Automatic Bengali Document Categorization Based on Deep Convolution Nets

Md. Rajib Hossain and Mohammed Moshiul Hoque

Abstract Automatic document categorization has gained much attention by natural language processing (NLP) researches due to the enormous availability of text resources in digital form in recent years. It is the process of assigning a document into one or more categories that help the document manipulate and sort quickly. An efficient information processing system is required due to the rapid growth of Bengali text contents in digital form for searching, organizing, and retrieving tasks. In this paper, we proposed a framework for classifying Bengali text documents using deep convolution nets. The proposed framework consists of word embedding and document classifier models. Experiments with more than 1 million Bengali text documents reveals that the proposed system worthy of classifying documents with 94.96% accuracy.

Keywords Bengali language processing · Document categorization · Word embedding · Deep convolution neural nets

1 Introduction

Automatic document categorization is a challenging task in the field of NLP where a text document or a sequence of text documents automatically assigned into a set of predefined categories. Bengali language is spoken by about 245 million people in all over the world and is being considered the seventh most spoken language in the world [1]. Number of Bengali text documents in digital form have grown rapidly day by day in size and variety due to the increased usability of the Internet. It is very difficult to manage such huge amount of text documents for human expert manually

Md. R. Hossain · M. M. Hoque (✉)
Department of Computer Science & Engineering, Chittagong University of Engineering & Technology (CUET), Chittagong 4349, Bangladesh
e-mail: moshiulh@yahoo.com

Md. R. Hossain
e-mail: rajcsecuet@gmail.com

N. R. Shetty et al. (eds.), *Emerging Research in Computing, Information, Communication and Applications*, Advances in Intelligent Systems and Computing 882,
https://doi.org/10.1007/978-981-13-5953-8_43

that also consume a lot of time and cost of money. Therefore, an automatic document categorization system will be developed to handle a large amount of text data so that documents can be organized, manipulated, or sorted easily and quickly.

Although very few research activities have been conducted on Bangla language processing (BLP) such as syntax analysis, English to Bangla MT, Bangla OCR, and so on. Bengali text document categorization is also an important research issue that needs to be solved. There are many linguistic and statistical approaches that have been developed for automatic documents categorization of English and European languages. However, no usable and effective system is developed for classifying the Bangla texts still today. Bangla document categorization system may be used by security agency to identify the suspected streamed web data or spam detection, the daily newspapers to organize news by subject categories, the library to classify papers or books, the hospitals to categorize patient based on diagnosis reports, archiving, or clustering the government/nongovernment organization data, improve Bengali content searching, retrieving or mining specific web data, and so on.

In this paper, we proposed a framework for automatic Bengali documents categorization that works with word embedding model and deep convolution neural networks (DCNNs) [2, 3]. The proposed model will be able to overcome the traditional document classification shortcomings and also hardware cost. The word embedding algorithm such as Word2Vec extracts semantic feature for each word and represents as 1D vector. The semantic feature carries the actual meaning with respect to surrounding words and word order. Each document represents a 2D feature vector where the rows represent the word and columns represent the feature values. There are lots of hyperparameters in Word2Vec algorithm and we tune these parameters for Bengali corpus and achieved the better accuracy with respect to other language corpora. In the recent year, the convolution neural networks (CNNs) achieved the very good result for English and some other languages [4]. We design a DCNNs architecture where each hyperparameters will be well trained for Bengali large-scale data set and generate a classifier model. The model obtained the better accuracy for categorization of Bengali text documents.

