Hindawi
Mobile Information Systems
Volume 2022, Article ID 5430569, 14 pages
https://doi.org/10.1155/2022/5430569

Research Article

Sentimental Analysis of Industry 4.0 Perspectives Using a Graph-Based Bi-LSTM CNN Model

Dhilipkumar Venkatesan,[1] **Senthil Kumar Kannan**,[1] **Muhammad Arif**,[2] **Muhammad Atif**,[3] **and Arulkumaran Ganeshan**[4]

[1]*Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai, India*
[2]*Department of Computer Science, Superior University, Lahore 54600, Pakistan*
[3]*Department of Computer Science and Information Technology, The University of Lahore, Lahore, Pakistan*
[4]*Department of Electrical and Computer Engineering, Bule Hora University, Bule Hora, Ethiopia*

Correspondence should be addressed to Arulkumaran Ganeshan; erarulkumaran@gmail.com

Received 20 June 2022; Revised 9 July 2022; Accepted 18 July 2022; Published 26 August 2022

Academic Editor: Praveen Kumar Reddy Maddikunta

Today's modern society mainly depends on Internet for every fundamental task in their life, like sharing thoughts, education, business, and Industry 4.0, etc. Internet has strengthened the base of society digitally. Searching for the reviews and comments for a particular product from the former or present customers has become compulsory for making decision or a purchase, helping them to make a fair deal of the product by view from social media on Industry 4.0. The increased use of data over Internet has led to rise of many e-commerce websites where people can buy things as per their requirement without even stepping out of their house using analysis of text using natural language techniques in Industry 4.0. The graph-based modelling of sentiment analysis classifier needs to divide the textual data into training dataset and testing dataset. First, preprocessing work is performed to improve the data reliability by removing unwanted information and fixing typing errors. To train the models, the whole dataset is used and to measure the classification performance of the categorized models 10-fold validation technique is being used. The paper proposes a combined hybrid model for sentiment analysis using CNN and independent bidirectional LSTM networks to enhance sentiment knowledge in order to address the issues mentioned for sentiment analysis. The proposed CNN model uses global max-pooling for retrieving context information and to downsample the dimensionality. Lastly, to acquire long term dependencies, a distinctive bidirectional LSTM is used. To emphasise each word's learning ability, parts-of-speech (PoS) are tagged in the LSTM layer. In addition, the regularization techniques, batch normalization, and dropout are used to prevent the overfitting issue. The proposed model is compared with a collaborative classifier with six classifiers and each of them predicts the sentiments separately, and the majority class prediction is taken under consideration. The proposed Bi-LSTM CNN model achieves an accuracy of 98.61% along with PoS tagging of the sentiments.

1. Introduction

For selling products, customer's likes and dislikes are an important aspect for decision making. The user-generated reviews on product or their services have increased drastically in social media. The user-generated feedback or comment in different areas or different context on a specific product or topic has greater influence of relevance on it. This kind of opinions or reviews makes a greater impact in the field of politics, business, and marketing both online and offline [1]. The ease of using the Internet in our day-to-day life has made online data increase enormously; to analyze all the data from the source is a challenging task. In order to predict automatically the sentiments or opinions from the document is not an easy task as opinions are complex, restrained, and sarcastic. Sentiment Analysis (SA) has become a promising stream for the researchers, as vendors and sellers are eager and looking forward for a system that can

Figure 1: Basic block diagram of sentiment analysis.

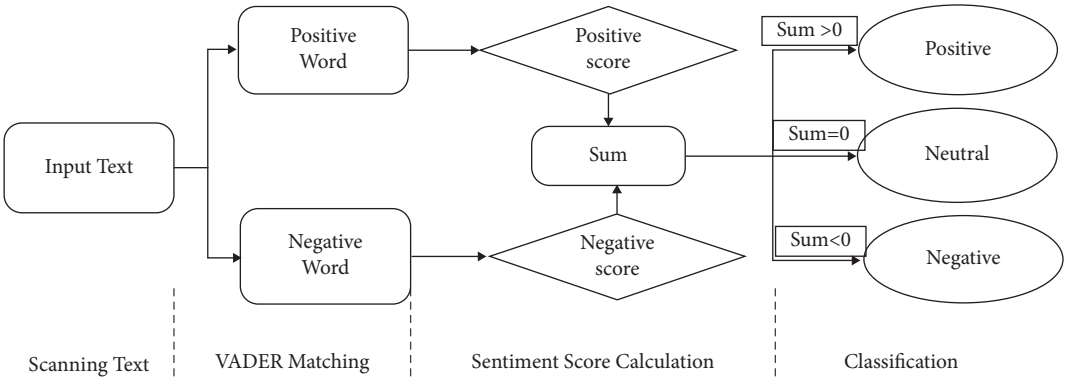

Figure 2: Lexicon-based classification.

automatically predict and analyze consumer sentiments in comments and reviews, rather than browsing and reading the review pages one by one. Figure 1 represents the basic block diagram of sentiment analysis.

2. Theoretical Framework for Sentient Analysis

2.1. Traditional Approach. Traditional approach with lack of ML techniques and untrained model is used for comparison module and varied machine learning techniques/approach are used to predict and classify the sentiment from the textual dataset. Sentiment classification methods are discussed below.

2.2. The Lexicon-Based Approach. The lexicon-based method classifies the sentiments and matches with sentiment word lists, which consists of both positive and negative sentiment word list. Then there is a check for number of positive sentiments to negative sentiments; the lexicon method computes the exact sentiment scores for every class. If the score is the same, then the classifier classifies them as neutral sentiment class. The lexicon-based sentiment classification approach is shown in Figure 2.

