



# A Study of fastText Word Embedding Effects in Document Classification in Bangla Language

Pritom Mojumder<sup>1</sup>(✉), Mahmudul Hasan<sup>2</sup>, Md. Faruque Hossain<sup>1</sup>,  
and K. M. Azharul Hasan<sup>2</sup>

<sup>1</sup> Department of Electronics and Communication Engineering,  
Khulna University of Engineering & Technology, Khulna 9203, Bangladesh  
pritom@ieee.org, fhossain@ece.kuet.ac.bd

<sup>2</sup> Department of Computer Science and Engineering,  
Khulna University of Engineering & Technology, Khulna 9203, Bangladesh  
{mahmudul,az}@cse.kuet.ac.bd

**Abstract.** Natural language processing is the current topic due to many important tasks like document classification, named entity recognition, opinion mining, sentiment analysis, textual entailment, etc. Such types of task in the Bangla language is also important. This research work endeavored to find out the word embedding of the Bengali language. Leveraging the fastText word embedding, it has shown significant performance in Bangla document classification without any preprocessing like lemmatization, stemming, and others. For the extrinsic evaluation of our word vectors, a classification problem-solving strategy has been used which showed an outstanding result. In the classification module, attempts have been made to classify 40 thousand News samples into 12 categories. For this purpose, three deep learning techniques have been used: Convolutional Neural Network (CNN), Bi-Directional LSTM (BLSTM) and Convolutional Bi-Directional LSTM (CBLSTM) alongside fastText. From the analogous study of all the parameters of every classifier implemented here, we found that the BLSTM technique is the most promising technique for this task. This technique achieved 91.49%, 87.87%, and 85.5% accuracies for Training, Testing, and Validation set, respectively.

**Keywords:** Natural language processing · Word embedding · Deep learning

## 1 Introduction

In natural language processing (NLP), document classification is a problem of assigning a document to one or more classes or categories. The approach for automatic document classification has a long history. These approaches involve the one-hot vector, co-occurrence matrix-based method [7], term frequency-inverse

document frequency (tf-idf) method [20], singular value decomposition (SVD) method [13], and more others.

The deep learning-based approach is the recent trend in the NLP task [22]. Training a deep learning model for NLP tasks requires pre-processed features that are represented in vector space. Previously this representation used to be done using term frequency-inverse document frequency (tf-idf) technique [20]. The constraint of this pretty old technique is that it does not subsist the semantic context of the sentence. On the other hand, the contemporary word embedding technique developed by Mikolov et al. [17] tries to retain this semantic context of the sentence. Figure 1 illustrates the distinction between these two techniques. In Fig. 1 (A), though word “Hospital” has a semantic relation with the word “Patient,” but tf-idf can not manifest this in vector space. In the case of word embedding, this relation is shown in Fig. 1 (B) using the edges.

Another word embedding technique, fastText, which is considered in this work, has a stable version, robust performance, can be easily implemented, and one of the de-facto standard methods for the present time. The main contribution of this paper is to demonstrate the implication of fastText word embedding[3] in Bangla document classification. The dataset that has been used in our research contains 12 classes, whereas most of the works in Bangla document classification have considered a few number of classes. We have compared the performance with three different neural network models. We have also presented a comparative study with previously done Bangla document classification tasks by various researchers.

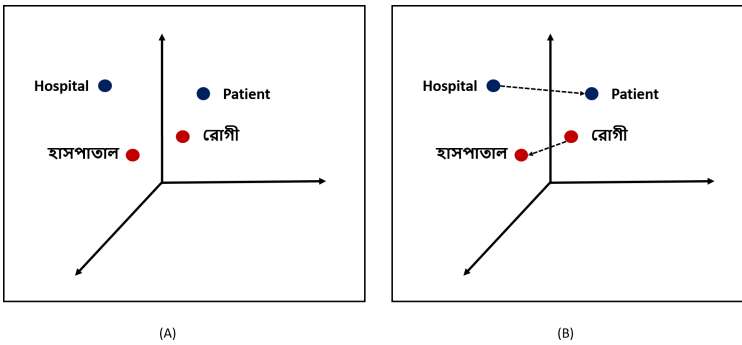


Fig. 1. Words in vector space (A) for tf-idf (B) for word embedding

## 2 Related Works

There are many word embedding techniques like word2vec, GloVe, fastText, etc. Popular word embedding word2vec by Mikolov et al. [16] has been used in many NLP tasks such as document classification [14] and sentiment analysis [23].

