International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 12– No.30, June 2020 – www.ijais.org

# A Comparative Study to Detect Emotions from Tweets Analyzing Machine Learning and Deep Learning Techniques

Sanzana Karim Lora
Ahsanullah University of Science and Technology
Dhaka, Bangladesh

Nazmus Sakib
Ahsanullah University of Science and Technology
Dhaka, Bangladesh

Shahana Alam Antora
Ahsanullah University of Science and Technology
Dhaka, Bangladesh

Nusrat Jahan
Ahsanullah University of Science and Technology
Dhaka, Bangladesh

## ABSTRACT
At the present time, people corresponding to share their thoughts, emotions and ideas through social media. Twitter is one of the most widespread sites among them. The goal of this study is to classify positive and negative emotions accurately by with a dataset of Twitter tweets. For this work, machine learning (Naïve Bayes, SVM, Logistic Regression using tf-idf and count vectors), deep learning(Stacked Long short-term memory (LSTM), Stacked LSTM with 1D convolution, CNN with pre-trained word embedding) and a BERT based model have been used. The CNN model with pre-trained word embedding has performed the best among all models with 84.1% accuracy after 3 epochs only. Among all machine learning and deep learning models, deep learning models have performed better than machine learning models.

## General Terms
Machine learning, deep learning, natural language processing, Twitter, classification, BERT, LSTM, CNN, tf-idf, count vector, binary classification.

## 1. INTRODUCTION
Feeling which is so called emotion is one kind of communication medium in our day to day life. Emotion can be articulated by the physical expression, facial expression, writing, speaking, behavior, etc. In our technology-based era, having a social media account is very common. In today’s world, people desire to express their emotions and thoughts from side to side different social media like Facebook, Twitter, Instagram and so on. Here millions of people in their everyday lives share their views, opinions and also their emotions or a particular thing through social media [1]. This huge amount of statistics containing emotions creates an opportunity for researchers for their work. So, detecting emotions by analyzing social media’s text is a big challenge for researchers. Among all social media, Twitter has become a popular and interesting research area because of its short text.

Twitter is a 'microblogging' structure that allows users to send and receive short posts called tweets. Tweets can be up to 140 characters long and can include links to relevant websites and resources [2]. The goal of emotion detection from text is to categorize the actual emotion polarity. It can be taken as a text classification problem. The dataset which we have used in this work contains “positive” and “negative” polarity of user’s tweets. So, it is so far a binary classification problem.

Our target work objects to mature and assess different models that classify as Twitter tweets “positive” or “negative” by using Natural Language Processing, Machine Learning techniques and Deep Learning.

The contents of the paper are consisting as follow: section 1 is the introductions of the work, section 2 introduces the related work or research on this field, the proposal of the research will be introduced at section 3, the models will be explained in section 4, the results and the analysis of them will be shown in section 5 and the conclusion and the future work will be on section 6 and 7.

## 2. LITERATURE REVIEW
Different researches use natural language processing, machine learning techniques, and neural network architectures to build, tune, and evaluate models that classify Reddit text comments as "depressed" or "non-depressed. The custom dataset for this work has been created by scraping two subedits comments that have been extracted by PRAW. Machine learning (logistic regression, support vector machines), a BERT-based model, and neural networks with and without word embedding (CNN) for this classification task have been used. Two models logistic regression and SVM to serve as a baseline for this project and developed BERT based model and a single layer character-level convolutional neural network (CNN) created using GloVe and fastText. This model contains 6 convolutional layers utilizing rectified linear units (ReLU) activation followed by 3 fully-connected layers [3].

The result showed that the CNN model without word embedding performed the best, followed by the BERT-based model and CNN model with embedding performed worst out of all models.

Identify positive and negative tweets experimented with very-deep convolutional neural networks (VDCNNs) and Google’s BERT architecture for classifying tweets as positive or negative have experimented. This work used the Sentiment140 data set and achieved an F1 score of 0.853 on the included test set. The models that have been build were trained using pre-trained 100dimensional word-embedding from the GloVe word embedding dataset except for the BERT model. For optimization Adam optimizer has been used with loss calculated via Binary Cross-Entropy Loss and sigmoid function have been used for classification task [4]. Some research proposed a model to A CNN sentiment analyzer that has been built as a base model. Besides these models, a shallower network containing two of the aforementioned convolutional blocks, and a deeper architecture using four convolutional blocks have been built.