2.3. Machine Learning Approach

2.3.1. Support Vector Machine (SVM). Classification of the given data is an easy task in machine learning techniques. Support Vector Machines (SVM) have good theoretical knowledge and high success rate. SVM are made to learn and train in such manner that the resulting function classifies the hidden sample data much precisely. For classifying the classes, SVM makes a hyperplane or decision boundary in the form

$$V^T x + s = 0, \tag{1}$$

where V^T is a vector with d dimensions and s is the scalar. By tuning the parameters V and s of (2), hyperplane is obtained with linearly separable data points x and decision-boundary function is defined as

$$f(X) = \{1 \text{ if } V^T x + s > 0 - 1 \text{ otherwise}\}. \tag{2}$$

The data point functions in (3) depict the decision boundary with label 1 for the points above the boundary and label -1 for the data points below the boundary. By using such type of labelling, (3) computes the decision function for data points above the decision boundary.

$$f(X) = \text{sign}\left(V^T x + s\right), \tag{3}$$

where $V^T x + s > 0$ for all data points above the hyperplane and $V^T x + s < 0$, for all below points shown in Figure 3.

The region between the line of decision boundary and adjacent common support vectors of the classes of the hyperplane needs to be maximized. This maximized gap in hyperplane region is called the Maximum Marginal Hyperplane (MMH). The MMH techniques make the classes distant from the decision boundary, so as to cover the supporting vectors and reduce their classification error. To maximize the boundary, let $V^T x + s = 1$ be a hyperplane, where all support vectors belong to class +1. Also, let $V^T x + s = -1$ be a hyperplane of all below belonging to class −1. Data points x_i are labelled as = (+1/1), by verifying if l_i $(V^T x + s) \geq 1$.

The parallel hyperplanes $V^T x + s = 1$ and $V^T x + s = -1$ of vector V; the distance between the data points are calculated in

$$\frac{|(s-1) - (s+1)|}{\|V\|} = \frac{2}{\|V\|}. \tag{4}$$

The objective of SVM is to maximize $2/\|V\|$ and minimize $2/\|V\| = \sqrt{V^T V}/2$ and final optimization is defined as Min

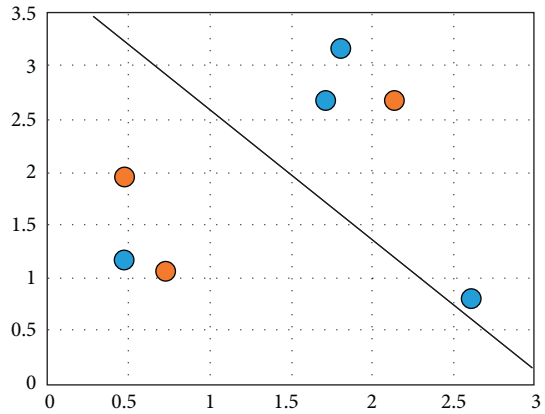

Figure 3: Linearly nonseparable data points.

$V, s\, V^T v/2$, subjected to $l_i\,(V^T xi + s \geq 1)$, for all $i = 1, 2, \ldots, K$. Figure 4 shows the linearly discrete data points in the plane.

Soft-margin extension: In hyperplane not all data points can be linearly separable; in such case those data points are made to be on the other side of the decision as shown in Figure 4. A slack variable $\zeta i \geq 0$ for all x_i is given as

$$\min v, s, \zeta \frac{V^T v}{2} + c \sum_{i=1}^{m} \zeta i. \tag{5}$$

It is subjected to $l_i\,(V^T xi + s) \geq 1 - \zeta i; \zeta i \geq 1$ for all $i = 2, 3 \ldots; m$. Slack variables permit the vector to lie on another side of the decision boundary by reducing it by less than one. Equation (5) make penalty by appending slack variables to the objective function of the data point occurrences.

Nonlinearity: To map the support vectors in higher dimensional space which are not linearly separable in original space, the optimized mapping of the support vectors uses φ, and (6) gives the mapping function.

$$\min v, s, \zeta \frac{V^T v}{2} + c \sum_{i=1}^{m} \zeta i. \tag{6}$$

It is subjected to $l_i\,(V^T \varphi\,(xi) + s) \geq 1 - \zeta i; \zeta i \geq 0$ for all $i = 1; 2 \ldots; m$.

Kernel trick in SVM: Kernel function is used to evaluate the dot products in low dimensions as data points are mapped in higher dimensional space. Equation (7) is used to calculate data points which are intractable in higher dimension and these vectors are used in kernel functions.

$$\varphi\,(x) = K\,(xi; xj) = \varphi\,(xi)|\varphi\,(xj) = (xj)^2. \tag{7}$$

Decision function to classify: After learning the kernel parameters from $\varphi\,(x)$, a test case for vector x is classified by calculating the function $f\,(x)$ in

$$\left(f\,(x) = \text{sign} V^T \varphi\,(x) + s\right). \tag{8}$$

2.3.2. Types of Kernels. Linear kernel: Linearly discrete data points in the original input region; the input is required to map into a high-dimensional space. In such case, linear