The authors in [16] have introduced two model architectures: Continuous Bag-of-Words (CBOW) and Continuous Skip-grams. These two models enable us to learn a distributed representation of words from a large dataset. According to [16], CBOW architecture works better on the syntactic task, and the Skip-grams architecture works better on the semantic tasks. Sumit et al. [21] showed that Skip-grams based method outperformed CBOW in Bangla sentiment analysis tasks due to better word representation.

Convolutional neural networks (CNN) based approach has been taken in sentence classification by Kim [11]. The author has archived superior performance using word2vec. Long Short Term Memory (LSTM), which is a variant of Recurrent Neural Network (RNN), is being used in many NLP tasks [22]. It has been proved that LSTM performs well on sequence data [9]. The hybrid architecture of bidirectional LSTM and CNN discussed by Chiu and Nichols for name entity recognition [5]. However, these methods have analyzed for the processing of English and other languages, whereas, there are a limited number of works in Bangla language.

Several works have been found in the literature on the Bangla document classification task. Decision Tree (DT), K-Nearest Neighbour (KNN), Naïve Bayes (NB), and Support Vector Machine (SVM) based classification methods have been mentioned by Mandal and Sen (2014) [15]. They have used tf-idf for feature extraction on a small dataset and only for five classes. In this work, the best performance has demonstrated for SVM with 89.14% accuracy. In another work, SVM based approach achieved 91% F1-score, researched by Ahmed and Amin [1]. In this work considered word embedding technique word2vec developed by Mikolov et al. [16]. The work suggests that word embedding can significantly improve the accuracy of the classification. In such a classification task, the largest dataset of Bangla articles was used by Alam et al. [2], where the articles were labeled into five categories. The work utilized word2vec and tf-idf with several machine learning approaches and showed that network based approach performs better (F1-score 95%). However, in our work, articles are labeled into twelve categories and the fastText word embedding technique has been used.

### 3 Methodology

#### 3.1 fastText Word Embedding

Word embedding is a distributional representation of words where each word should be mapped to a shared low dimensional space, and each word will be associated with a d-dimensional vector [8]. Unlike many word embeddings, fastText does not ignore the morphology of words. This method is based on continuous skip-grams. Here each word is represented as character n-gram. So for  $n = 3$ , the word *quick* will be represented as:

$$\langle qu, qui, uic, ick, ck \rangle$$

This approach preserves subword information and can compute valid word embedding for out-of-vocabulary words [3]. So, it can give the vector for unseen words during training the word embeddings.

To learn word representations, fastText follows the continuous skip-grams introduced by Mikolov et al. [16], which is a simple model and works well with a small amount of the training data. However, this model ignores the internal structure of words. The authors of fastText proposed different scoring functions to preserve the subword information.

Given the word  $w$ , the set of n-grams appearing in  $w$  will be  $N_w \subset \{1, \dots, N\}$ , Where  $N$  is the dictionary size of n-grams. Vector representation  $Z_g$  is assigned for each n-gram  $n$ . Thus the driven scoring function becomes:

$$s(w, c) = \sum_{n \in N_w} Z_g^T V_c$$

here,  $c$  = context word,  $V_c$  = context vector

### 3.2 Bi-directional Long Short Term Memory (LSTM)

When a deep learning models tends to be deeper then vanishing gradient problem arises. LSTM models can resolve this problem by saving past context and use it any time. For this reason LSTM are widely used for sequential data structure. In the sequential data structures, data can be three dimensional where first two dimensions capture the context of the sentence using samples and words, and the last dimension captures the notion of time. In unidirectional LSTM the network captures chronological order of the data where older past data have impacts on hypothesis. But in the case of Bi-Directional LSTM the network captures both chronological and reverse chronological pattern where both older past and recent