They found that adding more layers to the model had not improved performance so they used a single layer and the most reliable model for classifying tweets has been the two Convolutional Block model utilizing the GloVe Twitter 27B 100d pre-trained word embedding’s, and early stopping for regularization.

There are three models: an aspect classification model, a sentiment polarity classification model, and a combined model for both aspect and sentiment classification. This paper showed the potential of using the contextual word representations from the pre-trained language model BERT, together with a fine-tuning method with additional generated text, to solve out-of-domain ABSA and outperform previous state-of-the-art results on SemEval2015 and SemEval2016 (task 5). The train data consisted of 'related' and 'unrelated' labels [5]. The combined model performed consistently better than the sentiment model in all domains. But the combined model performs worse than the aspect model in classifying relevant aspects.

A classifying model to classify positive, negative, or neutral sentiments in tweets of twitter data has been proposed in this paper [6]. Tweets have been used as the raw data, collected through Twitter API using a secret token, and has been preprocessed using text mining and applied the Dictionary-Based Approach. This paper focused on Document-level Sentiment Analysis. In the dictionary-based method, many seed emotional words used to build a dictionary structure using equivalent and opposite words, and the lately identified words are added to the seed list. In their experiment within 2500 English tweets in the dictionary-based approach, negative words scored 2195 and positive words scored 4096. Polarity mining has been done using the Dictionary-Based Approach while Sentimental Analysis has been performed using CNN. CNN model used for training purposes and compared to the existing system i.e., Dictionary –Based Approach and CNN the hybrid approach produced a better result. To estimate accuracy K-fold cross-validation has been used.

The impact of simple text preprocessing decisions on the performance of a standard word-based neural text classifier has been analyzed [7]. While comparing different systems, the paper also highlighted the importance to be consistent and the choice of how to preprocess the data. In general, except for domain-specific datasets in which sole tokenization performs poorly, simple tokenization works equally or better than more complex pre-processing processing techniques such as lemmatization or multiword grouping. As well when applied to simple tokenized datasets, word embedding trained on multiword-grouped corpora perform surprisingly. This research preprocessing choice which is ±2.4% on average for the best performing model.

A survey that focused on various deep learning methods used in different applications of sentiment analysis at the sentence level and aspect/target level. The sentiment analysis has been discussed based upon the textual target, like document, sentence, or aspect. It has been found that aspect level sentiment analysis helps to gain more insight into the different aspects/targets of the product or service or tweets. Sentence modeling uses a bag-of-words (BoW) approach that omits the information on semantics which could be overcome by Ngram but it suffers from data sparsity. Some common machine learning algorithms such as Support Vector Machine (SVM), Naive Bayes, random forest, and decision tree has been used in NLP. But deep learning has been more popular as it could learn by itself without training. Among many models of deep learning that use word embedding as input CNN and LSTM are most commonly used. The pre-trained word vectors like word2vec, GloVe, and fastText are used for the embedding of the input text. CNN which has been widely used in image processing could not handle sequential data but its' recent approaches employ CNN in NLP [8].

A combination of neural network architecture that employs Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) on top of pre-trained word vectors. ConvLstm exploits LSTM as a substitute for the pooling layer on CNN. Also using LSTM as an alternative for the pooling layers in CNN gives the model enhancement to capture longterm dependencies. The Stanford Large Movie Review Dataset has been used and divided into 50:50 training and testing sets where full training examples have been randomly split into training and validation. For training stochastic gradient descent has been applied by minimizing the negative log-likelihood or cross-entropy loss. To prevent overfitting early stopping strategy has been utilized while the gradient of the cost function has been computed with backpropagation through time (BPTT) [9].