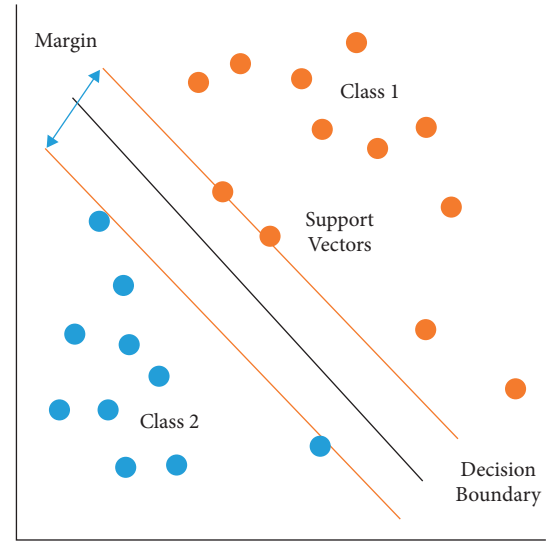

Figure 4: Linearly discrete data points in plane.

kernel is depicted as a dot product of the discrete vectors in actual region. $F\,([x], [y]) = (x, y)$.

Polynomial kernel: polynomial kernel works on p factor, i.e., by $F\,([x], [y]) = (x, y + 1)^p$.

Radial basis function kernel or Gaussian kernel: $F\,(x, y) = \exp^{-\|xi - yi\|^2/2\sigma^2}$ where σ is radius, $\sigma \geq 0.4$.

Sigmoid based kernel: $F\,([x], [y]) = \tanh\,(m\,(([x, y]) + n)$, for some (not every) $m \geq 0$ and $n \geq 0$.

2.3.3. Naïve Bayes Classification Method. Naïve Bayesian theorem: NB theorem is the foundation for the Naïve Bayes classification technique.

The NB theorem is pretty hard to apply the theorem to real world classification. X is the attributes of textual document and tweet data is huge to distribute the attribute which is hard to implement.

For example, there are three attributes, which are the words "Love," "hate," and "cry." So there are 23 possibilities of event X. Moreover, for sentiment classification, high-dimensional attributes are found and they will have 2n possible events of X in thousands or more.

Classification in NB requires only testing the value of $P\,(X|Ci)$ in

$$P\,(X|Ci) = P\,(w_1, w_2, w_3, \ldots\ldots\,(C_i)). \tag{9}$$

Posteriori probability is given in (10) as by assuming the independent attributes

$$P\,(X|Ci) = \prod_{k=1}^{n} Pi\,(w_k|[Ci]), \tag{10}$$

$$P\,((X)|Ci) = P\,(w_1|Ci)xP\,(w_2|Ci)xP\,(w_3|Ci)x \ldots\ldots xP\,(w_n|Ci). \tag{11}$$

Equation (11) shows the calculation of probabilities of the classes as $P\,(w_1|Ci)xP\,(w_2|Ci)xP\,(w_3|Ci)$ $x \ldots\ldots xP\,(w_n|Ci)$ is easy from the training dataset. NB is

used to classify the text data. Similar to the aggregated lexicon-based classifier, the Naïve Bayes classifier approach considers textual document as a bag-of-words. In the proposed work, the sentimental positioning probabilities were calculated based on the Naïve Bayes algorithm for each word performing in the training set and implement distribution matrices of sentiment present in the bag-of-words in training set data.

2.3.4. Decision Tree Algorithm. A decision tree approach is similar to graph representation. The uppermost nodule of tree is the root node. Algorithms used in building a decision tree follow top-down approach. In decision trees, for predicting a class label for a textual data we need to start from the root of the tree. A comparison is carried out between the root attribute and recorded attribute. The result of comparison value is noted by branch and moves to the next node. Similar way comparison is carried out to all internal node until the tree reaches leaf node with predicated class label shown in Figure 5.

2.3.5. The Random Forest Classifier. Robust Random Forest (RF) is a combined effect of multiple decision tree classifiers and the output of the classifier is based on the result of all tree predicted data. Figure 6 depicts that each decision tree is built by using a random subset of the training set with a static probability distribution. When more trees are built, the performance accuracy increases and the problem of overfitting is fixed. For the classification, a training dataset is needed to train the model with two class labels of positive and negative tweets. Finally, the RF learned model produces the predicted results using the test set data.

3. Deep Artificial Neural Network Architectures

This section gives a brief note on neural network architectures. The architecture CNN and LSTM networks are elaborated in subsequent subsections.

3.1. Convolutional Neural Network (CNN). Convolutional Neural Network (CNN) represents the brain neurons cell structure. Figure 7 depicts the CNN architecture with the different layers of the model like dropout; embedding output dimension and filters are in different model learning. The working of CNN is discussed by the author in [1]. The author in [2] discusses the word embedding method using unsupervised learning.

The semantic modelling for sentences using dynamic CNN is discussed by [3]: firstly the input layer and folding embedding layer: the input form is one-dimensional and contains a long sentence to predict. The output of this model will be a 2D tensor. An important argument is that the shape of the input array is served to the neural network. It can be retrieved from tensorflow collections. For example, if a picture has a shape of (32, 32, 3), then it is three-dimensional. 32×32 for pixels and 3 for the RGB colors are present in them. In this paper, the input form is one-dimensional and contains long sentence to predict. The output of this model will be a 2D tensor. The author in [4] discusses simple model that is used for static vectors and is work specific which includes the question classification for sentiment analysis.

