applied sciences

MDPI

Article

An Enhanced Neural Word Embedding Model for Transfer Learning

Md. Kowsher [1], Md. Shohanur Islam Sobuj [2], Md. Fahim Shahriar [2], Nusrat Jahan Prottasha [3], Mohammad Shamsul Arefin [4,*], Pranab Kumar Dhar [4] and Takeshi Koshiba [5,*]

1 Department of Computer Science, Stevens Institute of Technology, 1 Castle Point Terrace, Hoboken, NJ 07030, USA; ga.kowsher@gmail.com
2 Department of Computer Science and Engineering, Hajee Mohammad Danesh Science and Technology University, Dinajpur 5200, Bangladesh; shohanursobuj@gmail.com (M.S.I.S.); fahimshahriar6832@gmail.com (M.F.S.)
3 Department of Computer Science and Engineering, Daffodils International University, Dhanmondi, Dhaka 1207, Bangladesh; jahannusratprotta@gmail.com
4 Department of Computer Science and Engineering, Chittagong University of Engineering and Technology, Chottagram 4349, Bangladesh; pranabdhar81@cuet.ac.bd
5 Faculty of Education and Integrated Arts and Sciences, Waseda University, 1-6-1 Nishiwaseda, Shinjuku-ku, Tokyo 169-8050, Japan
* Correspondence: sarefin@cuet.ac.bd (M.S.A.); tkoshiba@waseda.jp (T.K.)

Abstract: Due to the expansion of data generation, more and more natural language processing (NLP) tasks are needing to be solved. For this, word representation plays a vital role. Computation-based word embedding in various high languages is very useful. However, until now, low-resource languages such as Bangla have had very limited resources available in terms of models, toolkits, and datasets. Considering this fact, in this paper, an enhanced BanglaFastText word embedding model is developed using Python and two large pre-trained Bangla models of FastText (Skip-gram and cbow). These pre-trained models were trained on a collected large Bangla corpus (around 20 million points of text data, in which every paragraph of text is considered as a data point). BanglaFastText outperformed Facebook's FastText by a significant margin. To evaluate and analyze the performance of these pre-trained models, the proposed work accomplished text classification based on three popular textual Bangla datasets, and developed models using various machine learning classical approaches, as well as a deep neural network. The evaluations showed a superior performance over existing word embedding techniques and the Facebook Bangla FastText pre-trained model for Bangla NLP. In addition, the performance in the original work concerning these textual datasets provides excellent results. A Python toolkit is proposed, which is convenient for accessing the models and using the models for word embedding, obtaining semantic relationships word-by-word or sentence-by-sentence; sentence embedding for classical machine learning approaches; and also the unsupervised finetuning of any Bangla linguistic dataset.

Keywords: Bangla NLP; BanglaLM; text classification; toolkit; web crawler; word embedding

check for updates

Citation: Kowsher, M.; Sobuj, M.S.I.; Shahriar, M.F.; Prottasha, N.J.; Arefin, M.S.; Dhar, P.K.; Koshiba, T. An Enhanced Neural Word Embedding Model for Transfer Learning. *Appl. Sci.* **2022**, *12*, 2848. https://doi.org/10.3390/app12062848

Academic Editor: Valentino Santucci

Received: 28 January 2022
Accepted: 6 March 2022
Published: 10 March 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Word embedding, also known as word representation [1], is becoming more important for constructing continuous word vectors based on their context in large corpora. In recent years, word representation or the vector depiction of the words has been demonstrated to achieve major results in the modeling of language and activities involving natural language processing (NLP). Word embedding contains both semantic and syntactic information in words and can be used to measure word similarity in information retrieval (IR) [2] and NLP applications [3].

Because of the availability of major public resources and criteria, most of the existing research is restricted to English and other resource-rich languages. However, Bangla is

the sixth most commonly spoken language on the planet (https://www.ethnologue.com/guides/ethnologue200 (accessed on 4 March 2022)), with more than 268 million speakers. Bangla is the native language of the Bengali people. This work aims to enrich Bangla natural language processing, to address the scarcity of Bangla datasets by the implementation of rigorous Bangla NLP models to translate, analyze, and predict similarities in the Bangla language. There are very few unsupervised Bangla datasets available for building complex models that can yield a good operation. Even the number of samples in the existing dataset is very small, which makes it difficult to train the latest model, including the transformer [4], BERT [5], etc. Therefore, this work utilized a collected and constructed dataset that contains Bangla paragraphs for training an unsupervised ML model called BanglaLM, containing approximately 14 gigabytes of text data.

However, there is also a pre-trained Bangla FastText model available from Facebook, which was trained on the Bangla Wikipedia dataset. However, it is not sufficient enough to show a word-by-word semantic relationship. In addition, it was trained on the much more organized dataset of Wikipedia. Consequently, this paper proposes two BanglaFastText word embedding models (Skip-gram [6] and CBOW), and these are trained on the developed BanglaLM corpus, which outperforms the existing pre-trained Facebook FastText [7] model and traditional vectorizer approaches, such as Word2Vec. The BanglaLM was gathered from a variety of sources, including newspapers, social media sites, blogs, and Wikipedia. The dataset has a total of almost 20 million observations. Since BanglaLM was collected from various sources, the training model on this dataset would test various Bangla real-life linguistic samples. This will be helpful for a better representation of the word-by-word semantic relationship. In addition, it showed an outstanding performance in text classification and clustering, superior to that of the proposed FastText of Facebook. Moreover, it will assist in extending future research on Bangla NLP.

By proposing two FastText pre-trained models, this task has also propounded a Python PyPI library (toolkit) for better and more convenient uses in unsupervised word representation. This toolkit aids in figuring out word-by-word or sentence-by-sentence relationships, sentence embedding, unsupervised fine-tuning, and easy access and downloading of the proposed pre-trained model (https://github.com/Kowsher/Bangla-Fasttext (accessed on 4 March 2022)).

From the modeling perspective and in terms of analyzing performance, both classical (e.g., SVM [8], random forest [9], K-nearest neighbors [10], XGB [11], and logistic regression [12]) and deep learning algorithms (e.g., convolution neural network (CNN) [13], and long short-term memory (LSTM) [14]) were explored. The literature lacks the use of recent successful algorithms, such as Transformer models [5].

In order to draw comparisons, this work used different Bangla text data and vectorized them using different feature extraction techniques (Word2Vec, FastText, and BanFastText). Finally, few machine learning approaches were used to analyze the performance and results. For obtaining vector representations of textual language, these models allow one to design unsupervised learning or supervised learning methods.

The main contributions of this work can be summarized as follows:

- The development of two publicly available pre-trained FastText word embedding models, called BanglaFastText, trained on almost 20 million observations of the BanglaLM (https://www.kaggle.com/gakowsher/bangla-language-model-dataset (accessed on 4 March 2022)) dataset;
- The proposed model shows enhanced performance in semantic relationship and text classification than the existing Facebook FastText model of Bangla;
- The proposed model uses the BanglaLM dataset, containing organized and non-organized trained samples collected from various sources, which help to extract embedding in all document label conditions;
- A proposed Python library for the generation of sentence embedding, word embedding, unsupervised fine-tuning, and the semantic relationships of Bangla FastText.

The remaining parts of the paper are organized as follows: Related studies are presented in Section 2. Section 3 discusses the detailed attributes of the collected BanglaLM dataset, as well as the textual statistical analysis of BanglaLM dataset articles. Subsequently, the proposed methodology is presented in Section 4. Then, Section 5 includes the performance of different datasets and the textual features of classifying Bangla articles. Finally, the concluding remarks of the paper are addressed in Section 6.

