CIS-positive: Combining Convolutional Neural Networks and LSTM for Sentiment Analysis in Twitter

1st Anusha Rasamalla
Edward E. Whitacre Jr. College of Engineering
Texas Tech University
Lubbock, Texas, USA
arasamal@ttu.edu

2nd Nikhil Adsul
Edward E. Whitacre Jr. College of Engineering
Texas Tech University
Lubbock, Texas, USA
nadsul@ttu.edu

3rd Prakriti Adhikari
Edward E. Whitacre Jr. College of Engineering
Texas Tech University
Lubbock, Texas, USA
prakadhi@ttu.edu

4th Prastab Dhakal
Edward E. Whitacre Jr. College of Engineering
Texas Tech University
Lubbock, Texas, USA
rkhamkar@ttu.edu

5th Rushikesh Khamkar
Edward E. Whitacre Jr. College of Engineering
Texas Tech University
Lubbock, Texas, USA
rkhamkar@ttu.edu

***Abstract*—the paper" Combining Convolutional Neural Networks and LSTM for Sentiment Analysis in Twitter" is the reimplementation and improvement of CNN+SVM model from our base paper. It introduces the automatic sentiment analysis system. We describe the approach taken by combining layer of CNN and LSTM network o achieve good accuracy. The authors of this paper propose that Twitter records be normalized in a way that maximizes sentiment lexicon coverage while minimizing distracting elements, and that the output of Convolutional Neural Networks (CNN) be integrated into Long Short Term Memory (LSTM) for polarity classification. Their system obtains a contextual Accuracy results of the positively and negatively 'Sentiment 140' dataset with a class of 78% which is better than our previous CNN-SVM model (59% accuracy) from our previous base paper. A few novel solutions implemented in the paper will be tried to increase the model accuracy for better performance. We have compared some classification, regression and deep learning models to compare and analyze the accuracy with our model.**

keywords: CNN, SVM, word embedding, f1-macro

I. Introduction

Texts that express various sorts of feelings are wherever on the Internet. A wide range of sorts of partners gets benefit from information mining. For instance, client criticism on the qualities and shortcomings of an item is important to the endeavor. Clients need to rapidly audit inn surveys and appraisals prior to booking their next get-away. Government officials attempt to anticipate the result of the official political decision. These requirements can be met with a mechanized state of mind investigation framework. Twitter, an interpersonal interaction administration, is a wellspring of this sort of data and ranges an assortment of disciplines and subjects. Its fame and client usefulness in creating new fixings have been intriguing examination points. Nonetheless, Twitter has its own concerns, as portrayed beneath. By and large, robotized opinion investigation is troublesome because of various elements, including: B. Uncertain word implications, setting touchy, amusing, certain properties of Twitter text make this assignment much more troublesome. 140 people limit A horde of abbreviations and contractions are shown for each message. What's more, a large portion of the tweets has a casual character and contains aim Incorrect spelling and abuse of syntax. Subsequently, Out of jargon (OOV) rate for Twitter text it is costly to the point that data is lost. One of the most widely recognized undertakings in SemEval 2015 - Tasks 10: Sentiment examination on Twitter to manage this Challenge (Rosenthal et al., 2015). I partook In subtask B," Message extremity characterization" task. The objective is to foresee a specific measure of extremity Tweet in certain, negative and nonpartisan. task The coordinator gave the Tweet ID and the comparing name to have a typical establishment for preparing extremity. Order framework. More data about Assignments, other subtasks, and how the information was chosen is portrayed in (Rosenthal et al., 2015). The contributions of the paper can be summarizing as follows: (1) we design a combined architecture of the 2 consecutive layer of the Convolutional Neural Network (CNN) and 2 consecutive layer of the long Short Term Memory (LSTM). (2) We have discussed a comparison about CNN-LSTM model with other 4 Model to check accuracy of the model including CNN-SVM Model from our previous experiment file. (3) We have achieved model accuracy around 78.20%, also achieve precision, recall and F1-score with 78, 78 and 0.7825 respectively for our Twitter Sentiment Data. This paper is organized as follows. Section 2 gives details about related work on previous implementation of CNN-SVM Model accuracy and current work. Section 3 related to Dataset used in this paper and Data preprocessing techniques followed for accuracy improvement. Section 4 related to different models used for comparison along with our proposed model – CNN-LSTM. Section 5 gives overview about implementation details that we used in our experiments. Section 6 talks about comparison and analysis of our all experimented model. Section 7 gives you an expected future work. And in Section 8, we are concluding our paper with conclusion section.

II. Related Work

Our previous paper ”CIS-positive: Combining Convolutional Neural Networks and SVM for Sentiment Analysis in Twitter” is part of SemEval 2015 Task 10 ”Sentiment Analysis in Twitter” with sub-task-B ”Message Polarity Classification,” introduces the automatic sentiment analysis system. We describe the approach taken by the authors, a re-implementation of the paper and few potential novel solutions to achieve better F1-score results. The authors of this paper propose that Twitter records be normalized in a way that maximizes sentiment lexicon coverage while minimizing distracting elements, and that the output of Convolutional Neural Networks be integrated into Support Vector Machines for polarity classification. Their system obtains a contextual Accuracy results of the positively and negatively ’SemEval 2015 test data’ with a class of 59.57. However, while reproducing the results we find the macro F1 score to be 63.40 which is similar to the paper. A few novel solutions proposed in the paper will be tried to increase the model accuracy for better performance. Convolutional Neural Network works better individually for feature extraction but it won’t provide text contextualization. And, LSTM provides sequential contextualization for text data but it might lead you to bias. So, we have merged the advantage of 2 models and achieve good accuracy. We have merged 2 CNN layer with 2 LSTM layer with fully connected layer which gives us accuracy around 78%. Along with CNN-LSTM, we have experimented other classification model for sentiment analysis such as Logistic Regression, individual CNN, individual LSTM, and Naïve Bayes. We have discussed advantages of those models and their accuracy in this paper.

III. Dataset

The twitter training data, Sentiment140 Dataset consists of a total training data of 696112 tweets that were downloaded, of which 347833 were positive and 348279 were negative. Evaluations were conducted on Sentiment 140 twitter dataset from Kaggle which consists of a total of 16 Lakh records: we used 7 lakh total records for our model. It contains - Short text with 347833 positives, 348279 negatives. SMS, blog posts (Live Journal), and tweets that are marked as sarcastic were also contained in this set. The tweets have been annotated (0 = negative, 1 = positive).