The convolutional layer is the most important layer in CNN and it is the first layer of the design. In convolution operation, a new featured positive value matrix with convolved features values is obtained. The new filter has neuron's weights which are rationalized at each training session. The working of convolutional operation is explained in Figure 8, with rolling filter on the input matrix to generate a new featured matrix.

CNN extracts features from the sentences by repeated convolution function with filter rolling on the pixels of matrix. This obtained feature map is represented as f^k where k is the number of classification feature maps obtained using convolution function in (12) with activation function.

$$f_{ij}^{k} = \mathrm{ReLu}\left(\left(W^{k} * x\right)_{ij} + b^{k}\right), \tag{12}$$

$$\mathrm{ReLu}\,(x) = (0.x), \tag{13}$$

where W^k is the weight and b^k is the bias of convolution function filter, and i and j are columns of the matrix. Equation (13) depicts the activation function, Rectified Linear Unit (ReLU) function, to avoid vanishing gradient problem in complex networks. The featured map is shown in Figure 9 which extracts the feature to form a predicted map of the input from previous layer.

A CNN model is designed with a few factors in mind. The first step is to choose the three layers of the convolution layer, which is a 2D tensor from the previous layer. The second and third steps are to use a 24 filter with a 3×3 filter size, and the fourth parameter is padding. Convolution is performed on all pixels by the filter by taking into account the zero sequence to make the matrix length even. ReLU activation function, which uses the fifth parameter max (0, input), is applied to every pixel in the matrix. ReLU converts all complex values below zero to zeroes and saves all complex values above zero. Additionally, nonlinear data in the network are taken into consideration beforehand, allowing CNN to achieve the higher accuracy mentioned by the author [5]. The sixth parameter is that L1 and L2 are the two regularization techniques to overcome the overfitting issue.

While training a neural network the errors are managed to measure the good weights for predicting classes. This is critical such that the vector weights can be set to their maximum point, but this leads the model to overfit. Hence, regularizations like L1 and L2 add penalties to the vector weight values. L1 outlines the values of the weights used and sums them for error calculation, while L2 sums the weights and squares the values. L1 and L2 are similar in their performance so it is easy to implement these techniques in the model after evaluating the best seen using L2.

Global max-pooling and embedding dimension: Max-pooling is a crucial component of CNN networks. The feature maps are subsampled, which can cut down on

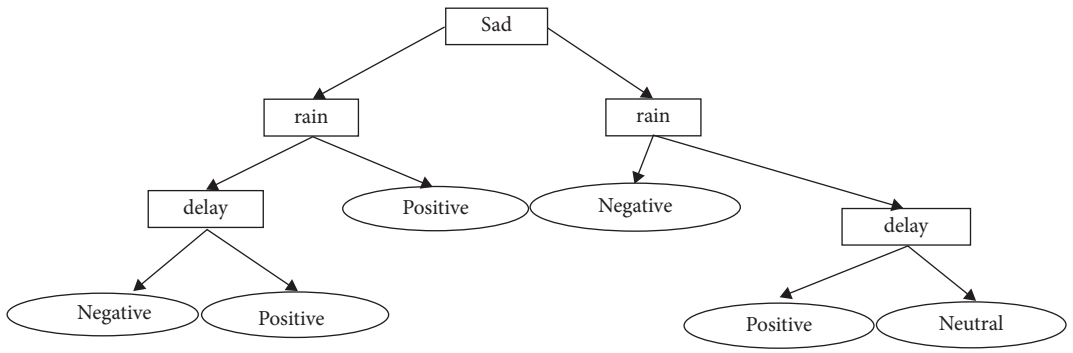

Figure 5: Decision tree-based approach.

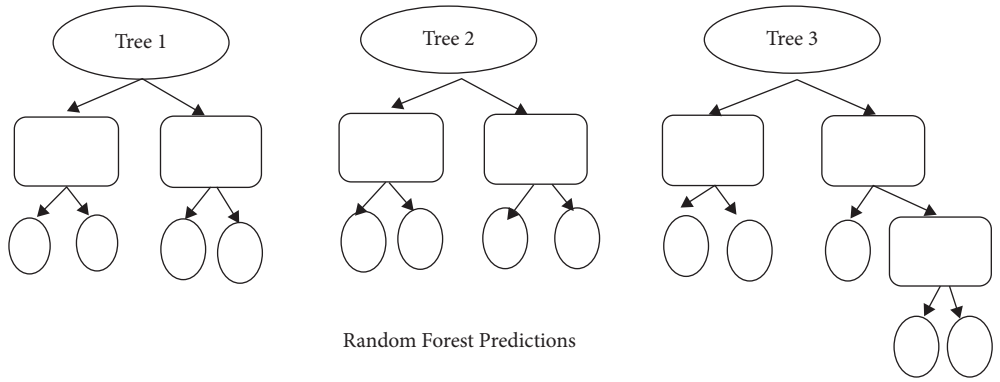

Figure 6: Random Forest approach.