2. Related Work

2.1. Word Embedding of Various Languages

Word embeddings are straightforward to use, because they allow for the efficient computation of word similarities using low-dimensional matrix operations. Mikolov et al. [1] proposed Word2Vec for the English language to learn high-quality distributed vector representations and demonstrated its availability in measuring syntactic and semantic word similarity to investigate the semantic relationship between words. The Global Vectors for Word Representation (GloVe) is a frequency-predicated text-to-feature representation technique that captured the semantic and syntactic linguistic features of the English language. Pennington et al. [15] suggested a method of learning vector space representations of words only in a word–word matrix on non-zero elements, instead of on a whole sparse matrix or in individual context windows in a wide corpus. Dash et al. [16] proposed a word-embedding-based named entity recognition (NER) [17] approach, in which they experimented on a morphologically rich and low-resource language. Tang et al. [18] proposed learning sentiment-specific word embedding (SSWE) for sentiment analysis by encoding sentiment information into the continuous representation of words in a system that learns word embedding for Twitter sentiment classifications.

2.2. Bengali Word Embedding

In Bengali word embedding, very few methods have been introduced. Abhishek et al. [19] proposed a Word2Vec-based neural lemmatizer for Bengali word embedding based on 18 randomly selected news articles from the FIRE Bengali News Corpus, each of which contains 3342 surface words (excluding proper nouns). The authors stated that their accuracy was 81.95%. Ritu et al. [20] analyzed the most commonly used Bengali Word2Vec models. They trained their model using multiple data sets and observed differences between the Word2Vec and FastText models. In another study, Amin et al. [21] performed sentiment analysis on Bengali comments, which were analyzed using the Word2Vec approach and achieved 75.5% in each of the two classes. Ahmad et al. [22] used Word2Vec for the Bengali document classification problem using Support Vector Machine (SVM) and obtained an F1-score of almost 91%.

2.3. Bengali Document Classification with FastText

Kunchukuttan et al. [23] used FastText to classify news articles using the IndicNLP corpus and pre-trained word embeddings. Rafat et al. [24] analyzed different word embedding models and trained them on the Skip-gram and CBOW Word2Vec, FastText models. Out of all the results, FastText provided them with a favorable outcome. In another study, Chowdhury et al. [25] evaluated the contribution of different types of word embeddings in Bengali Literature Authorship Attribution, particularly the Skip-gram and CBOW models generated by Word2Vec and FastText, as well as the word vectors generated by GloVe. Chowdhury et al. [26] proposed an Authorship Attribution in Bengali Literature using FastText's technique, in which they achieved 82.4% in authorship attribution as the classifier and n-grams ranging from 1 to 5 as the feature, while training the classifier in only 147 ms.

3. Data Description

In this research, the largest Bangla unsupervised dataset to date was developed, named BanglaLM. This dataset has various lengths of samples, and there are three versions of the dataset based on a few preprocessing steps. The total volume of one version of this dataset

is around 14 GB. This dataset contains text information from a variety of sources, including newspapers, social media platforms, blogs, Wikipedia, and other online publications. The BanglaLM dataset contains approximately 20 million observations. Remarkably, it is available in three versions: (i) raw data, (ii) preprocessed V1, and (iii) preprocessed V2. Preprocessed V1 is better suited to LSTM-based machine learning models, while preprocessed V2 is better suited to statistical models.

In this work, the raw version of the dataset is utilized and preprocessed according to the novel instruction. This dataset contains 1,710,431 unique words and a total of 821,007,301 words, split into training, test, and validation sets. Figure 1 shows the data samples from BanglaLM corpus that were used as the input in the training of the BanFastText model.

Index	Text	In English
0	ইসলামী ব্যাংকের চেয়ারম্যান ভাইস চেয়ারম্যান ও ব্যবস্থাপনা পরিচালক পদে পরিবর্তন এসেছে বৃহস্পতিবার ব্যাংকের পরিচালনা পর্ষদের সভায় এ পরিবর্তন আনা হয় সরকারের উচ্চপর্যায়ের সিদ্ধান্তেই এ পরিবর্তন এসেছে বলে জানা গেছে	On Thursday, Islami Bank's chairman, vice-chairman, and chief executive officer positions were replaced. The modification was made during a meeting of the board of directors. As it is heard, the decision to modify was made by the government's higher authority.
1	রোগ প্রতিরোধ জাম্বুরাতে প্রচুর ভিটামিন সি রয়েছে এটি রক্তনালীর সংকোচন প্রসারণ ক্ষমতা বৃদ্ধি করে এবং ডায়াবেটিস জ্বর নিদ্রাহীনতা পাকস্থলী ও অগ্ন্যাশয়ের বিভিন্ন রোগ নিয়ন্ত্রণে সাহায্য করে এছাড়া কোলস্টেরল নিয়ন্ত্রণ করে বিভিন্ন ধরনের হৃদরোগের হাত থেকে রক্ষা করে জাম্বুরা নিয়মিত জাম্বুরা খেলে কোষ্ঠকাঠিন্য দূর হয় ও পেটের নানা রকম হজমজনিত সমস্যার প্রতিকার হয়	Grapefruit is high in vitamin C, which helps prevent disease. Additionally, erectile dysfunction, diabetes, nausea, vomiting, pneumonia, and osteoporosis are treated with this medication. Additionally, it regulates cholesterol levels and protects against several forms of heart disease. Grapefruit consumption on a regular basis alleviates constipation and treats a variety of stomach-related digestive issues.
2	সাদা শার্ট কালো কোট কালো প্যান্ট কালো জুতো লম্বা টাই পরিপাটি করে আঁচড়ানো চুল ক্লিনশেভড মুখ আমাকে যখন খুব কাছ থেকে দেখবেন আমার কেরানি পরিচয় আপনার কাছে আরো স্পষ্ট হবে মনে মনে তখন হয়তো গালিদিয়ে বলতে পারেনশালা বীমা কোম্পানির চুতিয়া দালাল আমি অবাক হব না	White shirt, black coat, black pants, black shoes, long tie, neatly combed hair,clean-shaved face. My clerk identity will be clearer from you when you show me up close. In your mind, you might say that, damn, you are a broker of Bima Company. I will not be surprised.
3	স্বল্পমেয়াদি এবং দীর্ঘমেয়াদি উভয় ক্ষেত্রে বিষণ্নতা উপসর্গ বিলোপের ক্ষেত্রে শারীরিক ব্যায়াম একটি কার্যকর পদ্ধতি।	Physical exercise is an effective way to alleviate depression, both short-term and long-term.
4	ইসলামই মানব জাতির জন্য সর্বোত্তম ধর্ম মুসলমানের জীবনে ইসলাম পূর্ণাঙ্গরূপে প্রবেশ করে বরং ইহা দৈনিক জীবনের সব কর্মকান্ডকে পরিচালনা করে আধুনিক বিশ্বে মানুষের সকল সমস্যা সমাধান ইসলাম ব্যতীত অন্য কোন ধর্ম নেই ইহা একমাত্র ইসলামেরই বৈশিষ্ট্য	Islam is the best religion for the human race. Islam enters the life of a Muslim in its entirety. Rather, it manages all the activities of daily life and solves all the problems of the people in the modern world. There is no religion other than Islam. This is the only feature of Islam.
5	জাবিতে বর্তমানে শিক্ষার্থীদের পরিবহনের জন্য ভাড়া ও নিজস্ব মিলিয়ে টি বাস আছে চারটি নতুন নিজস্ব বাস পরিবহন পুলে যুক্ত হলে বাসের সংখ্যা হবে টি এতে পরিবহন সংকটের অনেকটা সমাধান হবে বলে মনে করছেন অর্থ কমিটির সদস্যরা	At present, JU has 5 buses for the transportation of students. If 3 new buses are added to the transport pool, the number of buses will be 6. The members of the finance committee think that this will solve a lot of the transport crisis.
6	ডিজিটাল সেবা সাধারন মানুষের হাতের কাছে পৌঁছে দেওয়া উদ্যোক্তা রাসেল হাওলাদার জানান এসএসসি পরীক্ষা শেষে চাচা স্বপন হাওলাদারের সহযোগিতায় কম্পিউটারে হাতে খড়ি তার তারপর এগিযে চলা অল্প সময়ের মধ্যে কম্পিউটার চালানো ই মেইল করাসহ বেশ কিছু কাজ রপ্ত করেন	Russell Hawlader, an entrepreneur who has made digital services accessible to the common man, said that after the SSC exam, his uncle Swapan Hawlader handcuffed him to his computer. Then he went on to do a lot of work, including running a computer, e-mailing in a short time.