IV. Data Preprocessing

Twitter texts are testing and contrast from other areas in a few explicit properties. Because of the 140 characters cutoff of tweet length, clients make weighty utilization of truncations and abbreviations. This prompts a high OOV rate and makes assignments like tokenizing, grammatical feature (POS) labeling, and dictionary search more troublesome. Moreover, extraordinary tokens, for example, client specifies (e.g., ”@isar”), urls, hashtags (e.g., ”happy”), and accentuation groupings like ”!?!?” are frequently used. In this way, standardization of all tweets is important to work with later extremity order. Our text preprocessing pipeline can be depicted as follows: Tweets are first tokenized what’s more, POS labeled with the CMU tokenize and tagger (Owoputiet al., 2013). This tagger is particular for Twitter and hence better than other general space taggers. Subsequently, all client specifies are supplanted by ”¡user¿” and all urls by ”¡web¿”, in light of the fact that they give no prompts of extremity. We try not to supplant hash tags, on the grounds that they frequently contain significant data like themes or even opinion. Accentuation arrangements like”!?!?” can go about as distortion or other extremity modifier. Notwithstanding, the sheer measure of potential successions builds the OOV rate significantly. Subsequently, all successions of accentuations are supplanted by a rundown of particular accentuations in this succession (e.g., ”!?!?” is supplanted by ”[!?]”). That lessens the OOV rate regardless keeps the majority of the data. In our Data preprocessing technique, we are following some sequence while preprocessing which gives us good accuracy and it is explained in Fig 1. We are converting all Unicode to text. In twitter, when the text is saved in their dataset, it converts little text into Unicode such as it converts symbol into Unicode or change number to Unicode. So, when we are preprocessing, we need to consider those Unicode into text for getting correct text contextualization. We are removing emoji and emoticons, duplicate punctuations and Tag user keywords to get exact context of the text. In next steps, we are removing words with less than length 2, because in sentiment analysis, it is not carrying any meaning of the text which might affect our textual sentiments.

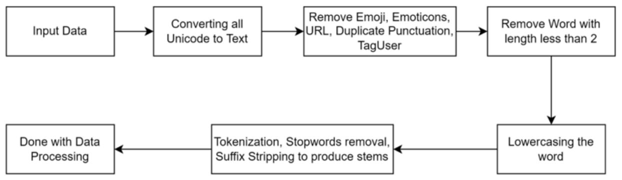
Input Data
Converting all Unicode to Text
Remove Emoji, Emoticons, URL, Duplicate Punctuation, TagUser
Remove Word with length less than 2
Lowercasing the word
Tokenization, Stopwords removal, Suffix Stripping to produce stems
Done with Data Processing

Fig. 1. Data processing steps

So, we are not considering those texts. We have considered a contraction mapping in sentence for getting full forms of words. Because of word limits, sometime users are using short forms of words which might affect our sentiment analysis. Using Contraction mapping, we are replacing short forms of text with full forms. We removed stop words, Part of Speech tagging (POS) words. POS words mean same word defines different meaning in different context. POS tagging has been applied after tokenization and it helps us to preserve sentence structure. In twitter data, due to short message formats, users are writing words with so many inflections. Inflection means text is processing through many grammatical formats such tense, voice, gender, number, and mood. Meaning of the words is different, but the base or main word is used in this entire context. While preprocessing, it will be difficult to preprocess same word different text. So we performed stemming technique on the word. For example – we have some list of words like Connection, Connecting, Connects, Connected, Connecting, Connecting's, and Connect. All of these words represent same word – Connect. So, we are using PorterStemmer for stemming for suffix stripping to produce stems. PorterStemmer is very simple and fast for execution. It removes morphological and inflexional endings of the words in English.

V. Model defined in the paper

The system architecture of the sentiment analysis model mentioned in this paper consists of three major components. The first component is a CNN, which uses the sequence of all words in a tweet to give a softmax output. The second component is a SVM classifier which makes use of some linguistic features and the CNN's softmax output from earlier as its input. Finally, another SVM is used on top of the CNN and the SVM to combine the polarity prediction , which consequently helps in receiving the final polarity label. All the components are explained in detail in this section.

A. CNN

The idea behind a CNN for sentence modeling is to have a model that captures sequential phenomena and takes into account words in their contexts. In a bag-of-words approach, the word does not indicate negation, but it is not related to the words it negates. Even if the problem could be solved by using n-grams, long-distance effects would still remain unaccounted for. A bag-of-words model also suffers from sparsity. Convolutional neural networks perform mathematical convolutions with the inputs and matrices to process sequences. The objective is to find salient features in the input sequence that indicate polarity and to merge them into a meaningful representation. More specifically, the words are represented by two matrices. Here, $P \in \mathbb{R}^{d_p \times V}$ denotes a matrix of low dimensional representations of words, which is referred to as a word embedding.d_p the size of the embeddings, is usually set to 50-300, depending on the task. V denotes the size of the vocabulary. The matrix P is learned during model training. It is initialized either randomly or with a pretrained matrix, as we will describe later. In addition to P, we introduce another matrix $Q \in \mathbb{R}^{d_q \times V}$ which contains external word features. In this case,d_q is the number of features per word. This approach allows us to add as much external knowledge into the training process as needed. The features are precomputed and not embedded into any embeddings space, i.e. Q is fixed during training. A description of all features is given later in this section. Both components are concatenated into a lookup table

$$LT = \begin{bmatrix} P \\ Q \end{bmatrix}$$

where each column corresponds to the entire representation of a certain word in the vocabulary. Given a sentence of n words w_1 to w_n,the model concatenates all n word representation to the input of the CNN