To implement BoW and SkIPgram architectures for computing vector representation of word unsupervised learning of word-level embedding using the word2vec, which have been trained on lOO billion words of Google News, word2vec tool before training. As multi-layer has not improved accuracy or performance a single LSTM layer has been used.

A model using a deep neural network to classify the polarity of the sentences has been proposed [10]. The dataset has been taken from the Twitter API of Twitter tweets. Initially, the dataset has been pre-processed and tokenized. Then it has been fed to the feed-forward network, followed by three fully connected hidden layers, ReLU, and a sigmoid activation function. A Multi-Layer Perceptron MLP has been taken as a baseline model. The result showed that the accuracy of the train and test dataset in the deep neural network above 75% which had been much better than MLP accuracy.

A classifying model using deep CNN applied on character-level embeddings to increase the information for word embeddings and bidirectional LSTM to classify the sentences based on sentiment [11]. The model used three twitter sentiment classification datasets. A module has been designed called tweet processor to remove the non-essential words from the tweet retaining necessary information by doing preprocessing work. Later the output is fed to DeepCNN and the tweet processor applies a set of semantic rules SR on the sentences. The output of the DeepCNN is a fixed size feature vector for all the words along with word embeddings obtained from pre-trained word vectors like word2vec or GloVE are concatenated and fed to bidirectional LSTM and tested on the above-mentioned datasets. From results, it has been found that Semantic rules are effective to increase classification accuracy. This paper also conducts two experiments on DeepCNN and Bi-LSTM separately and found that the combined model gives a better result. The model using TwitterGlove outperforms the model using GoogleW2V as TwitterGlove captures more information on Twitter than GoogleW2V.

## 3. PROPOSAL

After analyzing different research papers described in section 2, it is clear that deep learning-based text classification using NLP is a popular topic nowadays. Identifying positive and negative emotions from social media is a very popular and challenging topic for research. As many people share their emotions through social media, it can be easy to collect huge data for research. In this paper, we have proposed some models to classify emotions from tweets of Twitter users. For this purpose, data preprocessing is an essential step after collecting the dataset. We have proposed both baseline models (traditional machine learning models) and deep learning models (Stacked LSTM, Stacked LSTM with 1D convolution, CNN with pre-trained word embedding, a BERT based model).

For baseline models, tf-idf(term frequency–inverse document frequency) and count vectors features have been used for Multinomial Naive Bayes, Support Vector Machine(SVM) and Logistic Regression separately as input. tf-idf is a weighting scheme that assigns each term in a document a weight based on its term frequency (tf) and inverse document frequency (idf). The terms with higher weight scores are considered to be more important [13].

CountVectorizer works on Terms Frequency, i.e. counting the occurrences of tokens and building a sparse matrix of documents x tokens [14].

For all deep learning models, epoch size has remained the same and it is 3 because of the large size of the dataset. But batch size, dropout probability, activation and optimization has been varied from model to model.

The main target of this proposed method is to obtain an improved accuracy to identify the emotions. We have trained all the models mentioned in fig 1 and found the accuracy. At last, the best-performed model has been got among them after comparing the accuracy results. The details of these models have been explained in section 4.

