.L., Gonnerman, L.M., Plaut, D.C.: An improved model of semantic similarity based on lexical co-occurrence. Communications of the ACM **8**(627-633), 116 (2006)

[20] Sun, X., Gao, Y., Sutcliffe, R., Guo, S.X., Wang, X., Feng, J.: Word representation learning based on bidirectional grus with drop loss for sentiment classification. IEEE Transactions on Systems, Man, and Cybernetics: Systems (2019)

[21] Zaib, M., Sheng, Q.Z., Emma Zhang, W.: A short survey of pre-trained language models for conversational ai-a new age in nlp. In: Proceedings of the Australasian Computer Science Week Multiconference. pp. 1–4 (2020)