$$S = \begin{bmatrix} \text{LT:,w}_1 & \dots & LT:,w_n \end{bmatrix}$$

A one dimensional convolution is a mathematical operation

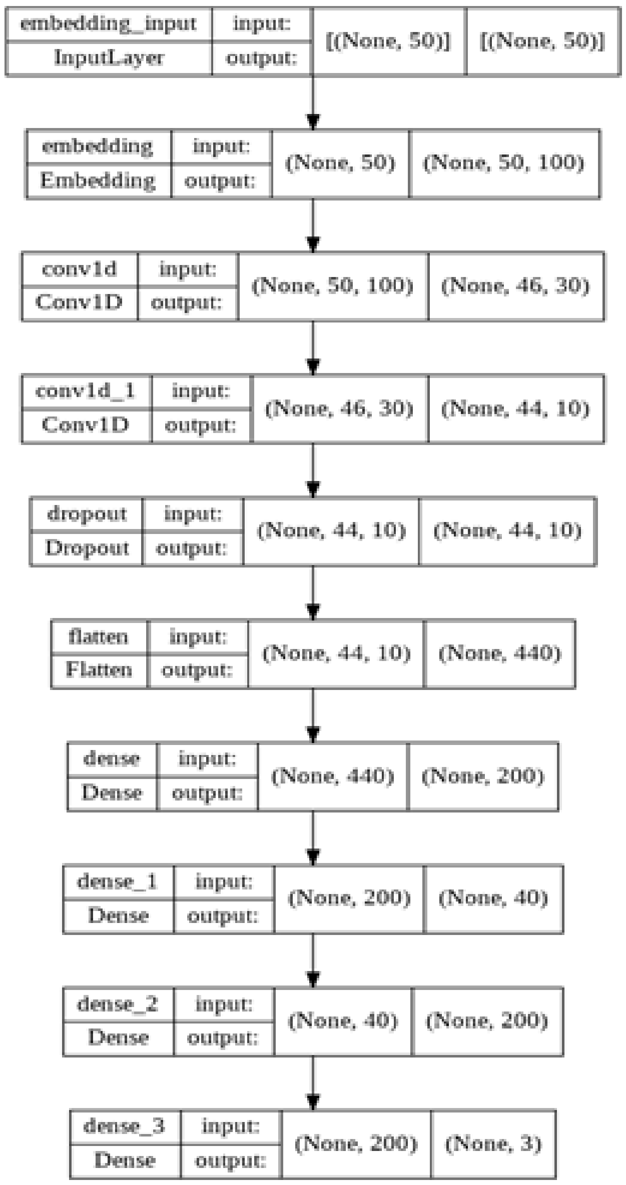
embedding_input | input: | [(None, 50)] | [(None, 50)]
InputLayer | output:
embedding | input: | (None, 50) | (None, 50, 100)
Embedding | output:
conv1d | input: | (None, 50, 100) | (None, 46, 30)
Conv1D | output:
conv1d_1 | input: | (None, 46, 30) | (None, 44, 10)
Conv1D | output:
dropout | input: | (None, 44, 10) | (None, 44, 10)
Dropout | output:
flatten | input: | (None, 44, 10) | (None, 440)
Flatten | output:
dense | input: | (None, 440) | (None, 200)
Dense | output:
dense_1 | input: | (None, 200) | (None, 40)
Dense | output:
dense_2 | input: | (None, 40) | (None, 200)
Dense | output:
dense_3 | input: | (None, 200) | (None, 3)
Dense | output:

Fig. 2. CNN model implementation.

that slides a filter $m \in \mathbb{R}^{1 \times m}$ over a vector and computes a dot product at every position. The length of the filter m specifies how many elements the filter spans. Applying this concept to a two dimensional input leads to a convolution matrix where the elements are computed by $C_{i,j} = m^T S_{i,j:j+m-1}$ where i is the ith row in S and j is the start index of the convolution. A, the output of the convolution layer is computed by an element-wise addition of a bias term (one bias per row) and an element-wise non-linearity: $A = f(C + b)$.As non-linear function we use a rectified linear unit: $f(x) = max(0, x)$. This non-linearity proved to be a crucial part in object recognition (Jarrett et al., 2009), machine translation (Vaswani et al., 2013), and speech recognition (Zeiler et al., 2013). Our model uses two layers of convolution. The concatenation of all rows of the second convolution layer output is the input to a sequence of three fully connected hidden layers. A hidden layer transforms the input vector x into $z = f(Wx + b)$, where W is a weight matrix that is learned during training and b is a bias. In order to convert the final hidden layer output z into a probability distribution over polarity labels $o \in \mathbb{R}^3$, the softmax function is used:$o_i = \frac{exp(z_i)}{\Sigma_j exp(z_j)}$

Pretraining of Word Embeddings Standard practice involves sampling from the uniform distribution when initializing the matrix of word embeddings. Because there is such a limited amount of training data, it is not possible to learn word representations from scratch without overfitting. To prevent this from happening, we precompute word embeddings on a large volume of Twitter text data using the word2vec toolkit instead of initializing the word embeddings matrix randomly.1 We first downloaded about 60 million tweets from the unlabeled Twitter Events data set (McMinn et al., 2013). V words are selected, comprising all the words of the SemEval training data, words from the sentiment lexicons, and the most frequent words of the Twitter Events data set. Finally a continuous bag-of-words model (Mikolov et al., 2013) with 50 dimensional vectors is trained and used to initialize P.

Word Features In addition to the word embeddings the CNN receives additional external features (matrix Q). These features are the following:

binary sentiment indicators Binary features that indicate the polarity of a token in a sentiment lexicon. The lexicons for this feature are MPQA (Wilson et al., 2005), Opinion lexicon (Hu and Liu, 2004) and NRCC Emotion lexicon (Mohammad and Turney, 2013).

sentiment scores Both the sentiment 140 lexicon and the Hashtag lexicon (Mohammad et al., 2013) assign a score to each token instead of just a label. These scores are directly used by us. Both lexicons also contain scores for bigrams and skip n-grams. In such a case each word of an n-grams receives the score of the entire n-grams.

binary negation As outlined in Christopher Potts' Sentiment Symposium tutorial, each token between a negation and the next punctuation is marked as negated.

B. *SVM1*

As epoch training for the CNN always led to overfitting, SVM is used as a second classifier. Similarly the following features of Mohammad et al. are followed in the paper (2013):

- **binary bag-of-words** Binary bags of words that include bigrams or trigrams are not used, because it degraded the performance on the validation set. In contrast to Mohammed et al. (2013), this system does not use character ngrams or trigrams of higher order.
- **sentiment features** The following features are added for every tweet and every lexicon :Total number of tokens in the tweet that are lexicon tokens, total sentiment score, maximum sentiment score, and the sentiment score of the last token in the tweet.
- **CNN output** is used to inform the SVM about the CNN's classification decision and certainty. Additionally, we add the CNN's softmax output as an additional feature.As linear SVM implementation, LIBLINEAR is used. (Fan et al., 2008).

C. *SVM2*

After the analysis of the CNN and SVM1 predictions it was found that both of the classifiers learn orthogonal features. In the classification pipeline, a second linear SVM replaces the CNN with a second linear SVM that combines the softmax probability scores of the CNN and the confidencescores of the first SVM. The output is the final predicted polarity label of the proposed system.