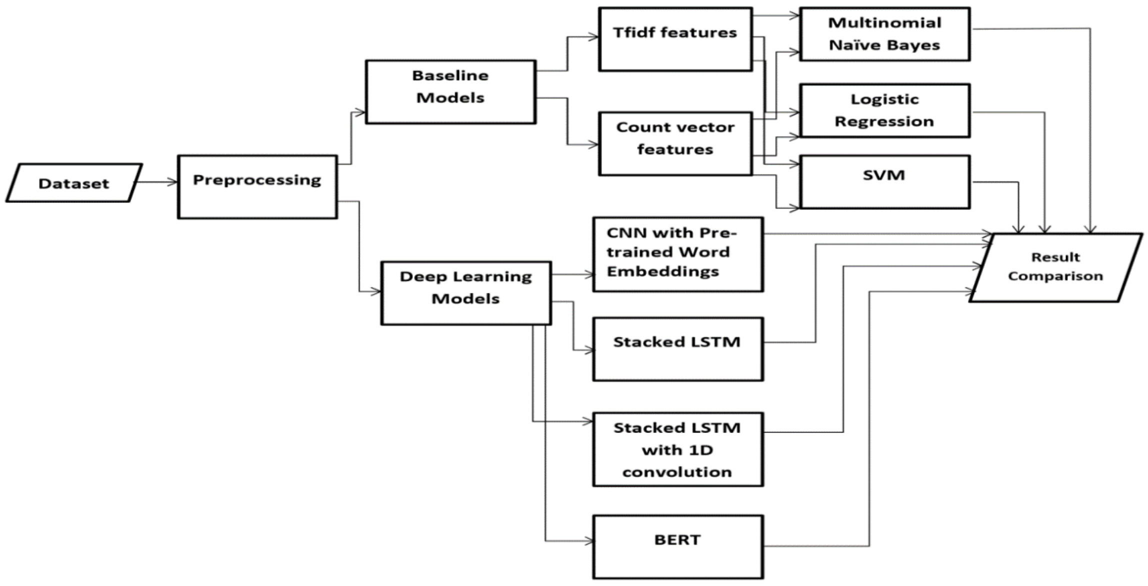

**Fig 1: Proposed Methodology**

## 4. IMPLEMENTATION

### 4.1 System Specification

Intel R core i5-7200U CPU has been used for this work. The machine has 4GB RAM and Windows operating system. GPU 0(Graphics Processing Unit) is used here. The language which is used for the target work is Python. So, Google colaboratory has been used for developing all the models described in this paper in later sections.

### 4.2 Dataset

For the dataset, sentiment140[12] dataset has been taken. The dataset contains 1048576 rows with six columns. But only two columns have been used for this work. They are "label" (the polarity of the tweet) and "tweet" (the text of the tweet). Here zero means negative emotion tweets and 4 means positive emotion tweets. The visualization of the initial number of tweets has been shown in fig 2 where units are in percentage.

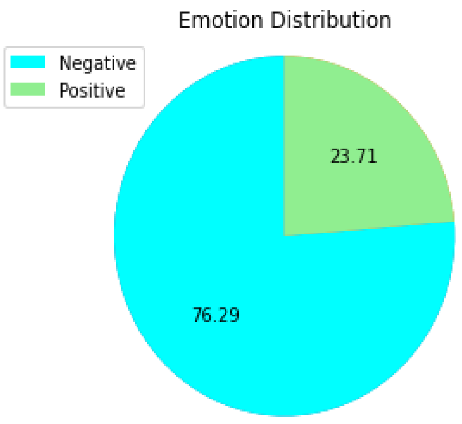

**Fig 2: Initial Emotion Distribution of Dataset**

### 4.3 Dataset Preprocessing

Data preprocessing is an essential phase after successfully collecting the dataset. In the real world, there can be some unnecessary, missing and noisy data for their huge size. By preprocessing, it is possible to remove this unnecessary data to reduce the size of huge data. Besides, preprocessing can reduce computation complexity. So, it needs to be preprocessed to remove noisy data and reduce complexity.

*International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868*
*Foundation of Computer Science FCS, New York, USA*
*Volume 12– No.30, June 2020 – www.ijais.org*

Data preprocessing which has been done involved the following steps:

- Converting uppercase letters to lowercase
- Removing punctuations
- Removing hashtags, URL, retweet mentions and user mentions
- Replacing multiple spaces with one space
- Tokenizing words
- Stemming of words
- Removing stopwords

## 4.4 Baseline Models

Three models have been created for this work. Both tf-idf and count vectors have been used for each model. Tweets were converted to a tf-idf features matrix to serve as inputs for the models. This process is done by TfidfVectorizer of scikit-learn with parameters max_features=1000, analyzer='word',ngram_range=(1,3).

For count vectors, tweets were converted to count vector features for the input of these models. This process is done by CountVectorizer of scikit-learn's with parameters analyzer= 'word'.