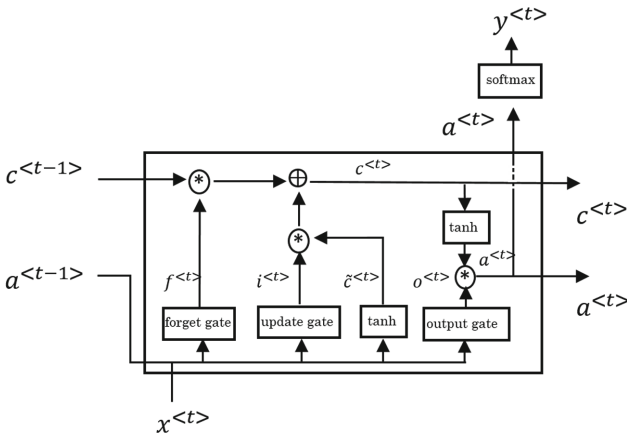


Fig. 2. A simple LSTM cell (Source: [18])

past impacts the hypothesis. For this reason Bi-Directional LSTM works very efficiently on Natural Language. Figure 2 depicts a single LSTM cell [6]. The equation for full LSTM cell is as following [18]:

$$\begin{aligned}
 C^{\tilde{t}} &= \tanh(W_c[A^{<t-1>}, X^t] + b_c) \\
 \Gamma u &= \sigma(W_u[A^{<t-1>}, X^t] + b_u) \\
 \Gamma f &= \sigma(W_f[A^{<t-1>}, X^t] + b_f) \\
 \Gamma o &= \sigma(W_o[A^{<t-1>}, X^t] + b_o) \\
 C^t &= \Gamma u * C^{\tilde{t}} + \Gamma f * C^{<t-1>} \\
 A^{<t>} &= \Gamma o * C^{<t>}
 \end{aligned}$$

Where,

$C^{\tilde{t}}$  = Candidate Value for Memory Cell, C.

$C^{<\tilde{t}-1>}$  = Candidate Value for Previous Memory Cell, C.

$\Gamma u$  = Update Gate

$\Gamma f$  = Forget Gate

$\Gamma o$  = Output Gate

$W_c, W_u, W_f, W_o$  = Weights for the Candidate Cell, Update Gate, Forget Gate and Output Gate respectively

$b_c, b_u, b_f, b_o$  = Bias for the Candidate Cell, Update Gate, Forget Gate and Output Gate respectively

$A^{<t>}$  = Activation Function

$A^{<t>}$  = Previous Cell Activation Function

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$\tanh(x) = 2g(2x) - 1$$

### 3.3 Convolutional Neural Network (CNN)

Usually, a convolutional neural network (CNN) consists of two primary layers named convolution and pooling layer. CNN works best for Image type data, which are mostly three-dimensional. In our case, the word embedding vector is three dimensional, which we can easily fit in the CNN. CNN can seize local spatial patterns in each of its convolutional layers. With the help of pooling layers, it lowers the spatiality of each dimension. CNN uses filters in convolution layers to adapt the weights of the network according to the loss using sliding. From the response of each filter, feature maps are created in each layer. In most cases, the filter size is taken to be  $3 \times 3$  or  $5 \times 5$ .

### 3.4 Evaluation Metrics

For the performance analysis of the word to vector model skip-grams, we used the classification task as an extrinsic evaluation technique. For this reason, we need standard metric values by which we can make a comparative analysis. So our work is indirectly focused on classification, and we have used the following evaluation metrics for the evaluation of the classification:

**Confusion Matrix.** The dimension of the confusion matrix is  $N \times N$ , where  $N$  denotes the number of classes or labels. In this matrix, the rows represent the number of samples which model has predicted, and the columns represent the number of actual sample values. Hence, the number of samples in the diagonal of the confusion matrix means the better performance of the deep learning model. From the confusion matrix, we can calculate the other metrics. The confusion matrix helps us to visualize the complete performance of the model. It represents a summary of prediction results on a classification task.