VI. Refined model

The model defined in the paper were refined by us after some extensive research for better accuracy and performance. The system architecture of the various sentiment analysis models that were compared and analyzed consist of various components. All the components are explained in detail in this section.

A. *Logistic Regression*

Logistic Regression is a supervised machine leaning technique for classification problems. It is Binary Logistic Regression where outcome is binary such as 0 and 1 or negative or positive or yes and no. This algorithm works on train labeled dataset along with predicted value [24]. The goal of the model is to learn from predicted value and map approximate values to Y value. In our model, we used X label as twitter comment and twitter prediction (positive / negative) is labeled as Y value. Accuracy of the model gives us 74.44% result.

B. *Naive Bayes*

Naïve Bayes classifier works on naïve assumptions of no relationship among different features. It checks the presence of the particular feature in a class is unrelated to other features [23]. Naïve Bayes model works best for large size dataset.

In our naïve bayes model, we are experimenting 4 naïve bayes algorithm –

- Multinomial NB – It works on discrete count of twitter text data. In our experiments, it gives us result of 72.83% accuracy.

- Complement NB – For imbalance datasets, multinomial model fails to give correct accuracy sometime. In complement NB, it is not only calculate probability for certain class, it will calculate probability of data belong to all classes. In our case, we got accuracy of 72.83% result.

- Bernoulli NB – This model is helpful when our prediction values are 0 and 1 [23]. This model gives accuracy of 73.82%.

- Gaussian NB – It consider that feature follows normal distribution. In our experiments, it gives us result of 70.31% accuracy.

C. SVM

Support Vector Machines are a type of supervised machine learning algorithm that provides analysis of data for classification and regression analysis. While they can be used for regression, SVM is mostly used for classification. In our model we have used the linear SVM classifier with train data of 70% and test data of 30%. Our model has the precision of 76%, recall of 76% and an accuracy of 77% on the kaggle data.

D. CNN

The idea behind a CNN for sentence modeling is to have a model that captures sequential phenomena and takes into account words in their contexts. In a bag-of-words approach, the word does not indicate negation, but it is not related to the words it negates. Even if the problem could be solved by using n-grams, long-distance effects would still remain unaccounted for. A bag-of-words model also suffers from sparsity. Convolutional neural networks perform mathematical convolutions with the inputs and matrices to process sequences. The objective is to find salient features in the input sequence that indicate polarity and to merge them into a meaningful representation. More specifically, the words are represented by two matrices. Here, $P \in \mathbb{R}^{d_p \times V}$ denotes a matrix of low dimensional representations of words, which is referred to as a word embedding.d_p the size of the embeddings, is usually set to 50-300, depending on the task. V denotes the size of the vocabulary. The matrix P is learned during model training. It is initialized either randomly or with a pretrained matrix, as we will describe later. In addition to P, we introduce another matrix $Q \in \mathbb{R}^{d_q \times V}$ which contains external word features. In this case,d_q is the number of features per word. This approach allows us to add as much external knowledge into the training process as needed. The features are precomputed and not embedded into any embeddings space, i.e. Q is fixed during training. A description of all features is given later in this section. Both components are concatenated into a lookup table

$$LT = \begin{bmatrix} P \\ Q \end{bmatrix}$$

where each column corresponds to the entire representation of a certain word in the vocabulary. Given a sentence of n words w_1 to w_n,the model concatenates all n word representation to the input of the CNN

$$S = \left[\ \text{LT:,w}_1 \qquad \ldots \qquad LT:, w_n\ \right]$$

E. LSTM

In Feed Forward Neural Network, Inputs are multiplied by weight and bias is merged with that and finally we get the output in last layer of the network. They don't store values in memory and can't use for sequential analysis. And, that's the reason, CNN is best in Feature Extraction, but it fails in Sequential context analysis. This is the Recurrent Neural Network is introduced in machine learning model which allows us to store weight after each calculation. It allows model to use the information of the specific time instance is used as input for the next time instance [25]. But the problem in RNN is, it works with Vanishing Gradient. In RNN, it calculates value based on previous available weights. Each gradient calculated for weights are multiplied back through their networks. If the size of neural network is more then, gradient value becomes weaker which causes the gradient to vanish. To overcome this problem, LSTM is introduced. LSTM is updated version of Recurrent Neural Network (RNN) which stores information from a particular time instance to next time instance. It uses extra memory cell which is only used for storing information from one instance to other instance which allows model to remember previous states and it prevents model from vanishing gradient problem [25]. In Our LSTM model, we have used one embedding layer which contains 100 dimensional glove vector arrays which has predefined values for word list. We are using 2 layer of Bidirectional LSTM. First LSTM layer contains unit size of 64 with return sequence as True. 64 units for first layer indicate the memory units to each input of the sequence. Return Sequence returns the hidden state output for each input time step. Second LSTM layer contains 128 size units. This LSTM layer is followed by 2 fully connected layers with size 64 and 1 with sigmoid activation function. For model training, we used 15 epochs with batch size of 256 which give accuracy of 77.83% which is better than our previous CNN model.

F. CNN-LSTM

Convolutional Neural Network (CNN) performs better in obtaining in important features using pooling. And, RNN Long Short Term Memory (LSTM) works better in obtaining in contextual information from a sequence data. But, both modules have their own disadvantage in terms of sentiment analysis in textual representations. In CNN, it will give extracted features from text but it will fail to return contextual information. And, In RNN (LSTM) Model, it will give us contextual information, but it might fail to gives us predictions and it might lead us to bias. Idea behind to combined those 2 models CNN and LSTM (RNN) is that take to take advantage of those models to avoid their disadvantage to get better performance. Combining the advantage of CNN that can extract effective features from the data and LSTM's finding in sequential contextual information will not only give interdependence of data but automatically detect the best suitable relationships in sentence context. We are using generic name for CNN-LSTM to refer to as CNN model's output will be input set for LSTM [22]. Following model demonstrates architecture CNN-LSTM model in Fig 2. Our CNN-LSTM network has 2 convolutional layer with filter size of 16 and 32, one dropout layer, 2 LSTM layer with units 64 and 128, 2 Fully connected layer with size 64 and 1 which followed by Sigmoid function. First convolutional layer contains 16 filters with Kernel size 3 and Second convolutional layer contains 32 filters with kernel size 3 with 'Relu' as Activation function. Then, we used Dropout function for data regularization. In LSTM, we introduced 2 LSTM layer with unit size of 64 with return sequence as True which allow model to learn from previous experiments and in other LSTM model, 128 units we used. 2 Fully Connected Layer, we used after LSTM layer with node size of 64 and 1 with Sigmoid function for representing probability value as our result for data. After training, Model gives us result in 2 categories – Positive and Negative value. Fig 3 proposed a summary of CNN-LSTM model for sentiment analysis.