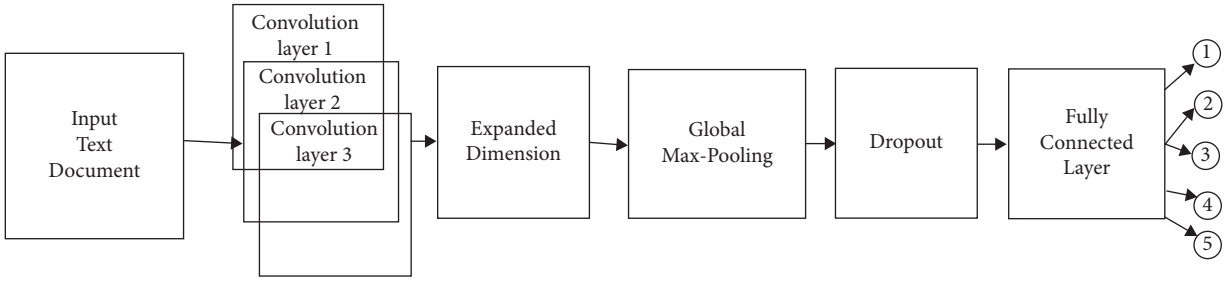

Figure 7: Basic convolution neural network design.

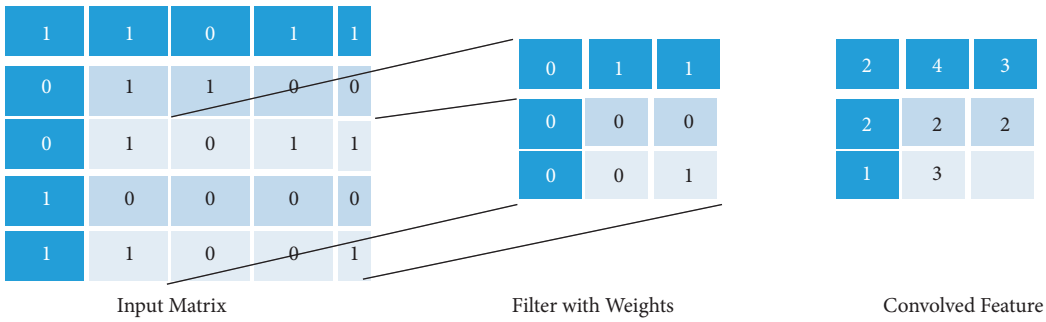

Figure 8: Example of filter rolling over input matrix to generate new matrix with convolved feature vectors.

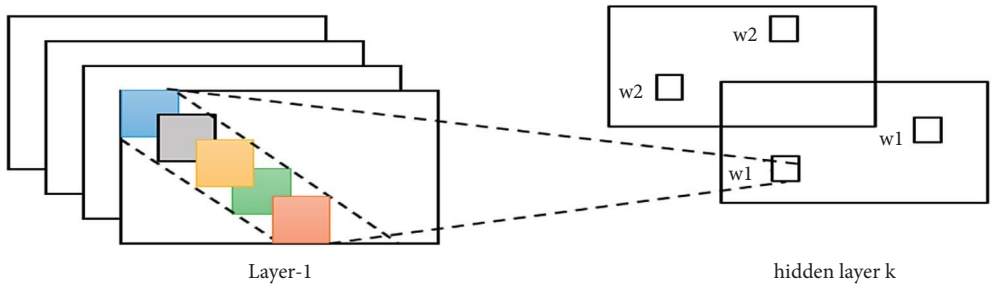

Figure 9: The extracted feature classified map from previous layer feature maps.

computation in the hidden layers. Additionally, low value features are eliminated. The same is true for convolution filter max-pool settings, which is applied to a specified window of neurons over the produced feature maps and only saves the highest value from each filter output. The feature maps are subsampled in this method without the input data being lost.

The convolutional layer returns a featured map tensor and it is fed to the max-pooling function; here in max-pooling layer the tensor is upgraded to a higher tensor. Embedding dimension expands the tensor in one dimension and this new dimension tensor is fed to the global max-pooling layer, where max-pool computation is performed. The computation holds only high value of the input in given filter window. Figure 10 shows how max-pool computation is performed on a sentence. Paper [6] discusses the massive volume of data that is gathered in digital form.

Figure 10 explains the functioning of general max-pooling layer, using 2×2 filter for better understanding; this filter rolls over the matrix and chooses the highest positive number value in corresponding filter. This operation is performed to extract the suitable feature vector from the output obtained by the convolutional layers. Every number depicts a word from the bag-of-words in newly formed embedding layer. The performance result obtained from global max-pooling layer is a 2-dimensional tensor with the very high value and tensor is moved onward to the next layer, i.e., fully connected layer.

Dropout layer: This dropout layer avoids overfitting problem in the neural network model. It is observed that the deep neural network performs good on training set data and acknowledges the uneven continuous variations on the datasets; else layer has increasingly growing accuracy on the training dataset and decreasing accurateness on actual prediction labels. The obtained prediction cannot be realistic to the given data and it may result into worst generalization [7]. The dropout layer computes in a way that overlooks an arbitrary set of neurons in separate training iteration. The biased weights of neurons are modified and specialized to particular features that hold the neighboring neurons to depend on the corresponding feature and finally be fed to the next layer with specified data only. The output of the dropout function shows the obtained results in deep neural network which does not rely on a particular neuron and its weight but has independent neurons which are accomplished by performing improvised prediction on the test set data.

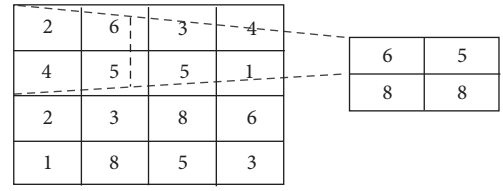

Figure 10: Example for max-pooling maximum value in a 2×2 filter window.

Fully connected layer: FC layer is similar to convolutional layer. The layer is performed as the previous layer tensor is taken as one parameter, the number of labelled classes (Sad, Hate, Neutral, Happy, and Anger) to predict, and a nonlinear activation function called softmax function. With these parameter settings classification of the sentence is calculated. Figure 11 is an overview of fully connected layer that performs convolution and flattening of the featured output.