Figure 1. Data samples from BanglaLM.

Data Preprocessing

Text data is always noisy, and consists of many symbols, emoticons, URLs, invisible characters, and non-Bangla characters. Prior research has found that filtering and cleaning the data before developing a pre-trained language model is extremely advantageous and helpful in text classification. Hence, the dataset was preprocessed before developing or training an unsupervised FastText language model. The preprocessing steps include the removal of digits, emoticons, punctuation, URLs, HTML tags, and hashtag signs. Tokenization of the articles was performed using space as a delimiter. The whole dataset is held to a structure that contains a minimum word count of 3, and the maximum word count is 512. The word count in the paragraph is reasonably normally distributed. Table 1 shows how the datasets were organized and distributed in the study.

Table 1. The preprocessing steps and property of the data-set before fitting into pre-trained model V2.

Attribute	Action
Total data	19,925,396
Min sentence length	3
Max sentence length	512
Total words	821,007,301
Unique words	1,710,431
Total char length	5,367,701,734
Noise	No
Emoticon	No
URL tag	No
HTML tag	No
Punctuation	No
Stop words	Yes
Stemming	No
Lemmatization	No

4. Methodology

The high-level overview of the enhanced technique for word embedding using pre-trained models for Bangla language processing is shown in Figure 2. In order to perform this study, some essential steps for the training of the unsupervised dataset, generating a language model, and evaluating these models were considered. At first, data is preprocessed and then split it into training, test, and validation sets. For comparison purposes, Word2Vec, Facebook FastText, and BanglaFastText were considered. Using these vector representations, machine learning classical approaches were applied. For the evaluations purpose, an ML-based model as well as a deep neural network LSTM and CNN-based model were used in order to perform classification using the vector representation as a feature. In the preprocessing step, all punctuation (e.g., ',', '.', and '?') from the dataset were removed.

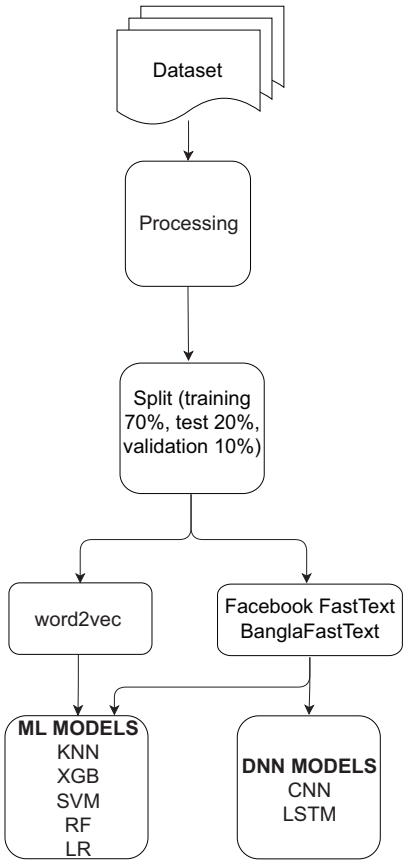

Figure 2. Different steps of the performance evaluation for the proposed word embedding technique.

4.1. Model Architecture

In order to train a FastText model, two popular methods, Skip-gram and cbow, were used. An explanation of the continuous Skip-gram model for generating FastText word embedding methods is given below: Let W be the length of the token vocabulary, and the index is the only identifier of every token w, where $w \in 1, \ldots, W$. The aim of training a model is to learn the vector representation of every token w. According to the distributional hypothesis, word embedding is trained in order to forecast a word that remains in its context by its index. Mathematically, for a large training data, which is a sequence of words $w_1, \ldots, w_T$, then, the objective function of the Skip-gram method with maximization is represented by the following equation:

$$\sum_{t=1}^{T} \sum_{c \in C_t} log p(w_c | w_t) \tag{1}$$

Here, the notation C_t represents the indices of tokens surrounding the word w_t. The probability of observing a context word w_c given wt will be parameterized using the aforementioned word vectors. Let s be a scoring function that maps the word and context to scores in $\mathbb{R}$.

The probability function using a softmax layer for a context word is as follows:

$$p(w_c | w_t) = \frac{e^{s(w_t, w_c)}}{\sum_{j=1}^{W} e^{s(w_t, j)}} \tag{2}$$

Here, all the context words are considered as positive examples, and negatives at random from the dictionary for the position of t. Considering a context index of c, according to the binary logistic loss, the negative log-likelihood can be obtained as follows:

$$log(1 + e^{-s(w_t, w_c)}) + \sum_{n \in N_{t,c}} log(1 + e^{s(w_t, n)}) \tag{3}$$

Here, $N_{t,c}$ represents a list of negative examples, which is sampled from the lexicons. Mathematically, the logistic loss function $\ell : x \mapsto log(1 + e^{-x})$ can be re-written as the objective:

$$\sum_{t=1}^{T} \left[\sum_{c \in C_t} \ell(s(w_t, w_c)) + \sum_{n \in N_{t,c}} \ell(-s(w_t, n)) \right] \tag{4}$$

As a natural parameterization, the scoring distribution s between a context word w_c and a word w_t is to utilize the word vectors. Let w be the word in the dictionary with two vectors u_w and v_w in $\mathbb{R}^d$. The two vectors are considered as input and output vectors in many works. Consider that the vectors are $\mathbf{u}_{wt}$ and $\mathbf{v}_{wc}$, whose corresponding words are w_t and w_c, respectively. Then, the score can be calculated as the scalar product between the word and context vectors represented by $s(w_t, w_c) = \mathbf{u}_{w_t}^{\top} \mathbf{v}_{w_c}$. The model illustrated in this section is the Skip-gram model with negative sampling, introduced by Mikolov et al. [6]

4.2. Subword Model

The Skip-gram model lacks the internal structure of words, since it uses a different vector representation for each word instead. As a result, in this section, an alternative scoring function s that incorporates this information is proposed.

An n-gram is a bag of characters that represents a word. Between words, the structurally diverse symbols < and > were added, which allow us to identify prefixes from suffixes. While practicing character n-grams, the word w was placed in the set of n-grams, so that it helps us to learn an additional representation for each word. Where $n = 3$, for example, will be represented by n-grams:

<be, ben, eng, nga, gal, al, i>

and the special sequence

<bengali>.

The n-grams larger than or equal to 3 and less than 6 are extracted in practice for n. There are many ways to go about this, such as taking into account all prefixes and suffixes in the n-grams.