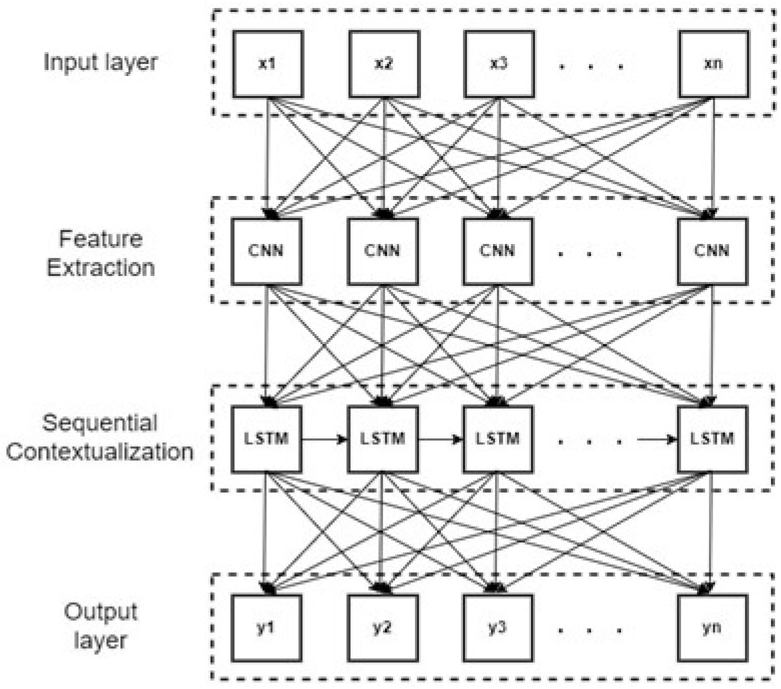

Fig. 3. CNN-LSTM Architecture.

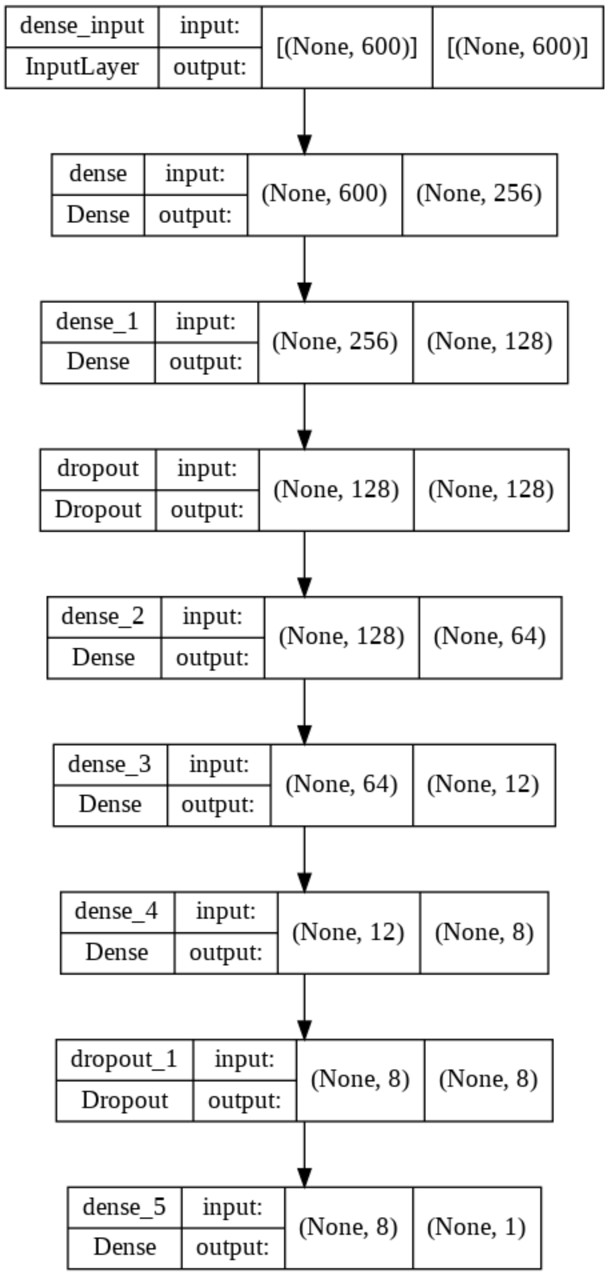
dense_input | InputLayer | input: | output: | [(None, 600)] | [(None, 600)]
dense | Dense | input: | output: | (None, 600) | (None, 256)
dense_1 | Dense | input: | output: | (None, 256) | (None, 128)
dropout | Dropout | input: | output: | (None, 128) | (None, 128)
dense_2 | Dense | input: | output: | (None, 128) | (None, 64)
dense_3 | Dense | input: | output: | (None, 64) | (None, 12)
dense_4 | Dense | input: | output: | (None, 12) | (None, 8)
dropout_1 | Dropout | input: | output: | (None, 8) | (None, 8)
dense_5 | Dense | input: | output: | (None, 8) | (None, 1)

Fig. 4. Configured CNN-LSTM model.

VII. Experiment

Using minibatch stochastic gradient descent the CNN-LSTM model is trained with a batch size of 256 and initial learning rate of 0.001 and learning rate adaptation as AdaGrad. This learning rate λ = 0.01 avoids over fitting as much as possible. In the system dveloped by the authors,the model uses two convolution layers. In the first convolution layer, 30 filters with m = 5 was used which means it spans 5 words. The second convolution layer uses 10 filters with the m = 3. There are three hidden layers that have sizes 200,40,200 that are used in most of the speech recognition systems.The C parameter on the validation data for both linear SVMs is

tuned. The Softmax output of the CNN is fed as an input to SVM! with pretrained values. The SVM2 uses the CNN softmax and SVM confidence matrix as inputs that gives the output as probabilities for classification.
In the refined system with added features, the model uses two convolution layers. In the first convolution layer, 16 filters with m = 3 was used which means it spans 3 words. The second convolution layer uses 32 filters with the m = 3. CNN layer is followed by LSTM. 2 LSTM layer is used in our model, first LSTM model contains 64 units and second model contains 128 units. There are two hidden layers that have sizes 64 and 1 that are used in most of the speech recognition systems. The Softmax output of the CNN is fed as an input to LSTM with pre-trained values. Our model is trained on total 13,588,529 parameters. Output of the CNN-LSTM model takes to sigmoid activation function.