$TP(S_i)$  = All the predicted cases are  $S_i$  and those are really  $S_i$ .

$TN(S_i)$  = All the predicted cases are non  $S_i$  and those are really non  $S_i$ .

$FP(S_i)$  = All the predicted cases are  $S_i$ , but those are not  $S_i$ .

$FN(S_i)$  = All the predicted cases are non  $S_i$ , but those are  $S_i$ .

It is useful for measuring accuracy, sensitivity, Recall and AUC-ROC curve.

**Accuracy.** It is the measure of correct prediction against total samples.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

**Precision.** It means the rate of correctly predicted positives against the total number of predicted positives.

**Recall.** It represents the actual positive rate claimed by the model, which means the number of correctly claimed positives compared to the actual number of positives in the dataset.

**F1 Score.** F1 Score is the weighted average of Precision and Recall.

### 3.5 Dataset Preparation

The open-source dataset has been used here is collected from Kaggle<sup>1</sup>. It contains 3999 Bangla news articles in 12 different categories. Maintaining the ratio of 87.5%:10%:2.5%, the whole dataset is split into Training Set, Validation Set, and Testing Set consisting of 35000, 4000, and 1000 samples, respectively. Dataset overview is shown in Table 1.

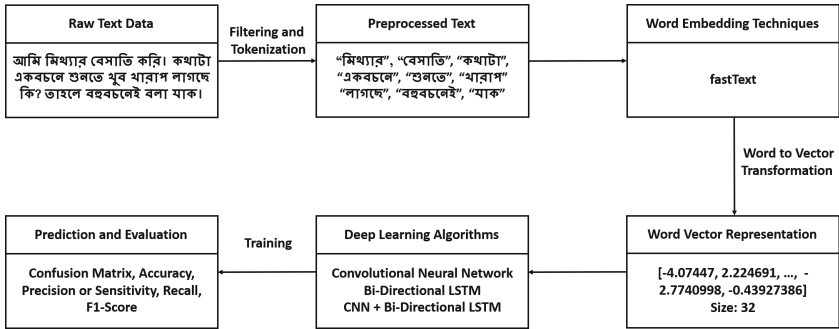
### 3.6 Architecture of the Work

In this project, raw Bangla text data is filtered with regular expression. With this filtering, common Bangla stopwords and noisy characters were removed. After filtering, each sample is tokenized into words. After tokenization, 100 words are

<sup>1</sup> <https://www.kaggle.com/zshujon/40k-bangla-newspaper-article>.

**Table 1.** Dataset overview

Categories	Frequencies	% Ratio of total
Art and literature	368	0.92
Bangladesh	12239	30.60
Durporobash	176	0.44
Economy	4771	11.93
Education	774	1.93
Entertainment	2448	6.12
International	1835	4.59
Life-style	1121	2.80
North America	189	0.47
Opinion	10611	26.53
Sports	3354	8.39
Technology	2113	5.28
Total	<b>39999</b>	100

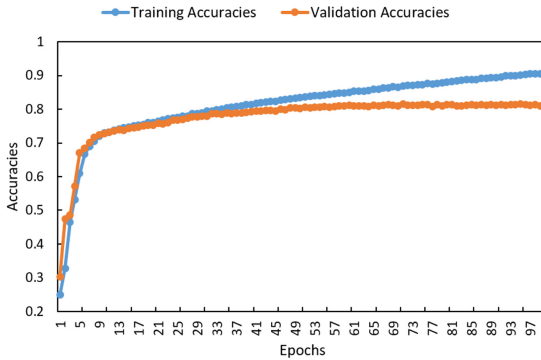
**Fig. 3.** Proposed architecture of the work