The softmax layer function is to result in probabilities of a class. It makes the output of each neuron either 1 or 0. The second job done by softmax function is to compute the sum of all vector values of 1. So, now a new vector has a probability of a particular class only. This way the new vector has all probabilities of all the classes to predict [8]. Figure 12 shows an example of a softmax function and how it predicts the probabilities that the vector belongs to one of five classes. The class number one gets a probability of 0.25 where 1 is the total sum of the feature map vectors.

The optimization is the most crucial task in neural network; thus choosing a best optimization algorithm improves the efficiency of the neural network. Adaptive Moment Estimation (Adam) optimization algorithm is used in this model. Adam is mainly used to lower the loss function of the model by minimizing the parameters and weights of the neurons. Adam optimizer is a variant of stochastic gradient descent algorithm which optimizes the gradient descent. Individually batch gradient descent performs and updates whole dataset for each and every epoch as it runs [9]. The gradient identifies the minimum point, the point from which the loss function is minimized. The gradient descent algorithm decreases the loss function to look for the best way and best parameters. Every parameter setting is controlled by the learning rate variable.

The learning rate of CNN model is depicted as 0.001, default value for gradient descent Adam optimizer. However, the learning rate is very much high; then steps become

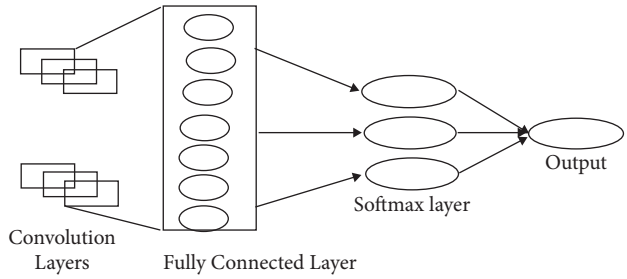

Figure 11: Overview of fully connected layer.

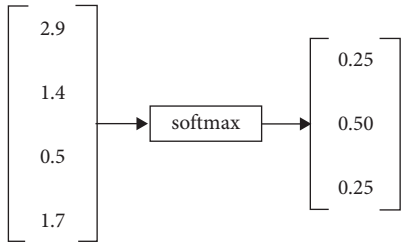

Figure 12: Predicting vector class by softmax.

too large and model results in minimum error rate and further cannot define the parameters. If the step is too small, the rate of epochs to get at minimum is delayed and model learning rate is decreased. Moreover, learning rate improves the accuracy of the model and decreases the loss function [11]. A loss function measures how accurate the model is in predicting the true class of the input data. The loss function is utilized to enhance the weights and parameters to minimize the loss score. This means that the model can predict the actual class of the word, the prediction score being above 0.53 after being processed from the softmax function, but this is not the only prediction form the model rather checks with another label with true value 0.47.

3.2. Long Short Term Memory (LSTM). Recurrent Neural Network (RNN) is a type of variant of neural networks which are connected in sequence to form directed cycle. RNNs use sequential information processing for classification. This sequential processing has a greater advantage in NLP, as sentences are arranged in sequence of words and textual document contains these sentences. Figure 13 illustrates the basic architecture of LSTM cell with different gates which sequence the data in given instance of a time. The recurrent signifies the repeated performance of the parameters of the functioning operations for each component in sequential order. One of the finest qualities of RNNs is that they have the capability to memorize the information in the sequence. However, in practice they are inefficient to learn a long term dependency model and even issue of vanishing gradient problem. Thus, this issue has been overcome by a special design named Long Short Term Memory (LSTM) gates and used primarily in deep learning application and sources.

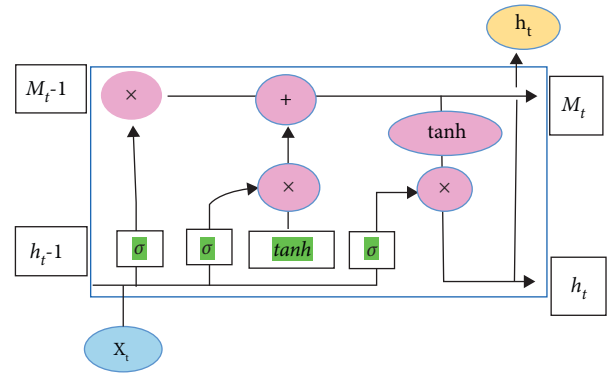

Figure 13: Basic architecture of an LSTM cell.

The author in [11] designed RNN to adhere the problem of vanishing gradient in huge networks. Moreover, LSTM can smoothly handle long term distance dependencies in a given sequence. These characteristics of LSTM make it worth to use in NLP applications where sentences and textual documents are represented in sequential manner. LSTM can be used to memorize only the required part of sequence. This memorizing is activated in network by the use of different type of gates that takes a decision on what (present input and hidden state) to keep in memory.