VIII. Results

After using preprocessed data for training model like logistic regression, naïve bayes, CNN, LSTM, CNN-LSTM, model is completed by training is used to predict the test data. The performance on negative examples is much better than previous used model (CNN-SVM). The fact that this system scores much better on Life Journal and the SMS data in terms of F1, negative suggests that Twitter is an especially difficult medium for automated analysis. The comparison of model accuracy has shown in below table 1. In this able, we have compared all experimented model. It shows our CNN+LSTM model works better than CNN+SVM model. CNN+LSTM model gives us accuracy of 78% which is better than our previous implemented model (CNN+SVM) that has accuracy of 63%. We have achieved a great accuracy using some good data preprocessing technique, glove predefined vector and model combination. Below graphical representation shows accuracy comparison between models.

In our efforts to improve the model for the better prediction of the sentiment of the twitter data, we analyzed and compared the accuracy and f1- scores of multiple models. These models were trained individually and in combination with other models. The models we trained and tested are Logistic Regression model, CNN (Convolutional Neural Network) model, LSTM (Long Short-Term Memory) model, Bernoulli Naïve Bayes model, SVM (Support Vector Machine) model and the combination of CNN +LSTM and CNN+SVM, where the latter was already proposed by the research paper. After training and testing these modules, we found that the Logistic regression Model gave the f1-score of 0.75, CNN gave 0.74, LSTM gave 0.76, Bernoulli NB gave 0.70, and CNN+LSTM gave 0.782. Hence, we can say that the model with the combination of CNN and LSTM performed the best with the highest f1-score of 0.782.

- Model performance based on research paper

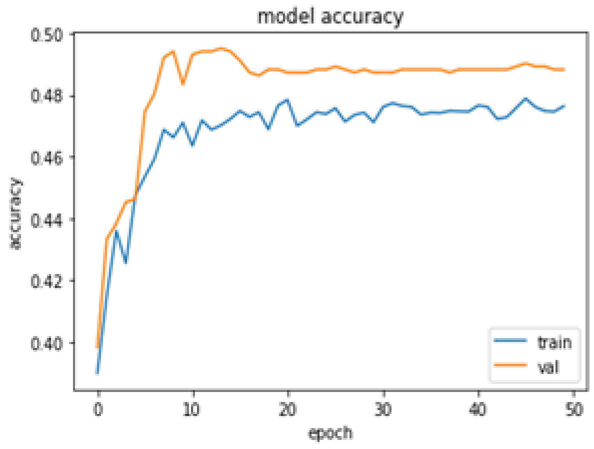

Fig. 5. CNN-SVM Model Accuracy

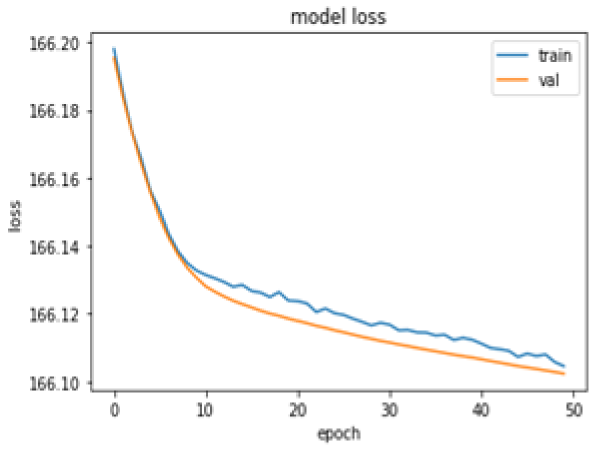

Fig. 6. CNN-SVM Model Loss

- Model performance after further improvement

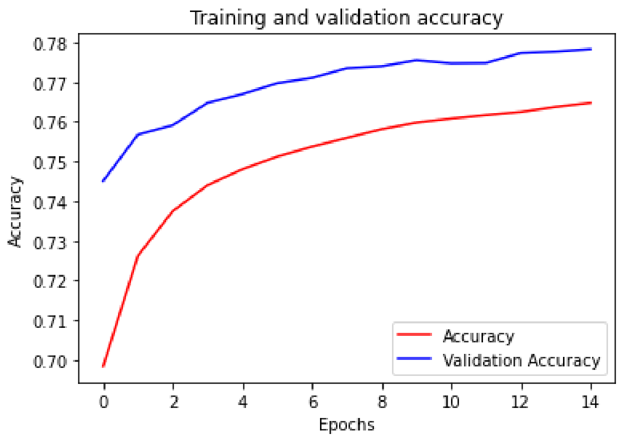

Fig. 7. CNN-LSTM Model Accuracy

TABLE I
COMPARISON ANALYSIS OF TRAINED MODELS

Model	Sentiment	Precision(%)	Recall(%)	F1-score
Logistic Regression	Negative	76	71	0.74
	Positive	73	78	75
Bernoulli NB	Negative	72	74	74.5
	Positive	73	71	72
CNN	Negative	76	75	75.5
	Positive	75	74	0.745
LSTM	Negative	76	74	75
	Positive	75.2	77	78
SVM	Negative	76	75	75.5
	Positive	75	74	74.5
CNN+SVM	Negative	62	62	63
	Positive	60.6	60	59.7
CNN+LSTM	Negative	77	79	78
	Positive	78	77	78