2 Related Work

A significant amount of researches has been conducted on document categorization in English and European languages. In the recent year, the Word2Vec [5–7] and Glove [8] algorithms achieved the state-of-the-art word embedding result for English, French, Arabic, and Turkish languages. The CNN- and RNN-based documents classifier approaches achieved 84.00 and 85.60% accuracy from English text [9]. Conneau et al. [3] introduced very deep CNN and achieved 96–98% accuracy from different English data set classification. In hierarchical, CNN (HCNN)-based and decision-tree-based CNN also have been achieved higher accuracy for English text [10, 11]. Stochastic gradient descent (SGD) based classifier perform lower accuracy due to feature scaling and lack of huge hyperparameters tuning [12]. The character-level

CNN is performing slower embedding system and memory consuming [13]. There are very few researches have been conducted on Bengali text document classification. The TF-IDF-based features are the traditional technique which only depends on documents term frequency or BoW model [1, 12, 14]. Lexical feature only carries the limited number of information such as average sentence length, average number of words length, number of different words, and so on [15]. TF-IDF feature performs lower accuracy due to absence of semantic information. The TF-IDF feature not contained the word position and correlation of the word. The lexical and TF-IDF feature are not working properly for Bengali language due to its large inflectional diversity in verbs, tense, noun, etc. In our work, we use DCNNs for Bengali documents categorization. This approach showed better performance than the previous Bengali text classification methods due to the hyperparameter tuning and deep network training architecture.

3 Methodology

The proposed document categorization architecture consists of three main modules: text to feature extraction module, documents classifier training module, and documents classifier projection module.

3.1 Text to Feature Extraction Module

Word2Vec with skip-gram algorithm is used to train a Bengali word embedding model. In order to train the model, we tune the window size and embedding dimension. The tuned hyperparameter shows the better result for Bengali text classification purpose. The embedding model row numbers represent the number of distinct words and columns represent the feature dimension. First split the sentence to word list for each text document and for each word feature vector is extracted from embedding model. Therefore, for each document, a 2D feature vector is generated.

3.2 Documents Classifier Training Module

Figure 1 shows the overall architecture of Bengali text document classifier model. The DCNNs is consisting of input layer, convolution layer, rectified linear units (ReLU) layer, pooling layer, and fully connected (FC) layer. $S(R, C, F)$ represents the input and output tensor where R represents the number of rows, C for the number of columns, and F for the depth or the number of filters.

Input layer: The input document has sequentially passed each word through the look-up table ($W_{v \cdot d}$) and generates a $n \times d$ representation vector, where n represents

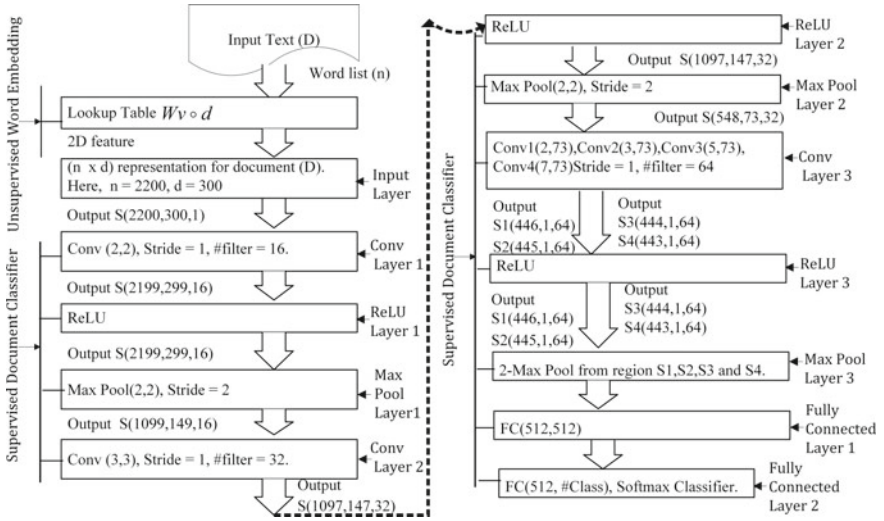


Fig. 1 DCNNs layered architecture

the number of word in document and d represents the feature dimension. In our case, n is fixed to 2200 and d to 300. If the number of word shorter than 2200 then a padding operation will be imposed. The 2D representation $S(2200, 300, 1)$ is fed to the input tensor that will be known by the input layer.