Equation (14) is used to build a LSTM network model:

$$fg_t = \sigma\left(w_f * x_t + s_t * hd_{t-1} + b_f\right), \tag{14}$$

$$\text{ip}_t = \sigma\left(w_{ip} * x_t + s_{ip} * hd_{t-1} + b_{\text{ip}}\right), \tag{15}$$

$$\widehat{Mt} = \tanh \quad \tanh\left(w_m * x_t + s_m * hd_{t-1} + b_m\right), \tag{16}$$

$$M_t = \text{ip}_t * \widetilde{Mt} + fg_t * M_{t-1}, \tag{17}$$

$$op_t = \sigma\left(w_{op} * x_t + s_{op} * hd_{t-1} + b_{op}\right), \tag{18}$$

$$hd_t = op_t * \tanh \quad \tanh\left(M_t\right), \tag{19}$$

where σ is the activation function, data at instance of time, M_{t-1} is previous memory, Hd_{t-1} is previous output, M_t is present memory, Hd_t is present output, and W_F, W_{Ip}, W_M, W_{Op} S_{Ip}, S_M, S_{Op}, S_T are weights and $B_F, B_{Ip}, B_{M,} B_{Op}$ are bias vectors.

The different gate values are coordinated by the current input and previous output. LSTM operation is to consider the previous information along with present data to make a decision on what information to remember and what to forget. Equation (14) presents the memory cell that is generated by forgetting some section of the present new input x_t . In equation (15), LSTM cell outputs a section of the new memory. Equations (16) and (17) are to calculate memory and final memory of the gate. The output from the aggregated cell represents prediction of the whole sequence. The hidden state value is computed using (19) with sigmoid activation function of the final memory gate output.

Bidirectional LSTM (Bi-LSTM) model is used for incorporating part-of-speech (PoS) in LSTM; the proposed

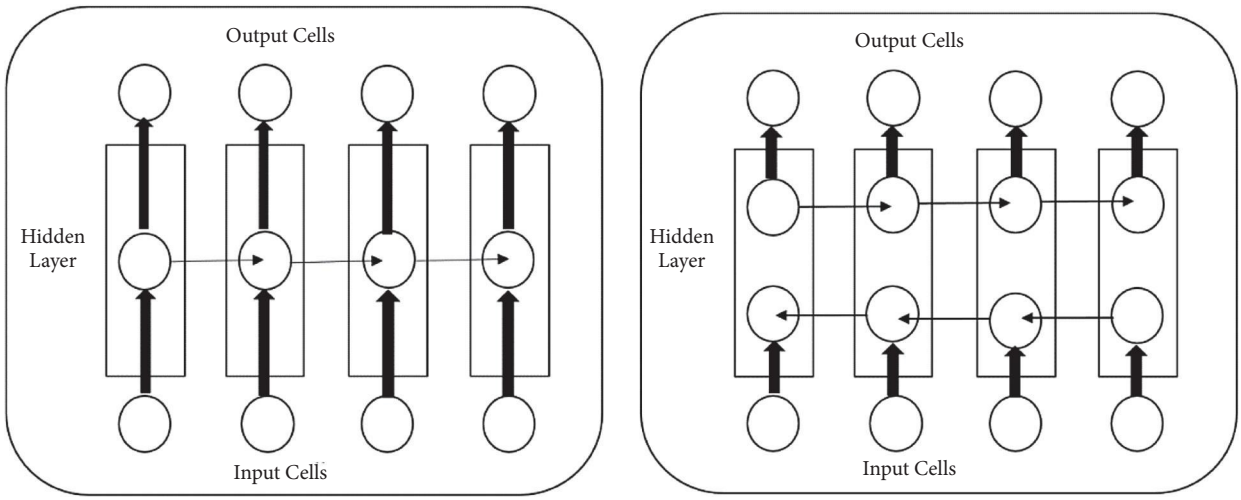

Figure 14: LSTM [11] and Bi-LSTM [13] architectures.

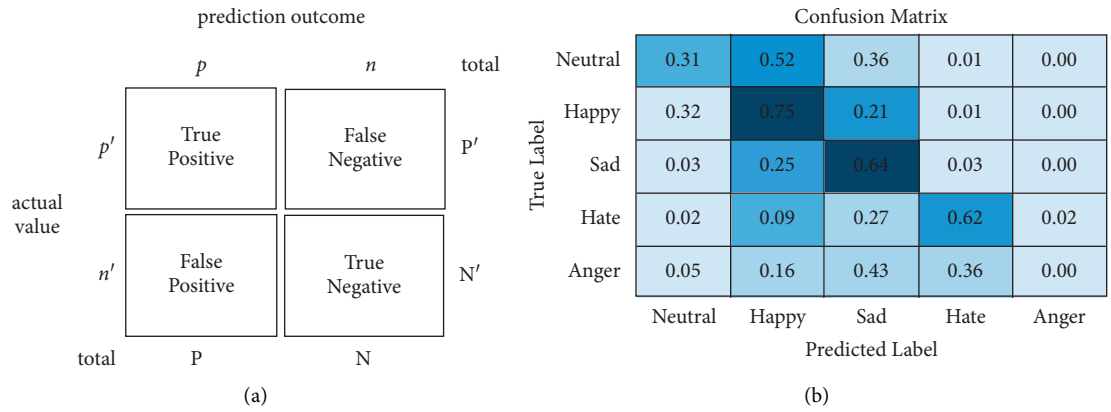

Figure 15: Fundamental block diagram representing the work process of sentiment analysis.

research work implements Bi-LSTM for improvising the results as it considers both past (forward LSTM) and future (backward LSTM) information. In a given sentence a present input feature word is PoS tagged not only with previous input but also with the forthcoming input. This design of bidirectional LSTM is efficiently used to restore the past and future information to recognize the present state [12].