taken for each article or sample. If any sample were lacking 100 words, then the remaining words were padded. These tokenized words are then represented in vector space using word embedding techniques fastText. Each word is represented into 32 sized word vector. Hence each sample contains  $100 \times 32$  word vectors. These word vectors of each sample are trained with Deep Learning techniques such as CNN, BLSTM, and CBLSTM. With the combination of embedding techniques and deep learning techniques totally, three models are trained with features. After training, each model is evaluated with the accuracy value. Other metrics: Confusion Matrix, Precision or Sensitivity, Recall, and F1-Score are shown in Sect. 4.2. The overall work-flow or architecture of this work is represented in Fig. 3.

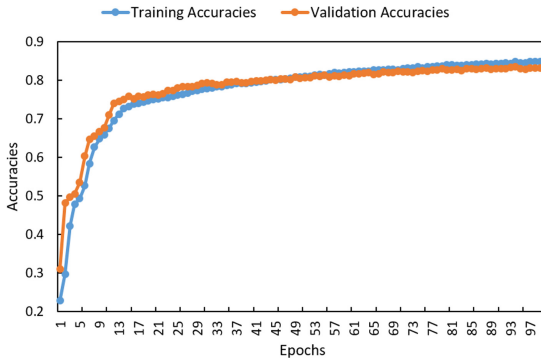
## 4 Experimental Setup and Result Analysis

### 4.1 Experimental Setup

In this work, 32-dimensional word vectors were trained using Gensim fastText implementation [19]. Neural networks are implemented using Keras [6]. We have used Adam optimizer [12] for optimization of the loss or cost function. For the classification task, all the neural network models are trained in Google Colab. Colab offers 1xTesla K80 GPU, having 2496 CUDA cores, compute 3.7, 12 GB (11.439 GB Usable) GDDR5 VRAM for leveraging deep learning techniques. Besides Gensim and Keras, we have also used numpy and matplotlib library for numerical computation and data visualization, respectively.



**Fig. 4.** Accuracies vs epochs for CNN

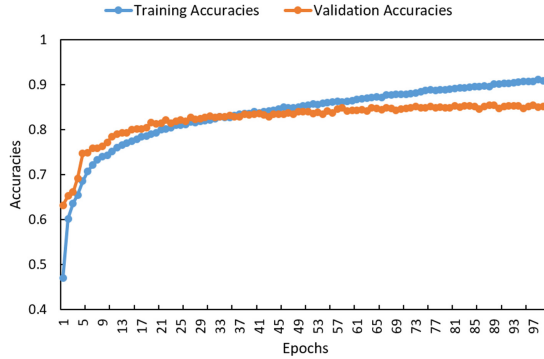


**Fig. 5.** Accuracies vs epochs for CBLSTM

### 4.2 Result Analysis

In this section, we present performance measures from three models to evaluate the impact of fastText word embedding on document classification. Besides, the results have been compared with previous works.





**Fig. 6.** Accuracies vs epochs for BLSTM

The first model is CNN, which acquires 80.2% testing accuracy. From the observation of the train vs validation accuracy in Fig. 4, in the beginning, training and validation accuracies sharply increased, and validation accuracies were slightly higher. After 12 epochs, validation accuracy started to increase slowly than training accuracy, and in the end, there is a clear difference between training accuracy and validation accuracy (Table 2). That means the model losing the ability to predict new data, which is indicating the over-fitting problem.

The second model is CBLSTM, which acquires 84.3% testing accuracy. From Fig. 5 we see that training and validation accuracy curves are overlapped. Training accuracy (85.01%) is also close to testing accuracy (84.3%). It indicates that the model is not losing the ability to predict new data.

Our third model is BLSTM, which acquires 85.5% testing accuracy, and in terms of testing accuracy, this is the best. But if we see Fig. 6 validation accuracy is not increasing with testing accuracy after 29 epochs. At the end of the training, validation accuracy is less than the testing accuracy. It indicates the model is suffering over-fitting. Comparisons among models have been shown in Table 2.