Convolution layer: The convolution layer extracts the local feature from the input tensor and trained the filter weights. The ith layer height and width are calculated by the following Eqs. (1)–(3).

$$D_i^l = D \tag{1}$$

$$H_i^l = H_{i-1}^{l-1} - D_h^l + 1 \tag{2}$$

$$W_i^l = W_{i-1}^{l-1} - D_w^l + 1 \tag{3}$$

Here, padding = 0 and stride = 1. H_i^l, W_i^l represents the output layer height and width, H_{i-1}^{l-1}, W_{i-1}^{l-1} represents the input layer height and width, respectively. The D denoted the number of filter in the current layer. D_h^l,D_w^l represents the filter height and width. Each of the convolution operation follow a Eq. (4).

$$S_{i^l j^l, d^l}^l = \sum_{i=0}^{H_i^l} \sum_{j=0}^{W_i^l} \sum_{d^l=0}^{D_i^l} S_{i,j,d}^{l-1} \times K_{i,j,d^l}^l \tag{4}$$

where K_{i,j,d^l}^l represents the kernel of each filter. In this architecture, convolution operation is imposed in three different layers. The first operation is applied at input tensor $S(2200, 300, 1)$ with 16 filters and kernel size $(2, 2)$ with stride is to

one, and output tensor is to $S(2199, 299, 16)$. The second operation is applied with 32 filter and kernel size $(3, 3)$ with stride is to one, and the output tensor is to $S(1097, 147, 32)$. The last operation is applied with four different kernel sizes such as Conv1(2, 73), Conv2(3, 73), Conv3(5, 73), and Conv4(7, 73) with zero padding and stride is to one only for heightwise and 64 filters. The output produces four tensors with $S1(446, 1, 64)$, $S2(445, 1, 64)$, $S3(444, 1, 64)$, and $S4(443, 1, 64)$.

Rectified Linear Units (ReLU) layer: The rectified linear units (ReLUs) are a special operation that combines nonlinearity and rectification layers in DCNNs. The input and output volumes remain same in this layer only changing the elementwise value.

$$S_i^l = \max(0, S_i^{l-1}) \tag{5}$$

Here, $\max(0, S_i^{l-1})$ denotes the elementwise operation. ReLU propagates the gradient efficiently, and therefore reduces the likelihood of a vanishing gradient problem which in turn reduces the time complexity. The ReLU sets negative values to zero, and therefore solved the cancellation problem.

Pooling layer: The pooling or feature reduction layer is responsible for reducing the volume of activation maps. They are used for reducing the computational requirements progressively through the network as well as minimizing the likelihood of overfitting. In the DCNNs architecture, the max polling operation is imposed in three different layer. In the pooling layer, the input and output are calculated by Eqs. (6)–(7).

$$H^l = \frac{(H^{l-1} - K_h^l + 2P^l)}{T^l} + 1 \tag{6}$$

$$W^l = \frac{(W^{l-1} - K_w^l + 2P^l)}{T^l} + 1 \tag{7}$$

where H^l and W^l are the output tensor height and width of pooling layer. K_h^l, K_w^l, and T^l are the pooling layer polling height, width, and stride size. In the DCNNs architecture, the padding parameter P is to zero and the first pooling layer input tensor $S(2199, 299, 16)$ with max pool$(2, 2)$, stride is to two, and the output tensor size is to $S(1099, 149, 16)$. The second polling layer input tensor is $S(1097, 147, 32)$ with max pool$(2, 2)$, stride is to 2 and the output tensor size is $S(548, 73, 32)$. In the last pooling layer, the notation 2-max pool from each region means the pool 2 nonoverlapping feature from each feature map. The output tensor of the pooling layer having $2 * 4 * 64 = 512$ feature.