Let us consider an n-grams-sized dictionary G. Given a word emphw, consider that the set of n-grams appearing in emphw by $G_w \subset 1, \ldots, G$. Each n-gram g is assigned a vector represented by $textbfz_g$. A word is represented by the vector sum of its n-gram vector representations. As a result, the scoring function is as follows:

$$s(w, c) = \sum_{g \in G_w} \mathbf{z}_g^\top \mathbf{v}_c \tag{5}$$

By using this simple approach, representations may be shared across words, which allows us to learn more reliable representations for rare terms that are difficult to learn.

A hashing function maps n-grams to integers in the range of 1 to K. This is achieved by utilizing the |FNV-1a|variant (http://www.isthe.com/chongo/tech/comp/fnv/ (accessed on 4 March 2022)) Fowler–Noll–Vo hashing function. In the example below, $K = 2.106$ is used. Words are represented by their index in the dictionary and their set of hashed n-grams.

4.3. Training Setup

In this study, the model is trained using the Cloud-based platform Google Colab Pro with Tensor Processing Units (TPUs), and Python version 3.7.11 was chosen as the computational language. Colab Pro has a disk space of approximately 107.72 GB and RAM 35 GB for TPU. In the training phase, this work used the preprocessed data of BanglaLM. It also analyzed the performance of each model individually by using different vector sizes, window sizes, and iterations. Here, the Gensim [27] version 4.01 was coded to create these pre-trained models and followed the FastText setups as building models.

4.4. Hyperparameters

Vector dimension, epochs, learning rate, and subword length are some of the four important basic hyperparameters that can be tuned in the FastText technique. The length of the vector size to represent a single word is known as the vector dimension. Larger vectors can carry more data, but they are more difficult to train, time consuming, and require more data. The number of times the model trains on a batch of data is measured in epochs. For larger corpora, the epoch should be shorter; otherwise, iterations will take longer. The learning rate is a metric that indicates how quickly the model should arrive at a solution. The length of substrings to consider for various processing tasks, such as resolving out of vocabulary terms, is specified by sub-word length. Increasing or decreasing the learning rate of the algorithm is another way of changing the learning speed of the enhanced model. This corresponds to how much the model changes after each example is processed. A learning rate of 0 indicates that the model does not change and thus does not learn. The learning rate should be in the range of 0.1 to 1.0. In this work, a learning rate of 0.1 was used.

In the training setup, applied various important parameters were applied based on previous experience, which are described in Table 2, and illustrated the hyperparameters used in this work to train the developed model. The training time for both models is described in Table 3.

Table 2. Optimized hyperparameters are chosen for the training of the FastText pre-trained model.

Embedding Hyperparameters	Value
Embedding dimension	300
Model	Skip-gram, CBOW
Window size	5
Min count	5
Loss function	Ns
Epochs	15
Max *n*-grams	6
Min *n*-grams	3
lr	0.1

Table 3. Training time of the BanglaFastText pre-trained models in Google Colab Pro (TPU).

Experiment	Training Time
FastText—Skip-gram	23.4 h
FastText—CBOW	23.2 h

5. Experiments

This section explains the evaluation of the proposed pre-trained Bangla FastText model using text classification datasets and then compares the performance with the existing Facebook Bangla FastText model including Word2Vec. The BanglaFastText model was evaluated using three common text classification Bangla datasets, where well-known machine learning classifiers such as SVM, random forest, XGBoost, KNN, and logistic regression were used as text classifiers. As deep learning models, deep LSTM and CNN for sequential tokens learning were used. In addition, the setup tool of the experiment in which all evaluations and comparisons were performed to show the enhancement of the proposed method were described, utilizing statistical processes, and analysis results. The evaluation statistical metrics are precision, recall, accuracy, F1-score, and Hamming loss. Since the dataset is imbalanced, the results of document classification using macro-averaged precision, recall, and F1-score were considered. The experimental steps to generate the result and statistical analysis are depicted in Figure 2

5.1. Experiment Setup

All the experiments in this study were conducted on a laptop computer with an Intel Core i7 CPU, 8 GB of DDR3 RAM, an NVIDIA 4 GB DDR5 GPU, and the Windows 10 Pro (64-bit) operating system. Python version 3.7 was used for all the experiments. This work used Pandas libraries as well as Numpy for reading the data and various types of mathematical operations. Seaborn and matplotlib were used for data visualization. Figure 3 shows how the words similar in meaning are mapped to embedding vectors that are close to each other in terms of proposed word embedding. For performance analysis and the development of machine learning classifiers, sklearn library was used in this study. To develop the model, in this paper FastText embedding was used. FastText embeddings are a modification of the Word2Vec embeddings. FastText considers a word consisting of /emphn-grams and breaks a given word into several sub-words [6] according to Word2Vec, which feeds the network with individual words. For training purposes, the Gensim library embedding and text categorization were considered; the Gensim module includes Facebook's AI FastText implementation. In this work both the Skip-gram and CBOW models were trained, and the window size was kept at 5 and the vector size at 300. The tuning of the parameters was performed and the results for both the Skip-gram and CBOW models were checked to show the performance enhancement.