First, we have developed a Multinomial Naive Bayes Model. For this model, Python's scikit-learn's MultinomialNB has been used. Then, we have created a Support Vector Machine(Linear SVM) model. For this model, Python's scikit-learn's SGDClassifier has been used using alpha=0.001, random_state=5, max_iter=15. Next, we have built a Logistic Regression Model. For this model, we have used Python's scikit-learn's LogisticRegression with the inverse of regularization strength set to C=1.

For these cases, 90 % of data for training and 10 % for testing have been used.

## 4.5 Deep Learning Models

### *4.5.1 Convolutional Neural Network (CNN) with Pre-trained Word Embeddings*

The idea of using a CNN to classify text was first presented in the paper[15]. CNN is familiar with NLP tasks. As it is a binary classification problem, the model needs to pass a two-dimensional output vector. For that, two one hot encoded columns have been added to the data frame.

Pre-trained word embeddings, that use distributed word representations, are useful in text classification problems in representing interactions between words. We have used the Google News Word2Vec model in our work. It includes word vectors for a vocabulary of 3 million words and phrases that they trained on roughly 100 billion words from a Google News dataset. The vector length is 300 features[16].

Five different filter sizes are applied to each tweet, and GlobalMaxPooling1D layers are applied to each layer. Then, all the outputs are concatenated. A Dropout layer then Dense then Dropout and then Final Dense layer is applied. Rectified linear units (ReLU) activation has been used for this process.

The training and validation accuracy and training and validation loss during implementation have been shown in fig 3.

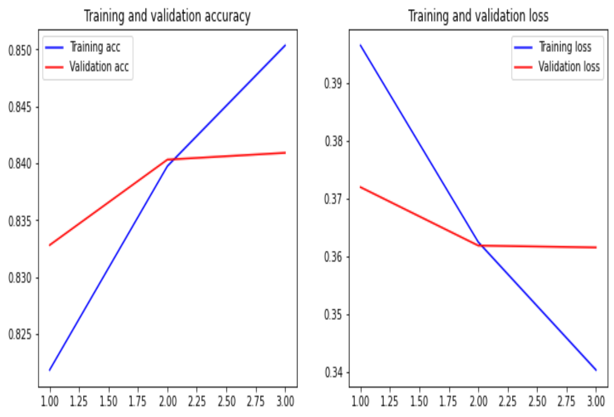

**Fig 3: Training and validation accuracy and training and validation loss for CNN with pre-trained word embeddings**

From fig 3, it is clear that when the training accuracy has increased, validation accuracy has also increased. This means that the model is fitting the training set better, and also has given better performance to predict new data.

Besides, validation loss is slightly greater than training loss. So, its performance can be better. In this model, we have used a batch size of 34, a dropout probability of 0.1, and Adam optimization. The model was trained for 3 epochs.

The whole dataset has been split as 70% for training, 10% for validation and 20% for testing for this model.

### *4.5.2 Stacked LSTM Model*

The original LSTM model is comprised of a single hidden LSTM layer followed by a standard feedforward output layer.

The Stacked LSTM is an extension to this model that has multiple hidden LSTM layers where each layer contains multiple memory cells[17].

An overview of the architecture of the model is shown in the below diagram.

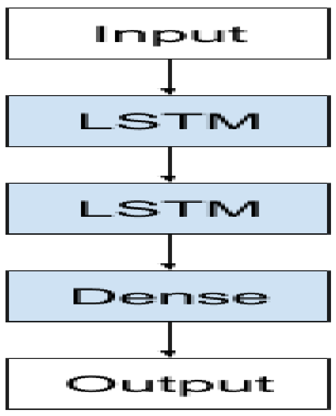

**Fig 4: Stacked Long Short-Term Memory Architecture [17]**

The model starts with an embedding layer. The layer lets the system expand each token to a more massive vector, allowing the network to represent a word in a meaningful way. The layer takes 10000 as the vocabulary size, 1000 as the dimension of the embeddings and the input_length of 100, which is the length of each tweet sequence.

The training and validation accuracy and training and validation loss during implementation have been shown in fig 5.

For this model, we have used a dropout probability of 0.2, Adam optimization and 3 epochs. Tanh activation has been used for this process. The whole dataset has been split as 70% for training, 10% for validation and 20% for testing for this model.