Fully connected (FC) layer: The fully connected or dense layer is a last layer of DCNNs. It follows the convolution, ReLU, and max pooling layer. The main goal of the layer is to design a flatten tensor which is input tensor activation map in three-dimensional volume. The transformation from input to output tensor of FC layer is $S(R, C, F) \rightarrow S(R * C * F, 1)$, where R, C, and F denote the input tensor rows ,columns, and number of filters. In a FC layer, every node in the layer is connected to each other in the preceding layer. In Fig. 1, the first $FC(512, 512)$ means that the input tensor $S(512, 1)$ is connected to $S(512, 1)$. The second FC layer $F(512, \#\text{Class})$

means that each of the class got a classification score using Softmax classifier. The value of the each node in FC layer is calculated by Eq. (8).

$$\theta_i^l = \sum_{j=1}^{M} W_{i,j}^l A_i^{l-1} \tag{8}$$

where $W_{i,j}^l$ represents the weight of current layer l and A_i^{l-1} represents the previous layer activation maps. The θ_i^l calculates the projected value.

$$A_i^l = F(\theta_i^l) \tag{9}$$

Here, A_i^l represents the current layer activation maps.

3.3 Documents Classifier Projection Module

In this module, unlabeled text data is taken as input and it is divided into word and if the input is less then 2200 words then padding is added. For each word is look-up by embedding model and generate a 2D feature vector where each row represents the word and corresponding column represents the feature vector. The DCNNs architecture trained a classifier model (pretrained model) which saves the layerwise different filter data. The DCNNs is initialized by the trained model and 2D feature is projected by the DCNNs architecture, and finally produces a score vector which means that by classwise expected score. The output of FC layer is projected by a weight matrix W with C distinct linear classification, and the predicted probability for the ith class given by Eq. (10).

$$P(\text{class} = i|X) = \frac{(e^{X^T W_i})}{\sum_{c=1}^{C} e^{X^T W_c}} \tag{10}$$

where X is a ith class feature vector and W is the trained weight matrix. For each unlabeled text data, the system provides 12 significant score, from this score we select the max value which is desired class value.

4 Experiments

We implemented DCNNs algorithm in Python with TensorFlow and ran the experiments on a Nvidia GeForce GTX 1070 GPU. This GPU has 8 GB of GPU RAM, which helps us for extending the networks and batch size. A GPU-based computer with 32 GB physical memory and Intel Core i7-7700K CPU with 256 GB SSD is

used for experiments. The DCNNs architecture has lots of hyperparameters. The proposed system is suited for better performance with the following parameters: number of batch size = 32, L2 regularization lambda = 0.0001, number of training epochs= 200, word embedding = Word2Vec, decay coefficient = 1.5, dropout keep probability = 0.50, save model after this many steps = 100, evaluate model on development set after this many steps = 100, and convolutions filter initialization method = Xavier initialization.

4.1 Corpus

Resource acquisition is one of the challenging hurdles to work with electronically low resource languages like Bengali. Due to the lack of available Bengali corpus for text categorization, we have built our own corpus and data in the corpus have extracted from the available online Bengali resources such as blogs and newspapers [16–19]. The unlabeled data is stored in the corpus for word embedding purposes. We have collected 836412 Bengali unique words. If a word is not found in the embedding table, then applying zero-value padding technique. Table 1 shows the summary of the dataset.

For the classification tasks, handcrafted labeled data are collected from Bengali online newspapers. Table 2 shows the statistics of labeled dataset. The number of training and testing documents are varied in each category with variable word length. The label of the training data is assigned by the human expert that help to achieve better accuracy. In the label corpus, if the number of words is lower than 2200 then zero padding is added to each of the input document.

4.2 Evaluation Measures

In order to evaluate the proposed Bengali document categorization system, we used the following measures: development/training phases loss versus iteration, develop-

Table 1 Embedding data summary

Number of documents	113082
Number of sentence	220000
Number of words	33407511
Number of unique words	836412
Embedding type	Word2Vec
Contextual window size	12
Word embedding dimension	300

Table 2 Summary of the categorical data

Category name	#Training documents	#Testing documents
Accident (AC)	6069	402
Art (AR)	1264	146
Crime (CR)	11,322	1312
Economics (EC)	4526	743
Education (ED)	5159	865
Entertainment (ET)	8558	1734
Environment (EV)	923	110
Health (HE)	2004	580
Opinion (OP)	6479	1248
Politics (PL)	24,136	1834
Science & Technology (ST)	3106	694
Sports (SP)	12,653	1039
Total	86,199	10,707

ment phase/training phases accuracy versus iteration. The testing phase statistical analysis is shown by precision, recall, F_1-measure, and accuracy.