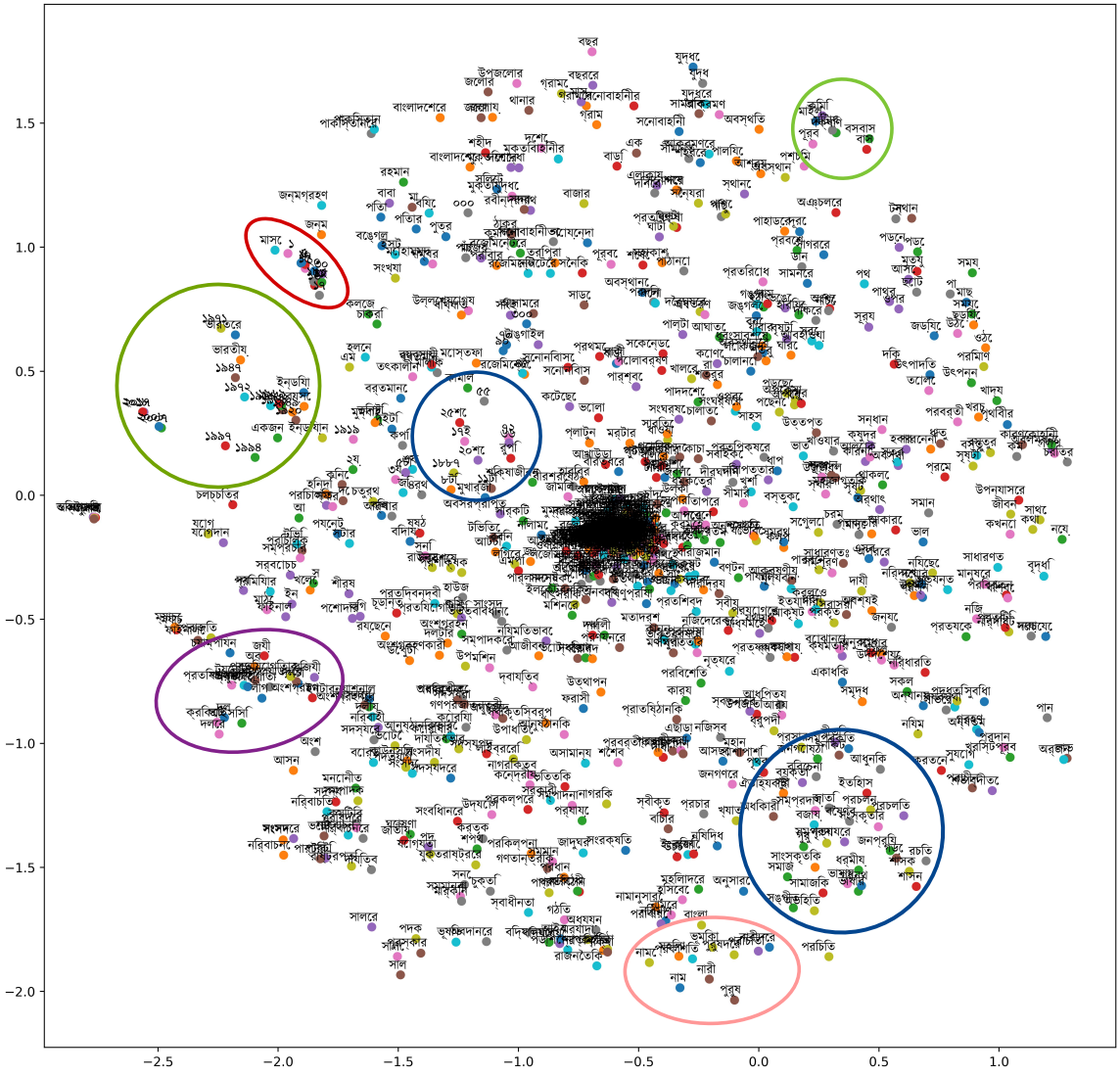

Figure 3. Sample data visualization.

5.2. Dataset for Analysis

In this section, the description of the datasets used for the experiments is explained.

- BanFakeNews [28] is a publicly available dataset (https://www.kaggle.com/cryptexcode/banfakenews (accessed on 4 March 2022)) for detecting fake news in Bangla. A benchmarking system with state-of-the-art NLP technology was developed to identify Bangla fake news. This dataset contains approximately 50k words of Bangla news data, which includes misleading, clickbait, and satire contexts, and organizes all of the news in the dataset into 12 categories, which are distributed into authentic and fake news in the dataset;
- The Bangla sentiment analysis classification benchmark dataset corpus (https://data.mendeley.com/datasets/p6zc7krs37/4 (accessed on 4 March 2022)), which is introduced by Sazzad et al. [29]. The corpus consists of 11,807 annotated reviews, where each review contains around 2300 Bengali words. This corpus is class imbalanced, comprising 3307 negative and 8500 positive reviews;
- A dataset for sentiment analysis on Bengali news comments [30], which is a dataset (https://www.kaggle.com/mobassir/bengali-news-comments-sentiment (accessed on 4 March 2022)) where every data point was annotated by three different individuals to achieve three different perspectives; the final tag was chosen based on the majority decision. Five sentiment labels were used to detect the true sentiments of the sentences,

which were taken from comments on different news stories. This dataset contains 13,802 data points in total.

5.3. Classical ML Model

For this study, the model that was used for both the classical and deep learning algorithms is discussed below.

For the classical models, the training was performed using KNN, XGB, support vector machines (SVM), random forest (RF), and LR as baselines using character n-grams and word uni-grams with Word2Vec weighting. The results were reported using FastText embedding methods. For SVM, the *'rbf'* was used [31] as the kernel with the hyperparameter C set to 1 and *'gamma=scale'*. For KNN, after hyperparameter tuning, it is found that *'k=5'* is the optimal value and the value of *'p=2'* was set for *'Minkowski'* [32] as a distance metric. For LR *'l2'*, a penalty was chosen and the C value was set to 1. For XGBoost, the open-source XGBoost library (https://github.com/dmlc/xgboost (accessed on 4 March 2022)) was used, and it was observed that *'estimator=200'* and *'max_depth=5'* provided us with enhanced performance in the implementation after hyperparameter tuning. Meanwhile, for RF, *'n estimators'* was set to 200 and *'max_depth=10'* was chosen for the classical approach.

5.4. Neural Network Models

Deep learning-based models were used for solving the text classification task. As deep learning algorithms, convolutional neural networks (CNN) and long short-term memory (LSTM) were used, which achieve results in different NLP tasks. FastText word representations on data that solely consisted of Bengali text were trained and then used for the classification and for other purposes.

5.4.1. Convolutional Neural Networks (CNNs)

CNN is popular for image classification applications [33], though it has also recently been used in text classification. CNN can extract the important information from a sequence of text, and then feed it into an FNN, which also works for text classification. Similar CNN architecture was used in [34], where the authors trained CNN on Bangla names for information extraction. In CNN, the embedding layer serves as the first layer, which is used to transfer the word vector representations of select words into the dataset under consideration to the model structure. One convolutional layer in parallel receives the output of the embedding layer, followed by a global max-pooling layer where the *'Relu'* activation function is used. Finally, two dense fully connected layers follow, in which the last layer is responsible for classification, where *'Relu'* and *'sigmoid'* activation functions have been used, respectively. The optimizer is set to *'adam'* and the loss function to *'binary_crossentropy'* for binary classification; however, the activation function was set to *'softmax'* and the loss function to *'categorical_crossentropy'* for multiclass classification.

5.4.2. Long Short-Term Memory (LSTM)

LSTM plays a vital role in text sequential text classification [35]. For LSTM, an embedding layer with weights was added first and then two stacked LSTM layers were considered. The first LSTM layer has the parameter return_sequences, which is set to True. When the return sequence is set to True, the output of the hidden state of each neuron is used as an input to the next LSTM layer. For two of the LSTM layers, one contains 100 neurons and the other has 64 neurons with activation function *'tanh'*, which performs best for multiclass classification. Finally, a dense layer was added with the activation function *'softmax'*. As with CNN, the optimizer was set to *'adam'* and the loss function to *'binary_ crossentropy* for binary classification, as well as *'softmax'* and *'categorical_crossentropy'* for multiclass classification.

All the models used Adam's [36] optimization algorithm for training, with the learning rate $\alpha = 1 \times 10^{-2}$ and the default value for $\beta 1 = 0.9$, $\beta 2 = 0.999$, $\epsilon = 1 \times 10^{-7}$.

The application of dropout led to better convergence and reduced the difference between the training and validation accuracy. Table 4 shows the architecture of the LSTM used in the experiments.

Table 4. LSTM architecture for the development of the classification model.

Layer (Type)	Output Shape	Value
embedding_1 (Embedding)	(None, 40, 300)	5,738,100
lstm_2 (LSTM)	(None, 40, 100)	160,400
lstm_3 (LSTM)	(None, 64)	42,240
dense_1 (Dense)	(None, 2)	130

5.5. Evaluation Metric

For each textual dataset, the proposed method was used for evaluation, using five evaluation metrics. The results are discussed in the order in which the evaluation metrics were applied. The performance of the test dataset is measured by precision (P), recall (R), accuracy (A), F1-score, and Hamming loss [37]. Equation (6)–(9) is used in the development stage to calculate precision, recall, accuracy, and F1-score.

Precision is used to calculate the percentage of positive patterns that are correctly predicted out of all positive patterns in a positive class, represented by:

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

Recall refers to the percentage of positive patterns that are correctly classified, which is given by:

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

In general, the accuracy metric computes the ratio of correct predictions to total instances evaluated, which is evaluated by:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

This metric is the harmonic mean of recall and precision values.

$$f1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{9}$$

where TP is the true-positive samples; TN is the true-negative samples; FP is the false-positive samples; and FN is the false-negative samples.