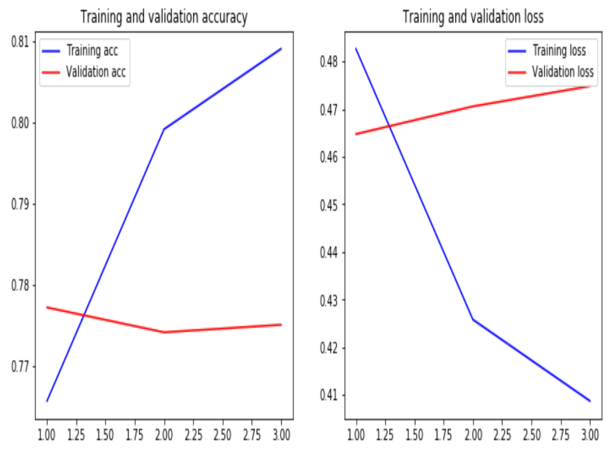

**Fig 5: Training and validation accuracy and training and validation loss for Stacked LSTM**

From fig 5, it is shown that when the training accuracy has increased, validation accuracy has decreased at epoch 2 then again increased which is lower than epoch 1. This means that the model is fitting the training set better, but is losing its ability to predict new data slightly. Besides, validation loss is greater than training loss. So, there can be overfitting.

### *4.5.3 Stacked LSTM with 1D Convolution*

An extra 1D convolutional layer has been added on top of the previously described model to reduce the training time. For this model, we have used a dropout probability of 0.3, kernel size of 5, pool size 4, nadam optimization and 3 epochs. Tanh activation has been used for this process.

The training and validation accuracy and training and validation loss during implementation have been shown in fig 6.

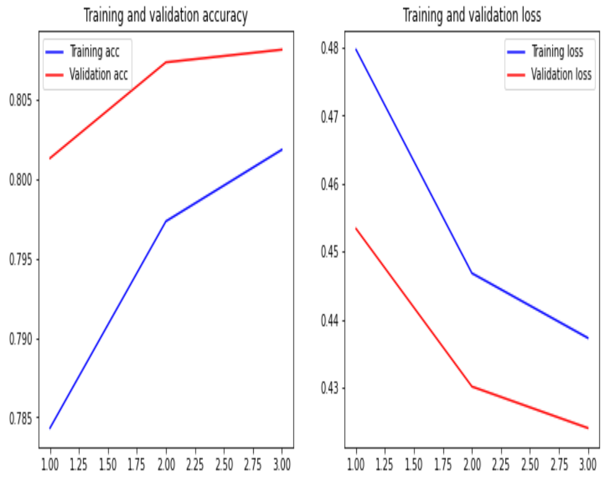

**Fig 6: Training and validation accuracy and training and validation loss for Stacked LSTM with 1D convolution**

From fig 6, it is observed that when the training accuracy has increased, validation accuracy has also increased. This means that the model is fitting the training set better, and also has given better performance to predict new data. Besides, training loss is slightly greater than validation loss. So, its performance can be better than previous model.

### *4.5.4 BERT Based Model*

Bidirectional Encoder Representations from Transformers BERT) is a deeply bidirectional, unsupervised language model which is processed for NLP pre-training using English Wikipedia and Brown Corpus by Google.BERT can assist computers to understand the language a bit more like humans do and focus on improving the execution of complex long-tail search queries and exhibit more relevant pursuit results.

We have implemented Bert-tokenizer for text classification with the help of TensorFlow 2.0. Here, we have done some steps like creating BERT Tokenizer, preparing data for training and generating the model. For training this model, we have used a batch size of 32, drop-out rate of 0.2 and 5 epochs. An accuracy of 83.2% has been obtained on the model.

## 5. EXPERIMENTAL RESULTS

Training and validation accuracy and loss have been measured while building, training, and tuning all models. A full overview of the models' accuracy can be found in the tables mentioned below.