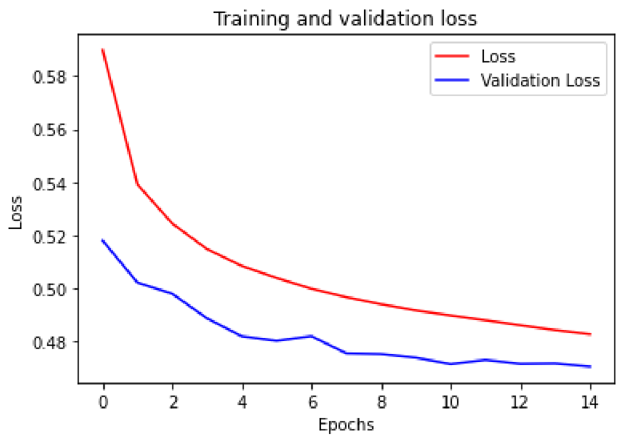

Fig. 8. CNN-LSTM Model Validation Loss

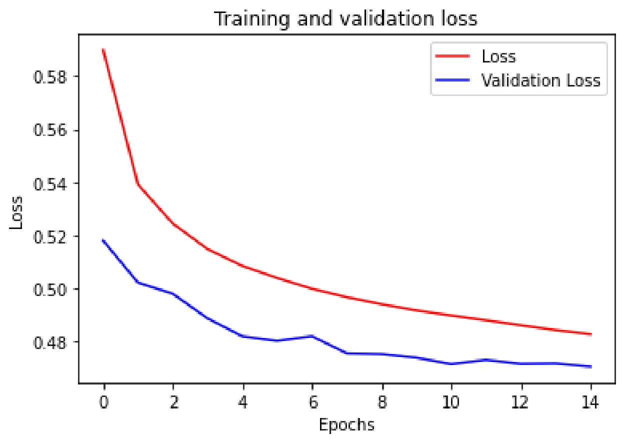

Fig. 10. LSTM Model Validation Loss

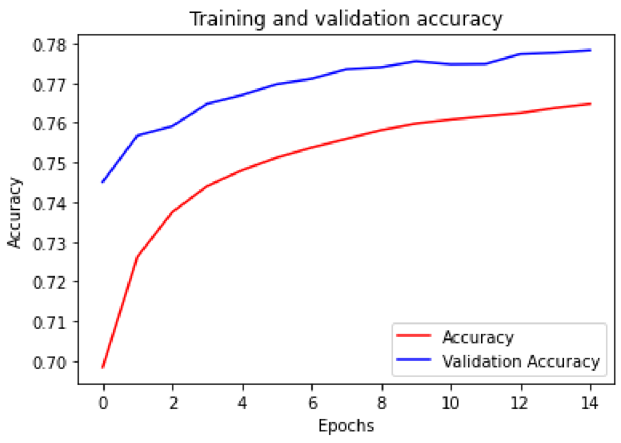

Fig. 9. LSTM Model Validation accuracy

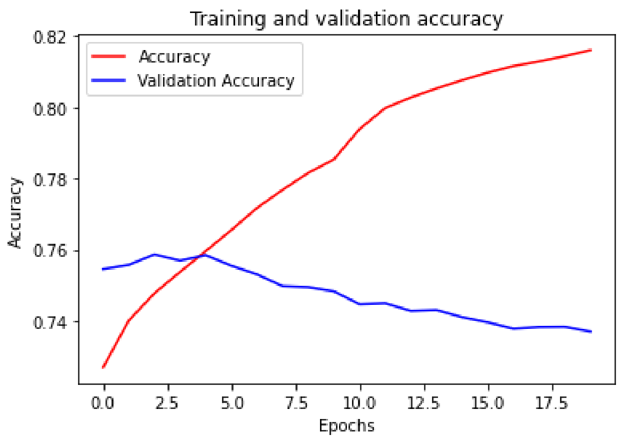

Fig. 11. CNN Model Validation accuracy

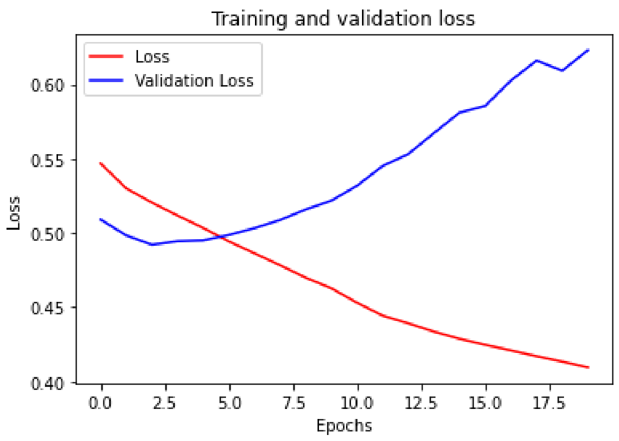

Fig. 12. CNN Model Validation Loss

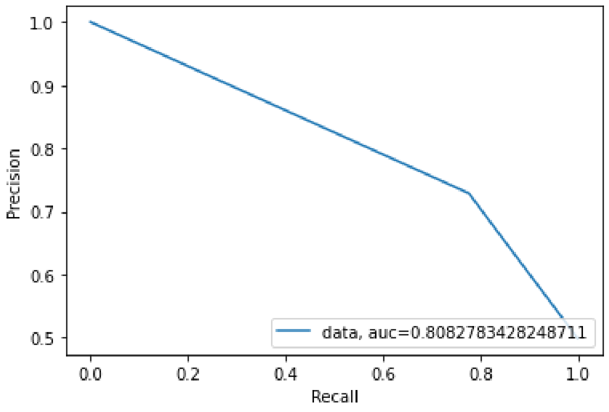

Fig. 13. LR model precision recall curve

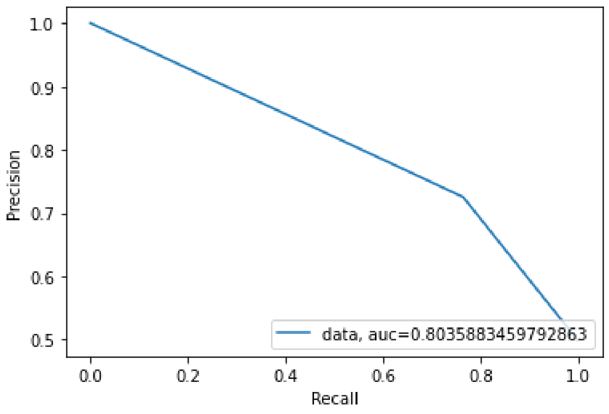

Fig. 14. Bernoulli NB model precision recall curve

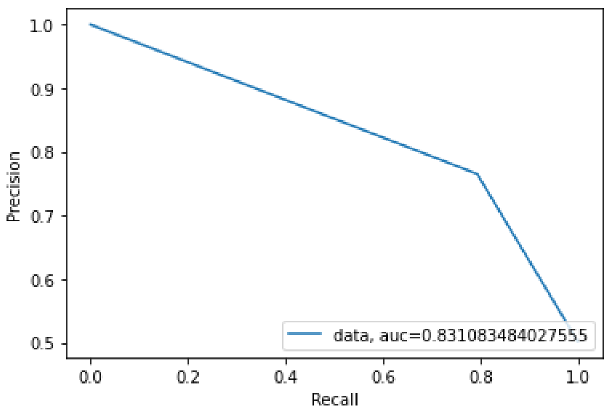

Fig. 15. SVM model precision recall curve

IX. Conclusion

This paper suggests using CNN and SVM for the classification of Twitter data removes the problem of overfitting and orthogonal features. The results on the negative sentiment of the Twitter texts are achieved poor because the data is not enough to train the model. Thus, we have proposed to use the Kaggle dataset for making better predictions. This paper summarizes the features of our automatic sentiment analysis system – CIS-positive – for the SemEval 2015 shared task - Task 10, subtask B. The Twitter data is normalized, pre-processed, filtered, and integrate the output of convolutional neural networks into support vector machines for the polarity classification. The system achieves a macro-F-score of 59.57 on the SemEval 2015 test data. While reproducing the results we gotthef1-macro of 63%which is better than the proposed paper. Moreover, we tried to improve the working of the model by using various other algorithms and their combinations. We checked and compared the accuracy of CNN, LR, LSTM, SVM, Naive Bayes and the combination of LSTM and CNN to find the best model for sentiment analysis. Hence, as per our comparison and analysis, the model with the combination of CNN and LSTM performed the best with the accuracy of 78.2% on the Kaggle data.

Acknowledgment

We would like to thank the authors of the original paper for providing the facts and figures for automated sentiment analysis. We would also like to thank them for providing the methodology for getting to the results. We would also like to thank our professor, Dr. Victor Sheng, for providing guidance to complete and run this research project.

References

[1] Sebastian Ebert, Ngoc Thang Vu and Hinrich Schütze “CIS-positive: Combining Convolutional Neural Networks and SVMs for Sentiment Analysis in Twitter” 2015 Association for Computational Linguistics.
[2] Samuel Brody and Nicholas Diakopoulos. 2011. Cooooooooooooooollllllllllll!!!!!!!!!!!!!! Using Word Lengthening to Detect Sentiment in Microblogs. In EMNLP
[3] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (almost) from Scratch. JMLR, 12.

[4] John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. JMLR, 12.

[5] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. JMLR, 9.

[6] Frantisek Grézl, Martin Karafiát, Stanislav Kontar, and Jan Cernocký. 2007. Probabilistic and Bottle-Neck Features for LVCSR of Meetings. In ICASSP.

[7] Minqing Hu and Bing Liu. 2004. Mining and Summarizing Customer Reviews. In KDD.

[8] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. 2009. What is the Best MultiStage Architecture for Object Recognition? In ICCV.

[9] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In ACL.

[10] Andrew J. McMinn, Yashar Moshfeghi, and Joemon M. Jose. 2013. Building a large-scale corpus for evaluating event detection on twitter. In CIKM.