The Hamming loss is the ratio of the number of incorrect labels to the total number of labels. The lower the Hamming loss, the higher the method's performance. The Hamming loss is 0 for a perfect classifier. The Hamming loss can be estimated using Equation (10).

$$hamming\_loss = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{L} (x_i \Delta Y_i) \tag{10}$$

Here, Δ stands for the symmetric difference of sets and considers the dataset to be given as $x_i, Y_i, i = 1 \ldots N$ with L number of labels.

5.6. Comparison with Previous Techniques

In order to evaluate the enhancement and effectiveness of the proposed system, the comparison of the proposed system with existing techniques was carried out. A word embedding is a representation of learned text in which words with related meanings are represented similarly. Methods of word embedding from a corpus of text learn a real-valued vector representation for a predetermined fixed-size vocabulary. There are different types of word embedding techniques available, such as bag of words, tf-idf, Word2Vec, FastText, etc.

The proposed model is compared with Word2Vec and Facebook's FastText word embedding pre-trained model, as those word embedding techniques perform better than others. It is observed that the proposed BanglaFastText word embedding model outperforms the previous models. Tables 5–7 show the performance of the different methods and algorithms in terms of accuracy, Hamming loss, recall, precision, and $f1$-score.

The BanFakeNews dataset, XGB classifier, and BanFastText feature extractor provide the lowest Hamming loss of 0.001 and the highest $f1$-score of 75.17%, while the accuracy is 97.23%, which is practically the same as Facebook's FastText accuracy of 97.45%. In terms of CNN, the Skip-gram and CBOW models of BanFastText show 94.25% and 96.23% accuracy, respectively, with the highest of 62.92% $f1$-score in Skip-gram and the lowest of 0.17 Hamming loss in CBOW, whereas FastText gives 92.98% accuracy. On the other hand, LSTM gives the highest $f1$-score of 78.12% for the BanFastText (CBOW) model, but the accuracy and Hamming loss of FastText and BanFastText are similar.

For the Bangla sentiment analysis classification benchmark dataset corpus, the BanFastText feature extractor has the lowest Hamming loss of 0.05 in both CBOW and Skip-gram, while FastText has 0.06, and the highest $f1$-score of 89.78% in CBOW and 90.03% in Skip-gram, while the FastText $f1$-score is 88.56%. BanFastText has the highest accuracy of 94.41%, while FastText has 92.25%, which is the lowest compared with the BanFastText model. For the LSTM model, the accuracy of BanFastText (CBOW) outperforms Facebook's FastText model with accuracies of 92.21% and 90.21%, respectively, and the $f1$-scores are 91.01% and 88.56%, respectively, which clearly indicates the better performance of BanFastText over Facebook's embedding model FastText. The Hamming loss is also the lowest at 0.07 in the BanFastText model, but FastText has a loss of 0.08. The BanFastText (Skip-gram) feature extractor obtained 73.08% accuracy using a CNN network and showed the lowest loss of 0.21, while FastText achieved 71.38% accuracy with a 0.29 Hamming loss. The $f1$-score was 61.07% for BanFastText, whereas the $f1$-score of FastText was 59.27% indicating that BanFastText performs better than FastText.

Table 5. Performance comparison of different feature extractors in the text classification of the BanFakeNews dataset.

Feature Extraction Methods	Algorithm	Performance Score				
		Accuracy	Hamming Loss	Recall	Precision	$f1$-Score
Word2Vec	KNN	0.8827	0.2671	0.6792	0.7745	0.5821
	XGB	0.9454	0.1814	0.6789	0.8723	**0.6545**
	SVM	0.7677	0.2318	0.7865	0.8124	0.6728
	RF	0.7212	0.2436	0.5267	0.9013	0.5876
	LR	0.8012	0.2764	0.6014	0.7067	0.5312
FastText	KNN	0.9224	0.2242	0.7174	0.9112	0.6012
	XGB	0.9745	0.2214	0.6489	0.9124	**0.7322**
	SVM	0.7436	0.2518	0.8913	0.8618	0.6523
	RF	0.7345	0.2142	0.5398	0.9533	0.6055
	LR	0.8356	0.2551	0.6295	0.7245	0.5423
	LSTM	**0.9814**	**0.1801**	0.7121	0.8942	0.7735
	CNN	0.9298	0.1954	0.9012	0.7823	0.5945
BanFastText (CBOW)	KNN	0.9321	0.2507	0.7401	0.9421	0.6217
	XGB	0.9645	0.0274	0.6267	0.9024	**0.7245**
	SVM	0.7581	0.2318	0.9003	0.8918	0.6728
	RF	0.7157	0.2542	0.5073	0.9333	0.5855
	LR	0.8824	0.2031	0.6419	0.7418	0.5946
	LSTM	**0.9814**	**0.1801**	0.7121	0.8942	0.7812
	CNN	0.9425	0.1759	0.9412	0.8019	0.6121
BanFastText (Skip-gram)	KNN	0.9664	0.1131	0.6309	0.9060	0.6733
	XGB	0.9723	**0.1105**	0.6309	0.9298	0.7517
	SVM	0.7687	0.2112	0.7012	0.9666	0.7214
	RF	0.7245	0.2345	0.5623	0.8733	0.5564
	LR	0.8766	0.1233	0.7421	0.7235	0.5823
	LSTM	**0.9896**	0.1802	0.7462	0.8344	**0.7723**
	CNN	0.9623	0.1968	0.9504	0.8272	0.6292

Table 6. Performance comparison of different feature extractors in the text classification of Bangla sentiment analysis classification benchmark dataset corpus.

Feature Extraction Methods	Algorithm	Performance Score				
		Accuracy	Hamming Loss	Recall	Precision	*f*1-Score
Word2Vec	KNN	0.8103	0.1897	0.5166	0.7277	0.6042
	XGB	**0.8586**	**0.1414**	0.6178	0.8346	0.7100
	SVM	0.8395	0.1605	0.5649	0.8043	0.6637
	RF	0.8005	0.1994	0.2915	0.9897	0.4504
	LR	0.8395	0.1604	0.8036	0.6811	**0.7373**
FastText	KNN	0.8848	0.1151	0.6873	0.875	0.7699
	XGB	**0.9225**	**0.0774**	0.8383	0.8795	0.8584
	SVM	0.9209	0.0660	0.8777	0.8776	0.8856
	RF	0.8796	0.1113	0.6933	0.9310	0.7948
	LR	0.8608	0.1312	0.8701	0.7245	0.7906
	LSTM	0.9089	0.0808	0.8432	0.8942	**0.8901**
	CNN	0.7138	0.2891	0.5103	0.7349	0.5927
BanFastText (CBOW)	KNN	0.9001	0.0999	0.7401	0.8844	0.8059
	XGB	0.9292	0.0707	0.8625	0.8825	0.8724
	SVM	**0.9441**	**0.0508**	0.9003	0.9003	0.9003
	RF	0.9098	0.0901	0.7673	0.8959	0.8266
	LR	0.8793	0.1206	0.6419	0.8985	0.7488
	LSTM	0.9221	0.0752	0.8511	0.9012	**0.9101**
	CNN	0.7238	0.2091	0.5203	0.7049	0.6027
BanFastText (Skip-gram)	KNN	0.8927	0.1072	0.6937	0.8796	0.7831
	XGB	0.9229	0.0773	0.8429	0.8773	0.8697
	SVM	**0.9373**	**0.0526**	0.8836	0.8917	0.8978
	RF	0.8988	0.1003	0.6752	0.9033	0.8126
	LR	0.8823	0.1287	0.6042	0.9049	0.8246
	LSTM	0.9123	0.0701	0.8712	0.9101	**0.9134**
	CNN	0.7308	0.2104	0.5813	0.7191	0.6107

Table 7. Performance comparison of different feature extractors in the text classification of a dataset.