Training and development phase evaluation: The loss and accuracy of training and development imply the model beauty or better fitness. The loss and accuracy is the summation of the errors made for each example in training and development sets. In each training iteration, the loss is calculated by the following equation:

$$\text{Loss}_i = -W * X_i^T + \sum_{c=1}^{C} e^{W_c * X_i^T} \tag{11}$$

where the Loss_i means the ith iteration loss mean value and W represents the Softmax layer weight matrix, where rows and column represent the feature and category values, respectively. The X_i represents the ith example feature vector. The C represents the total category of our system. The training accuracy is calculated by Eq. (12).

$$\text{Acc}_i = \frac{P_i}{M_i} \tag{12}$$

Here, Acc_i, P_i, and M_i represent the ith iteration accuracy value, total number of correctly predicted category, and total number of sample data point in that iteration. The development loss and accuracy are also calculated using Eqs. (11) and (12).

Testing phase evaluation: In order to measure the overall performance of the proposed system, we used precision, recall, accuracy, and F_1-measure [Eqs. (13)–(16)].

$$\textbf{Precision} = \frac{T_p}{T_p + F_p} \tag{13}$$

$$\textbf{Recall} = \frac{T_p}{T_p + F_n} \tag{14}$$

$$\textbf{Accuracy} = \frac{T_p + T_n}{T_p + F_p + T_n + F_n} \tag{15}$$

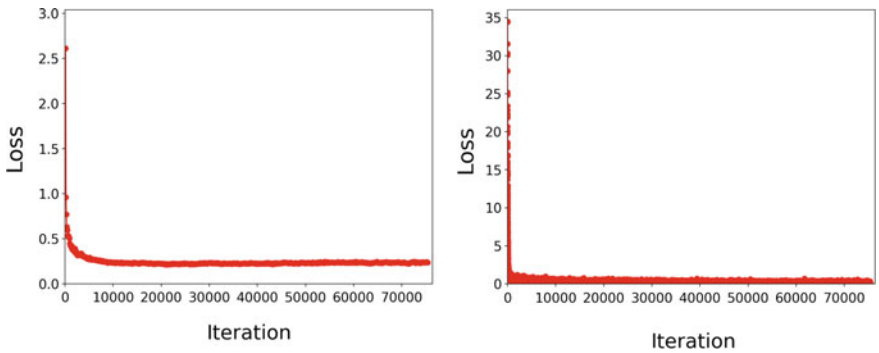
$$\textbf{F}_1\textbf{-measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Recall} + \text{Precision}} \tag{16}$$

where T_p, T_n, F_p, and F_n represent the true positive, true negative, false positive, and false negative, respectively.

5 Results

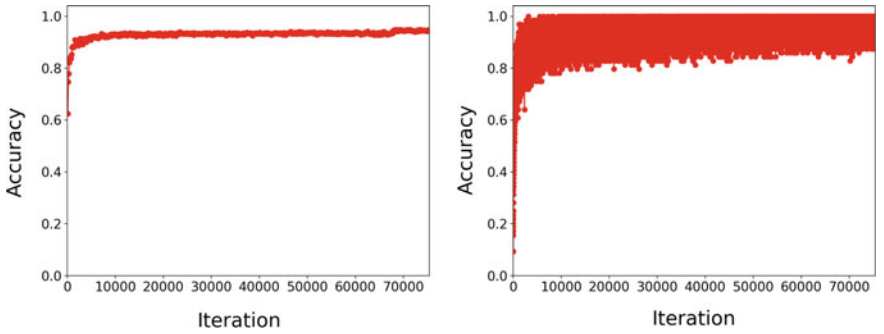
In order to check the effect of size of training data on the performance, we plotted two learning curves: loss versus iterations and accuracy versus iterations. Moreover, we applied the ROC measure for the testing phase accuracy.