[11] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In ICLR.

[12] Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a Word-Emotion Association Lexicon. Computational Intelligence, 29(3).

[13] Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. NRC-Canada: Building the State-of-theArt in Sentiment Analysis of Tweets. In SemEval.

[14] Preslav Nakov, Sara Rosenthal, Zornitsa Kozareva, Veselin Stoyanov, Alan Ritter, and Theresa Wilson. 2013. SemEval-2013 Task 2: Sentiment Analysis in Twitter. In SemEval.

[15] Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved Part-of-Speech Tagging for Online Conversational Text with Word Clusters. In NAACL HLT.

[16] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs Up?: Sentiment Classification Using Machine Learning Techniques. In EMNLP.

[17] Sara Rosenthal, Alan Ritter, Preslav Nakov, and Veselin Stoyanov. 2014. SemEval-2014 Task 9: Sentiment Analysis in Twitter. In SemEval.

[18] Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif M. Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. SemEval-2015 Task 10: Sentiment Analysis in Twitter. In SemEval.

[19] Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with Large-Scale Neural Language Models Improves Translation. In EMNLP.

[20] Theresa Ann Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing Contextual Polarity in PhraseLevel Sentiment Analysis. In HLT/EMNLP.

[21] Matthew D. Zeiler, Marc'Aurelio Ranzato, Rajat Monga, Mark Z. Mao, K. Yang, Quoc Le Viet, Patrick Nguyen, Andrew W. Senior, Vincent Vanhoucke, Jeffrey Dean, and Geoffrey E. Hinton. 2013. On rectified linear units for speech processing. In ICASSP.

[22] Ahmad, M., Aftab, S., Ali, I. (2017). Sentiment analysis of tweets using svm. Int. J. Comput. Appl, 177(5), 25-29.

[23] Yenter, A., Verma, A. (2017, October). Deep CNN-LSTM with combined kernels from multiple branches for IMDb review sentiment analysis. In 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON) (pp. 540-546). IEEE.

[24] Wang, J., Yu, L. C., Lai, K. R., Zhang, X. (2016, August). Dimensional sentiment analysis using a regional CNN-LSTM model. In Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: Short papers) (pp. 225-230).

[25] Huang, Q., Chen, R., Zheng, X., Dong, Z. (2017, August). Deep sentiment representation based on CNN and LSTM. In 2017 international conference on green informatics (ICGI) (pp. 30-33). IEEE.

[26] Behera, R. K., Jena, M., Rath, S. K., Misra, S. (2021). Co-LSTM: Convolutional LSTM model for sentiment analysis in social big data. Information Processing Management, 58(1), 102435.

[27] Parveen, H., Pandey, S. (2016, July). Sentiment analysis on Twitter Data-set using Naive Bayes algorithm. In 2016 2nd international conference on applied and theoretical computing and communication technology (ICATCCT) (pp. 416-419). IEEE.

[28] Abbas, M., Memon, K. A., Jamali, A. A., Memon, S., Ahmed, A. (2019). Multinomial Naive Bayes classification model for sentiment analysis. IJCSNS Int. J. Comput. Sci. Netw. Secur, 19(3), 62.

[29] Ahuja, R., Chug, A., Kohli, S., Gupta, S., Ahuja, P. (2019). The impact of features extraction on the sentiment analysis. Procedia Computer Science, 152, 341-348.

[30] A. Poornima and K. S. Priya, "A Comparative Sentiment Analysis Of Sentence Embedding Using Machine Learning Techniques," 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), 2020, pp. 493-496, doi: 10.1109/ICACCS48705.2020.9074312.