Feature Extraction Methods	Algorithm	Performance Score				
		Accuracy	Hamming Loss	Recall	Precision	*f*1-Score
Word2Vec	KNN	0.5677	0.4322	0.6620	0.5908	0.5908
	XGB	0.5952	0.4041	0.7969	0.5953	0.6815
	SVM	**0.6057**	**0.3942**	0.7288	0.6155	0.6674
	RF	0.5824	0.4175	0.9471	0.5693	**0.7111**
	LR	0.5989	0.4013	0.6369	0.6294	0.6326
FastText	KNN	0.6211	0.3788	0.6388	0.6753	0.6188
	XGB	0.6694	0.3314	0.7389	0.6798	0.6815
	SVM	0.6848	0.3151	0.8095	0.6747	**0.7262**
	RF	0.6621	0.3378	0.8297	0.6566	0.7231
	LR	0.6186	0.3813	0.3217	0.6825	0.6324
	LSTM	**0.7459**	**0.2132**	0.6822	0.7113	0.6946
	CNN	0.7326	0.2321	0.5423	0.7636	0.6225
BanFastText (CBOW)	KNN	0.6418	0.3581	0.6469	0.6532	0.6499
	XGB	0.6838	0.3032	0.7495	0.6815	0.7137
	SVM	0.7033	0.2966	0.7901	0.6966	**0.7491**
	RF	0.6803	0.3196	0.8215	0.6663	0.7361
	LR	0.6295	0.3709	0.6387	0.6646	0.6514
	LSTM	**0.7625**	**0.2124**	0.6667	0.7245	0.6734
	CNN	0.7411	0.2245	0.5612	0.7876	0.6467
BanFastText (Skip-gram)	KNN	0.6328	0.3672	0.6578	0.6534	0.6435
	XGB	0.6825	0.3191	0.7583	0.6729	0.7296
	SVM	0.7195	0.2879	0.8035	0.9829	**0.7427**
	RF	0.6712	0.3245	0.8198	0.6721	0.7382
	LR	0.6287	0.3711	0.6455	0.6712	0.6629
	LSTM	**0.7505**	**0.2142**	0.6845	0.7269	0.6921
	CNN	0.7327	0.2421	0.5245	0.7427	0.6456

For the Bengali News Comment dataset, the accuracy of BanFastText and FastText is 71.95% and 68.48%, and the $f1$-score is 74.27% and 72.62%, respectively. However, the Hamming loss for BanFastText is 0.28, which is also low. For the LSTM model, the accuracy of BanFastText (CBOW) outperforms Facebook's FastText model in terms of accuracy, but both techniques produce a nearly identical $f1$-score and Hamming loss, where the accuracies are 76.25% and 74.59%, and the $f1$-scores are 69.21% and 69.46%, respectively, and the Hamming loss is 0.21 for both the BanFastText and FastText models. For CNN, BanFastText outperforms FastText both in accuracy and $f1$-score, where the accuracy is 74.11% for BanFastText and 73.26% for FastText, the $f1$-score is 64.67% in BanFastText and 62.25% for FastText, and the lowest Hamming loss of 0.22 is found for BanFastText.

5.7. Comparison with Datasets of Previous Work

The experiments for this purpose were carried out with three datasets. In the BanFakeNews dataset, Word2Vec pre-trained word embedding techniques were used, and for classification, some classical approaches such as SVM, RF, and LR were used, achieving 46%, 55%, and 53% $f1$-scores, respectively. For neural networks, 53% and 59% $f1$-scores of the fake class were achieved for CNN and LSTM, respectively. However, in classical approaches such as SVM, RF, and LR, the proposed word embedding outperforms by a large margin. This work achieved 64% and 66% $f1$-scores for CNN and LSTM, respectively, which is better than that found in previous work.

For the Bangla sentiment analysis classification benchmark corpus dataset, 93%, 89%, and 91% accuracy was achieved for SVM, RF, and LR, respectively, where bag of words and $tf\text{-}idf$ word embedding techniques were used. Here, Bangla FastText word embedding performs slightly better than the other methods, as shown in the table.

For the Bengali News Comment dataset, the proposed word embedding method outperforms other methods. Comparisons between the proposed method and other techniques in terms of accuracy and $f1$-score are shown in Table 8.

Table 8. Comparison of BanglaFastText's performance with previous experimental datasets.

Dataset	Algorithm	Performance Score	
		Existing	Proposed
BanFakeNews	SVM ($f1$-score)	0.46	0.57
	RF ($f1$-score)	0.55	0.59
	LR ($f1$-score)	0.53	0.60
	LSTM ($f1$-score)	0.53	0.66
	CNN ($f1$-score)	0.59	0.64
Bangla sentiment analysis classification benchmark dataset corpus	SVM(acc)	0.93	0.94
	RF(acc)	0.89	0.91
	LR(acc)	0.91	0.88
Bengali News Comment	SVM(acc)	0.61	0.76
	LSTM(acc)	0.74	0.75
	CNN(acc)	0.61	0.72

6. Conclusions

In this paper, two Bangla FastText word embedding pre-trained models were presented, with a toolbox trained on a huge Bangla corpus (around 20 million points of textual data), including organized and non-organized datasets. In order to evaluate the proposed word embedding performance, three text classification datasets were used and extracted the embedding, and then trained the method with the help of common machine learning classifiers. Next, the performance of the proposed method was compared with the Facebook FastText and Word2Vec, where the proposed models showed enhanced performance for all datasets. The results show that without preprocessing such as lemmatization, stemming, and other techniques, significant performance improvements can be achieved with BanglaFastText word embedding. BanglaFastText also exceeds Facebook's FastText by a wide margin in text classification and word similarity.

This work was carried out by utilizing the following neural word embedding FastText techniques, so it does not consider word position, context, and sensitivity. It always provides similar word vectors for a word in any position. For example, the word 'Python', describing either the programming language or the animal, would be the same vector. This is a limitation of this work.

Future research direction can be the development of the advanced models using the BanglaLM dataset, such as BERT, ELMO, and GTP to overcome the limitations of this work. In addition, a Python-based Bangla toolkit can be developed, which will make it easier to develop linguistic applications and research. Design APIs to access data in multiple applications can also be a good future consideration.