– **Development and training loss**: Figure 2 shows the impact of number of iterations on loss in development and training phases. In the development phase at iteration number 1 loss is more than 2.5 and loss is decreased with the increase of iteration. The decreasing rate of loss is stable at iteration number 20,000 (Fig. 2a). In that case, we trained on the training data only and test on the tested data. In the training phase, at iteration number 1 the loss is more than 35 and the loss is decreasing from the iteration number 5000 (Fig. 2b). This decreasing rate remains stable after



(a) Development loss (b) Training loss

Fig. 2 Losses in development and training phases



(a) Development accuracy **(b)** Training accuracy

Fig. 3 Accuracy with iteration in development and training phases

10,000 iterations. In this case, we trained on a set consisting of both, training and development data, and test on the test data.

– **Impact of number of iteration on accuracy**: Figure 3 illustrates the impact of number of iterations on accuracy. Figure 3a shows that the development accuracy is increasing with respect to iteration number and varies from 87.00% (iteration no. 8000) to 93.00% (iteration no. 65000). However, the accuracy is almost stable from iteration number 68,000 and remains constant with accuracy about 94.50%. Figure 3b shows that the training accuracy is increasing with respect to the iteration numbers.

5.1 Testing Performance

Table 3 shows the precision, recall, and F_1-measure of the proposed system. The maximum accuracy is achieved by the entertainment (ET) class and the minimum accuracy is achieved by the environment (EV) class.

5.2 Comparison with Existing Approaches

To evaluate the effectiveness, we compare the proposed system with existing approaches [1, 5]. Table 4 summarizes the comparison performance. This result shows that the proposed system outperforms the previous work in terms of accuracy.

– **ROC measures**: Figure 4 shows the ROC curve for multiclass Bengali documents categorization. This curve reveals that the maximum area under the curve covered by both classes SP and ST, whereas the minimum area covered by the EV class

Table 3 Summary of the analysis

Category name	Precision	Recall	F_1-score	Support
HE	0.90	0.88	0.89	580
AC	0.90	0.89	0.89	402
AR	0.81	0.83	0.82	146
CR	0.99	0.92	0.95	1312
EC	0.93	0.94	0.94	743
ED	0.98	0.93	0.95	865
ET	0.99	0.99	0.99	1734
EV	0.85	0.62	0.72	110
OP	0.96	0.96	0.96	1248
PL	0.93	0.98	0.96	1834
ST	0.93	0.98	0.95	694
SP	0.94	0.97	0.96	1039
Avg./total	0.95	0.95	0.95	10,707

Table 4 Performance comparison

Method	#Training documents	#Testing documents	#Category	Accuracy (%)
TF-IDF + SVM [1]	1000	118	5	89.14
Word2Vec+K-NN+SVM [5]	19,750	4713	7	91.02
Proposed	86,199	10,707	12	94.96

(0.87). According to area under cover value, ten categories go to excellent class and only one class (EV) falls into good class and there is no category into bad class.

6 Conclusion

Text document classification is a well-studied problem for the highly resourced languages like English. However, it is a relatively new problem for an under-resourced language, especially in Bengali. Bengali text document categorization is a challenging task due to the lack amount of digitized text, and scarcity of available corpora. In this work, we introduce a new technique for Bengali document categorization system based on DCNNs. In this system, semantic features are extracted by Word2Vec algorithm and classifier is trained by DCNNs. We used 86,199 documents for training and 10,707 documents for testing and both sets have been handcrafted from online newspapers. The system obtains 94.96% accuracy for text document classification