In terms of bias-variance, the second model (CBLSTM) is the best. However, in terms of testing accuracy, the third model (BLSTM) is the best. For the third model confusion matrix have been shown in Table 4 and evaluation metrics in Table 5. In the work [15], it has been suggested that stemming is an important part of reducing dimension. In our work, fastText word embedding facilitates

**Table 2.** Comparative analysis on different models

Network type	Training accuracy	Validation accuracy	Testing accuracy
CNN	0.9146	0.8080	0.802
CBLSTM	0.8501	0.8323	0.843
BLSTM	0.9149	0.8787	0.855

**Table 3.** Comparative description with other works

Author and year	Dataset	Number of class	Model	Evaluation metric
Mandal and Sen [15]	Own	5	TFIDF	
			NB	FS 85.22
			KNN	FS 74.24
			DT	FS 80.65
			SVM	FS 89.14
Chakraborty and Huda [4]	Own	16	TFIDF	
			DenseNN	FS 84.0
Ahmed and Amin [1]	Own	7	Word2vec	
			SVM	FS 91.0
Kabir et al. [10]	Own	9	SGD	N/A
Alam et al. [2]	BARD	5	Word2Vec	
			LR	FS 95.0
			NN	FS 96.0
Our Study	40k News Article	12	fastText	
			CNN	FS 80.0
			CBLSTM	FS 84.0
			BLSTM	FS 85.0

FS = F1-Score, SGD = Stochastic Gradient Decent

LR = Linear Regression, NN = Neural Network, N/A = Not Available

CBLSTM = CNN + Bi-directional LSTM, BLSTM = Bi-directional LSTM

**Table 4.** Confusion Matrix of Testing Set using fastText + Bi-Directional LSTM

	ARL	BND	DUR	ECO	EDU	ENT	INT	LIF	NRA	OPN	SPR	TEC
ARL	4	0	0	0	0	2	0	0	0	2	0	0
BND	1	252	0	6	3	3	2	0	0	20	1	1
DUR	1	1	0	0	0	1	1	1	0	3	0	0
ECO	0	9	0	101	0	0	0	0	0	9	0	4
EDU	2	2	0	4	12	0	0	0	0	1	0	1
ENT	0	1	0	0	0	62	1	0	0	1	1	0
INT	0	5	0	0	0	2	42	0	0	8	0	1
LIF	0	2	0	0	1	2	0	28	0	4	0	0
NRA	0	0	0	0	0	0	1	0	0	2	0	0
OPN	1	10	0	0	1	1	1	3	0	237	2	0
SPR	0	2	0	0	1	1	0	0	0	2	73	0
TEC	0	1	0	3	1	0	1	1	0	0	0	44

ARL = Art & Literature, BND = Bangladesh, DUR = Durporobash, ECO = Economy, EDU = Education, ENT = Entertainment, INT = International, LIF = Life-Style, NRA = North America, OPN = Opinion, SPR = Sports, TEC = Technology

**Table 5.** Evaluation Metrics of Testing set for FastText + Bi-Directional LSTM

Labels	Precision	Recall	F1-score	Support
	$\frac{TP}{FP + TP}$	$\frac{TP}{FN + TP}$	$\frac{2(Recall * Precision)}{(Recall + Precision)}$	
	Art and literature	0.44	0.50	
Bangladesh	0.88	0.87	0.88	289
Durporobash	0.00	0.00	0.00	8
Economy	0.89	0.82	0.85	123
Education	0.63	0.55	0.59	22
Entertainment	0.84	0.94	0.89	66
International	0.86	0.72	0.79	58
Life-style	0.85	0.76	0.80	37
North America	0.00	0.00	0.00	3
Opinion	0.82	0.93	0.87	256
Sports	0.95	0.92	0.94	79
Technology	0.86	0.86	0.86	51

dimensional reduction and lessen the task of stemming. We have presented comparisons with previous works in Table 3.