Author Contributions: Conceptualization, M.K., M.S.I.S., M.F.S., N.J.P. and M.S.A.; investigation, M.K., M.S.I.S., M.S.A., P.K.D. and T.K.; methodology, M.K., M.S.I.S., M.F.S., N.J.P., M.S.A., P.K.D. and T.K.; software, M.K., M.S.I.S. and M.F.S.; validation, M.K., M.S.I.S., M.F.S., N.J.P., M.S.A., P.K.D. and T.K.; writing—original draft preparation, M.K., M.S.I.S., M.F.S. and N.J.P.; writing—review and editing, M.K., M.S.A., P.K.D. and T.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2013**, arXiv:1301.3781.
2. Baeza-Yates, R.; Ribeiro-Neto, B. *Modern Information Retrieval*; ACM Press: New York, NY, USA, 1999; Volume 463.
3. Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; Kuksa, P. Natural Language Processing (almost) from Scratch. *arXiv* **2011**, arXiv:1103.0398.
4. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. *arXiv* **2017**, arXiv:1706.03762.
5. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv* **2019**, arXiv:1810.04805.
6. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; Dean, J. Distributed Representations of Words and Phrases and Their Compositionality. *arXiv* **2013**, arXiv:1310.4546.
7. Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T. Enriching Word Vectors with Subword Information. *arXiv* **2017**, arXiv:1607.04606.
8. Hearst, M.; Dumais, S.; Osuna, E.; Platt, J.; Scholkopf, B. Support vector machines. *IEEE Intell. Syst. Their Appl.* **1998**, *13*, 18–28. [CrossRef]
9. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
10. Cover, T.M.; Hart, P.E. Nearest Neighbor Pattern Classification. *IEEE Trans. Inf. Theory* **1967**, *13*, 21–27. [CrossRef]
11. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining—KDD'16, San Francisco, CA, USA, 13–17 August 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 785–794. [CrossRef]
12. Hosmer, D.; Lemeshow, S. *Applied Logistic Regression*; Wiley-Interscience: Hoboken, NJ, USA, 2000; Volume 354. [CrossRef]
13. Yamashita, R.; Nishio, M.; Do, R.K.G.; Togashi, K. Convolutional neural networks: an overview and application in radiology. *Insights Into Imaging* **2018**, *9*, 611–629. [CrossRef] [PubMed]
14. Hochreiter, S.; Schmidhuber, J. Long Short-term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
15. Pennington, J.; Socher, R.; Manning, C. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; Association for Computational Linguistics: Doha, Qatar, 2014; pp. 1532–1543. [CrossRef]
16. Das, A.; Ganguly, D.; Garain, U. Named Entity Recognition with Word Embeddings and Wikipedia Categories for a Low-Resource Language. *ACM Trans. Asian -Low-Resour. Lang. Inf. Process.* **2017**, *16*, 18:1–18:19. [CrossRef]
17. Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; Dyer, C. Neural Architectures for Named Entity Recognition. *arXiv* **2016**, arXiv:1603.01360.
18. Tang, D.; Wei, F.; Yang, N.; Zhou, M.; Liu, T.; Qin, B. Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Baltimore, MD, USA, 22–27 June 2014; pp. 1555–1565. [CrossRef]
19. Chakrabarty, A.; Garain, U. BenLem (A Bengali lemmatizer) and its role in WSD. *ACM Trans. Asian -Low-Resour. Lang. Inf. Process.* **2016**, *15*, 1–18. [CrossRef]

20. Sultana Ritu, Z.; Nowshin, N.; Mahadi Hasan Nahid, M.; Ismail, S. Performance Analysis of Different Word Embedding Models on Bangla Language. In Proceedings of the 2018 International Conference on Bangla Speech and Language Processing (ICBSLP), Sylhet, Bangladesh, 21–22 September 2018; pp. 1–5. [CrossRef]
21. Al-Amin, M.; Islam, M.S.; Das Uzzal, S. Sentiment analysis of Bengali comments with Word2Vec and sentiment information of words. In Proceedings of the 2017 International Conference on Electrical, Computer and Communication Engineering (ECCE), Cox's Bazar, Bangladesh, 16–18 February 2017; pp. 186–190. [CrossRef]
22. Ahmad, A.; Amin, M.R. Bengali word embeddings and it's application in solving document classification problem. In Proceedings of the 2016 19th International Conference on Computer and Information Technology (ICCIT), Dhaka, Bangladesh, 18–20 December 2016; pp. 425–430. [CrossRef]
23. Kunchukuttan, A.; Kakwani, D.; Golla, S.; Gokul, N.C.; Bhattacharyya, A.; Khapra, M.M.; Kumar, P. AI4Bharat-IndicNLP Corpus: Monolingual Corpora and Word Embeddings for Indic Languages. *arXiv* **2020**, arXiv:2005.00085.
24. Rafat, A.A.A.; Salehin, M.; Khan, F.R.; Hossain, S.A.; Abujar, S. Vector Representation of Bengali Word Using Various Word Embedding Model. In Proceedings of the 2019 8th International Conference System Modeling and Advancement in Research Trends (SMART), Moradabad, India, 22–23 November 2019; pp. 27–30. [CrossRef]
25. Ahmed Chowdhury, H.; Haque Imon, M.A.; Islam, M.S. A Comparative Analysis of Word Embedding Representations in Authorship Attribution of Bengali Literature. In Proceedings of the 2018 21st International Conference of Computer and Information Technology (ICCIT), Dhaka, Bangladesh, 21–23 December 2018; pp. 1–6. [CrossRef]
26. Chowdhury, H.A.; Imon, M.A.H.; Islam, M.S. Authorship Attribution in Bengali Literature Using fastText's Hierarchical Classifier. In Proceedings of the 2018 4th International Conference on Electrical Engineering and Information Communication Technology (iCEEiCT), Dhaka, Bangladesh, 13–15 September 2018; pp. 102–106. [CrossRef]
27. Rehurek, R.; Sojka, P. Software Framework for Topic Modelling with Large Corpora. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, Valletta, Malta, 17 May 2010; pp. 45–50. [CrossRef]
28. Hossain, M.Z.; Rahman, M.A.; Islam, M.S.; Kar, S. BanFakeNews: A Dataset for Detecting Fake News in Bangla. In Proceedings of the 12th Language Resources and Evaluation Conference, Marseille, France, 11–16 May 2020; pp. 2862–2871.
29. Sazzed, S. Cross-lingual sentiment classification in low-resource Bengali language. In Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020), Online, 19 November 2020; pp. 50–60. [CrossRef]
30. Ashik, M.A.U.Z.; Shovon, S.; Haque, S. Data Set For Sentiment Analysis On Bengali News Comments And Its Baseline Evaluation. In Proceedings of the 2019 International Conference on Bangla Speech and Language Processing (ICBSLP), Sylhet, Bangladesh, 27–28 September 2019; pp. 1–5. [CrossRef]
31. Han, S.; Qubo, C.; Meng, H. Parameter Selection in SVM with RBF Kernel Function. In Proceedings of the World Automation Congress Proceedings, Puerto Vallarta, Mexico, 24–28 June 2012.
32. Merigó, J.M.; Casanovas, M. A New Minkowski Distance Based on Induced Aggregation Operators. *Int. J. Comput. Intell. Syst.* **2011**, *4*, 123–133. [CrossRef]
33. Kowsher, M.; Alam, M.A.; Uddin, M.J.; Ahmed, F.; Ullah, M.W.; Islam, M.R. Detecting Third Umpire Decisions & Automated Scoring System of Cricket. In Proceedings of the 2019 International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering (IC4ME2), Rajshahi, Bangladesh, 11–12 July 2019; pp. 1–8.
34. Kowsher, M.; Sanjid, M.Z.I.; Das, A.; Ahmed, M.; Sarker, M.M.H. Machine learning and deep learning based information extraction from Bangla names. *Procedia Comput. Sci.* **2020**, *178*, 224–233. [CrossRef]
35. Kowsher, M.; Tahabilder, A.; Sanjid, M.Z.I.; Prottasha, N.J.; Uddin, M.S.; Hossain, M.A.; Jilani, M.A.K. LSTM-ANN & BiLSTM-ANN: Hybrid deep learning models for enhanced classification accuracy. *Procedia Comput. Sci.* **2021**, *193*, 131–140.
36. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2017**, arXiv:1412.6980.
37. Elisseeff, A.; Weston, J. A kernel method for multi-labelled classification. In Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic—NIPS'01, Vancouver, BC, Canada, 3–8 December 2001; MIT Press: Cambridge, MA, USA; pp. 681–687.