### 5.1 Results of Baseline Models

**Table 1. Summary of all models using tf-idf features**

| Model | Accuracy | Precision | Recall | F1 score |
|---|---|---|---|---|
| **Multinomial Naive Bayes** | 0.748 | 0.97 | 0.77 | 0.85 |
| **Linear SVM** | 0.761 | 1.00 | 0.76 | 0.86 |
| **Logistic Regression** | 0.734 | 0.92 | 0.77 | 0.84 |

**Table 2. Summary of all models using count vectors features**

| Model | Accuracy | Precision | Recall | F1 score |
|---|---|---|---|---|
| **Multinomial Naive Bayes** | 0.825 | 0.94 | 0.85 | 0.89 |
| **Linear SVM** | 0.777 | 0.99 | 0.78 | 0.87 |
| **Logistic Regression** | 0.834 | 0.94 | 0.86 | 0.90 |

The comparison between all machine learning models using tf-idf and count vector has been shown in fig 7 according to their accuracy results.

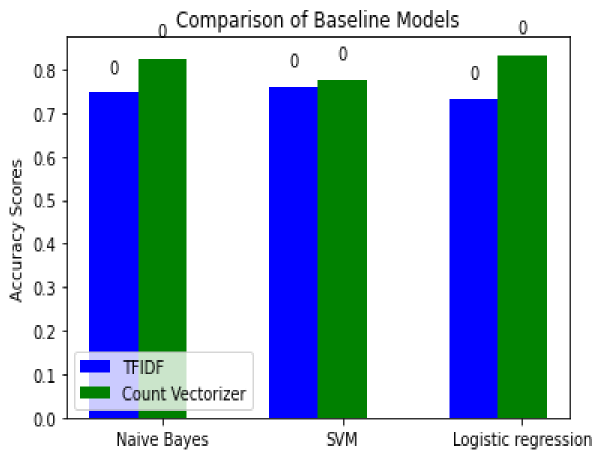

**Fig 7: Comparison of baseline models based on tf-idf and count vector**

## 5.2 Results of Deep Learning Models

**Table 3. Summary of deep learning models**

| Model | Accuracy | Precision | Recall | F1 score |
|---|---|---|---|---|
| Stacked LSTM Model | 0.824 | 0.80 | 0.16 | 0.27 |
| Stacked LSTM with 1D convolution | 0.81 | 0.82 | 0.95 | 0.88 |
| Bert based Model | 0.832 | 0.861 | 0.783 | 0.816 |
| CNN with pre-trained word embeddings | 0.841 | 0.839 | 0.84 | 0.839 |

The comparison between all deep learning models has been shown in fig 8 according to their accuracy results.

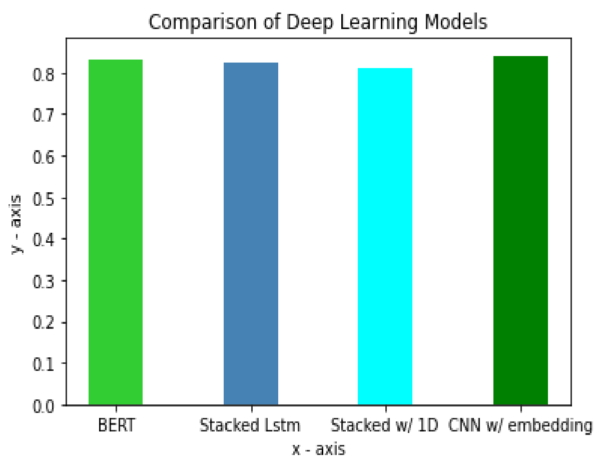

**Fig 8: Comparison of deep learning models**

## 5.3 Result Analysis

As we have worked on a binary classification problem, theoretically there is an accuracy of 50% for these models. That means, there is a 50% chance to be classified the tweets as "positive" and 50% chance for "negative" as our dataset contains only "positive" and "negative" tweets of the users. So all models which have been built performed better than the theoretical process.

Our CNN model with pre-trained word embeddings has performed the best with the accuracy of 84.1% after only 3 epochs.

Logistic regression using count vectors has performed slightly lower than the CNN model with an accuracy of 83.4%. Bert based model has performed almost the same as logistic regression with the accuracy of 83.2% after 5 epochs.