## 5 Conclusions

This work presents the effective Bangla document classification method with the fastText word embedding technique. A relatively large dataset was used in this work where articles are classified into 12 classes. The work shows that with fastText word embedding significant performance can be gained without some preprocessing like lemmatization, stemming and others. The results from three different models (CNN, CBLSTM, BLSTM) show that the LSTM based approach can gain better performance (CBLSTM 84.3%, BLSTM 85.5%) than the other approach (CNN 80.2%). The over-fitting problem was encountered for the first and the third model due to class imbalance in the dataset. In the future, to improve the performance, other better word embedding techniques with a different deep learning approach need to investigate for Bangla document classification.

## 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. Alam, M.T., Islam, M.M.: Bard: Bangla article classification using a new comprehensive dataset. In: 2018 International Conference on Bangla Speech and Language Processing (ICBSLP), pp. 1–5. IEEE (2018)
3. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. *Trans. Assoc. Comput. Linguist.* **5**, 135–146 (2017)
4. Chakraborty, M., Huda, M.N.: Bangla document categorisation using multilayer dense neural network with TF-IDF. In: 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), pp. 1–4. IEEE (2019)
5. Chiu, J.P., Nichols, E.: Named entity recognition with bidirectional LSTM-CNNs. *Trans. Assoc. Comput. Linguist.* **4**, 357–370 (2016)
6. Chollet, F.: *Deep Learning with Python*, 1st edn. Manning Publications Co., Greenwich (2017)
7. Figueiredo, F., Rocha, L., Couto, T., Salles, T., Gonçalves, M.A., Meira Jr., W.: Word co-occurrence features for text classification. *Inf. Syst.* **36**(5), 843–858 (2011)
8. Goldberg, Y.: Neural network methods for natural language processing. *Synth. Lect. Hum. Lang. Technol.* **10**(1), 117–118 (2017)
9. Jozefowicz, R., Zaremba, W., Sutskever, I.: An empirical exploration of recurrent network architectures. In: International Conference on Machine Learning, pp. 2342–2350 (2015)
10. Kabir, F., Siddique, S., Kotwal, M.R.A., Huda, M.N.: Bangla text document categorization using stochastic gradient descent (SGD) classifier. In: 2015 International Conference on Cognitive Computing and Information Processing (CCIP), pp. 1–4. IEEE (2015)
11. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint [arXiv:1408.5882](https://arxiv.org/abs/1408.5882) (2014)
12. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980) (2014)
13. Li, C.H., Park, S.C.: An efficient document classification model using an improved back propagation neural network and singular value decomposition. *Expert Syst. Appl.* **36**(2), 3208–3215 (2009)
14. Lilleberg, J., Zhu, Y., Zhang, Y.: Support vector machines and Word2vec for text classification with semantic features. In: 2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI\*CC), pp. 136–140. IEEE (2015)
15. Mandal, A.K., Sen, R.: Supervised learning methods for Bangla web document categorization. arXiv preprint [arXiv:1410.2045](https://arxiv.org/abs/1410.2045) (2014)
16. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint [arXiv:1301.3781](https://arxiv.org/abs/1301.3781) (2013)
17. 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)
18. Olah, C.: Understanding LSTM networks (2018). <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>
19. 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, May 2010. <http://is.muni.cz/publication/884893/en>
20. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. *Inf. Process. Manag.* **24**(5), 513–523 (1988)

21. Sumit, S.H., Hossan, M.Z., Al Muntasir, T., Sourov, T.: Exploring word embedding for Bangla sentiment analysis. In: 2018 International Conference on Bangla Speech and Language Processing (ICBSLP), pp. 1–5. IEEE (2018)
22. Young, T., Hazarika, D., Poria, S., Cambria, E.: Recent trends in deep learning based natural language processing. *IEEE Comput. Intell. Mag.* **13**(3), 55–75 (2018)
23. Zhang, D., Xu, H., Su, Z., Xu, Y.: Chinese comments sentiment classification based on Word2vec and SVMperf. *Expert Syst. Appl.* **42**(4), 1857–1863 (2015)