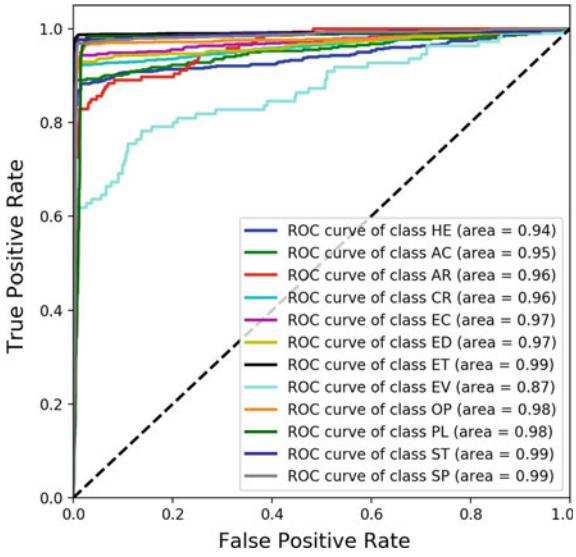


Fig. 4 Receiver operating characteristic (ROC)

and shows the better performance compared to the existing techniques. For future research, we will extend our framework for other forms of text classification such as books, blogs, tweets, and online forum threads. Moreover, we will also include more classes with more data to improve the overall performance of the system.

References

1. Mandal, A. K., & Sen, R. (2014). Supervised learning methods for Bangla web document categorization. *International Journal of Artificial Intelligence and Applications (IJAIA)*, *5*(5), 93–105.
2. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *Journal of CoRR*.
3. Conneau, A., Schwenk, H., & Cun, Y. L. (2017): Very deep convolutional networks for text classification. In *The 15th Conference of the European Chapter of the Association for Computational Linguistics* (Vol. 1, pp. 1107–1116), Valencia, Spain.
4. Xu, K., Feng, Y., Huang, S., & Zhao, D. (2015). Semantic relation classification via convolutional neural networks with simple negative sampling. In *Empirical Methods in Natural Language Processing* (pp. 536–540), Lisbon, Portugal.
5. Ahmad, A., & Amin, M. R. (2016). Bengali word embeddings and its application in solving document classification problem. In *19th International Conference on Computer and Information Technology* (pp. 425–430).
6. Johnson, R., & Zhang, T. (2017). Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)* (Vol. 1, pp. 562–570).
7. Tang, D., Qi, B., & Liu, T. (2015). Document modeling with gated recurrent neural network for sentiment classification. In *Empirical Methods in Natural Language Processing* (pp. 1422–1432), Lisbon, Portugal.

8. Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543).
9. Lee, J. Y., & Dernoncourt, F. (2016). Sequential short-text classification with recurrent and convolutional neural networks. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)* (pp. 515–520).
10. Kowsari, K., Brown, D. E., Heidarysafa, M., Meimandi, K. J., & Barnes, L. E. (2017). Hierarchical deep learning for text classification. In *16th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 364–371).
11. Bahassine, S., Madani, A., & Kissi, M. (2017). Arabic text classification using new stemmer for feature selection and decision trees. *Journal of Engineering Science and Technology*, *12*, 1475–1487.
12. Kabir, F., Siddique, S., Kotwal, M., & Huda, M. (2015, March). Bangla text document categorization using stochastic gradient descent (SGD) classifier. In 2015 International Conference on Cognitive Computing and Information Processing (CCIP) (pp. 1–4).
13. Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolution networks for text classification. In *NIPS'15 28th International Conference on Neural Information Processing Systems* (Vol. 1, pp. 649–657).
14. Krendzelak, M., & Jakab, F. (2015). Text categorization with machine learning and hierarchical structures. In *Proceedings of 13th International Conference on Emerging eLearning Technologies and Applications* (pp. 1–5).
15. Liebeskind, C., Kotlerman, L., & Dagan, I. (2015). Text Categorization from category name in an industry motivated scenario. *Journal of Language Resources and Evaluation*, *49*(2), 227–261.
16. The Daily Prothom Alo, Online, http://www.prothom-alo.com.
17. The Daily Jugantor, Online, https://www.jugantor.com.
18. The Daily Ittefaq, Online, http://www.ittefaq.com.bd.
19. The Daily Manobkantha, Online, http://www.manobkantha.com.