Logistic regression using TFIDF features has performed worst among all models.

The other models have also performed well with more than or equal 81% accuracy.

## 6. CONCLUSION

In this paper, we have presented our proposed models(Baseline models, BERT based model, CNN with pre-trained word embeddings, stacked LSTM model, stacked LSTM model with 1D convolution) for the emotion classification problem. Among them, CNN with pre-trained word embeddings has shown the best accuracy. Analyzing the results of all machine learning and deep learning models, it is clear that all the deep models have performed better than the machine learning models with more than 80% accuracy for our dataset. As it is a very challenging and tough task for research, our models have some limitations. It cannot work on the texts that have no emotion. Besides, as our dataset was very large, it takes a long time for training.

## 7. FUTURE WORK

The models which have been developed are working well. But still, these models can be improved by increasing accuracy. In the future, we will work on the Bengali dataset of any social media on these models and observe how these models work for the Bengali social media dataset.

As this work is a binary text classification problem, it is possible to apply these models on different text classification type problem like spam filtering, depression detection etc. We will use these models on the dataset of the mentioned text classification problem and observe how the developed models perform for different datasets.

## 8. REFERENCES

[1] R. Sawyer and G.-M. Chen, "The impact of social media on intercultural adaptation,"2012.

[2] "What is Twitter."https://esrc.ukri.org/research/impact-toolkit/social-media/twitter/what-is-twitter/.Accessed: 2019-05-30.

[3] KaliCornn,"Identifying Depression on Social Media."

[4] M. Cai, "Sentiment analysis of tweets using deep neural architectures,"

[5] M. Hoang, O. A. Bihorac, and J. Rouces, "Aspect-based sentiment analysis using bert,"in NEAL Proceedings of the 22nd Nordic Conference on Computional Linguistics

(NoDaL-iDa), September 30-October 2, Turku, Finland, no. 167, pp. 187–196, Linköping University Electronic Press, 2019.

[6] S. t. S. T. Avudaiappan.T, Jenifer, "Twitter sentimental analysis using neural network," in INTERNATIONAL JOURNAL OF SCIENTIFIC TECHNOLOGY RESEARCH VOLUME 9, ISSUE 02, pp. 2277–8616, 2020.

[7] J. Camacho-Collados and M. T. Pilehvar, "On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis," arXiv preprint arXiv:1707.01780, 2017.

[8] G. U. Srikanth et al., "Survey of sentiment analysis using deep learning techniques,"in2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT), pp. 1–9, IEEE, 2019.

[9] A. Hassan and A. Mahmood, "Deep learning approach for sentiment analysis of short texts," in2017 3rd international conference on control, automation and robotics (IC-CAR), pp. 705–710, IEEE, 2017.

[10] A. M. Ramadhani and H. S. Goo, "Twitter sentiment analysis using deep learning methods," in2017 7th International Annual Engineering Seminar (InAES), pp. 1–4, IEEE, 2017.

[11] H. Nguyen and M.-L. Nguyen, "A deep neural architecture for sentence-level sentiment classification in twitter social networking," in International Conference of the Pacific Association for Computational Linguistics, pp. 15–27, Springer, 2017.

[12] "Dataset."http://help.sentiment140.com/for-students/.Accessed: 2019-06-01.

[13] "Tfidf."https://www.geeksforgeeks.org/tf-idf-model-for-page-ranking/. Accessed: 2019-07-27.

[14] "Countvectors."https://medium.com/greyatom/an-introduction-to-bag-of-words-in-nlp-ac967d43b428.Accessed: 2019-07-27.

[15] Y. Kim, "Convolutional neural networks for sentence classification," in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP),pp. 1746–1751, 2014.

[16] "Googlnewsword2vecmodel."https://mccormickml.com/2016/04/12/googles-pretrainedword2vec-model-in-python/.Accessed:2019-07-16.

[17] "Stackedlstm."https://machinelearningmastery.com/stacked-long-short-term-memory-networks/.Accessed:2019-07-15.