In every Bi-LSTM unit, it consists of two different hidden states for memorizing the past and future information of the present input as shown in Figure 14 [13]. The author in [14] uses a max-pooling layer playing most important roles in classification of texts. The author in [15] uses LSTM model to handle and bring a value or weightage of the whole sentence in a document. In each batch of training set, the hidden states will be updated. Lastly, the two hidden states are combined as one output after the batch training. In the proposed model, the input for Bi-LSTM is a combination of words, subword, and phrases that are processed from the CNNs. In this model, a fixed value of the length of each sentence is called maximum time step. If the maximum time steps are fixed to 50, then there are 50 units of Bi-LSTM sequentially connected to one another. If the length of a sentence is lower than 50, then the values are padded with tokens. The two hidden layers in Bi-LSTM are forward and backward iterations of previous and forthcoming information separately. The two input hidden layers' output is combined to form the final output.

4. Proposed CNN-Bi-LSTM Method

The proposed method is a sentence-level sentiment classification technique that uses the features vector representation model to accurately predict sentiment of tweets. The implementation is broken down into three levels/parts: firstly, preprocessing the dataset, secondly the development of sequential occupied model, and better predictions of sentiments. These parts change the sentence data to a valuable label by determining the sentiment present in tweet of dataset (Figure15).

The given textual documenthave set of N sentences, $L = \text{fL1, L2, ..., SLi}$, each sentence Li, with n words fr1, r2, ..., rni, n 2 K, where K represents a set of real whole

numbers, a sentence is fragmented down into its relevant text/words, and each corresponding word of the sentence is tokenized, hence creating a vocabulary or bag-of words: lt = *f*'l1', 'l2', . . ., 'ln'*i*. Individual word vector or obtained token is allotted a catalog on frequency according to the number of its occurrence in the textual document. The array sentences are then padded to equalize the dimension by filling the corresponding sentence with 0 and returning the padded sentence array, lp.

Following the convolution layers and changing input features in the CNN-Bi-LSTM model are two adaptive dense layers. The proposed model's fully connected layers (FC) use pertinent activation functions to communicate with the input units of the subsequent layer. The input sampled vectors' dimensions are changed using a flatten layer. Each of the corresponding thick layers has two mandatory dropout layers to reduce extraneous units and prevent overfitting. To avoid saturation problem, ReLu activation function is employed in all dense layers. The ReLu function is shown in equation

$$R = \frac{1}{1 + e^{-x}}. \tag{20}$$

It gives an output activation from 0 to 4. The output from LSTM or max-pooling layer of embedding matrix is labelled 0 to 4 which is represented as happy, sad, neutral, anger, and hate sentiments.

The model is finally used to calculate the loss function and obtain the improvised classification. The process of analyzing the given sentiment analysis from a set of sentences to produce a class label for classification is shown and explained in Algorithm 1, where L represents the matrix of class labels for each sentence in test set data (L_ts) and training data (L_tr). L = [l1, l2, . . ., ls] (4.5) where each label is presented in column of the matrix for a given sentence.

4.1. Parameter Settings. The CNN-Bi-LSTM model is trained for fifty epochs; either more epochs lead to overfitting problem or the same result is obtained for more epochs. The problem of overfitting occurs when during the training error gradient decreases at much higher rate than the corresponding testing error parameter; thus there is an enormous gap among the training and testing set data accuracies. This remains observed when epochs are more than fifty; the performance remains stagnant or model overfits. A batch size of proposed model is 32 which has been chosen for twitter dataset that contains small texts in some sentences, shown in Table 1.

4.2. Evaluating Performance Measures. The design evaluation techniques are discussed to compute the quality of prediction and classifications from the models:

Precision: Precision and recall are defined as a recovery of true positive sentences; i.e., the proportion of true positive (TP) in the set of all Positive Predicted Values (PPV). Therefore, precision is the fraction of true positive values to that of combination of true positive and false positive (FP) appropriate to the request. The procedure used for precision is

$$PPV = \frac{TP}{TP + FP}. \tag{21}$$

Recall: Recall is the hit rate (total positive rate) and the formula is

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{22}$$

F-1 Score: A method used to associate precision and recall in their harmonic mean of precision and recall that gives the actual performance of the model. It is given as

$$F1 = \frac{\text{Precision.Recall}}{\text{Precision} + \text{Recall}}. \tag{23}$$

Accuracy: Accuracy is defined as weighted arithmetic mean of precision, recall, and inverse precision. Accuracy is

$$\text{Accuracy} = \frac{\sum (TP + TN)}{TP + FP + TN + FN}. \tag{24}$$

Confusion matrix: A technique for compiling the performance of a classification model: the method visualizes the classification report of the data. In case of the binary classification case, only two possible outputs are classified, i.e., true (positive) and false (negative), whereas for multiclass label output categorical entropy output is used to classify five different classes. Figure 16 shows the confusion matrix for the binary fold and multiclass prediction classification. The prediction result of the experiment in the proposed algorithm calculates multiclass classification confusion matrix.

5. Experiments Results and Discussion

5.1. Results of CNN-Bi-LSTM Approach. The results of different neural network models performance are summarized in Tables 2 and 3 which depict the classification report using recall, precision, and F-measure for the proposed Bi-LSTM CNN model. The three-layer model of CNN shows an improvised accuracy. Improvised performance for CNN is due to the following reasons, as deep CNN acts upon nonlinear activation function (ReLU) that is good for capturing the piecewise elements of nonlinearity, CNN generates n-grams word vectors as sequence and these form a feature vector and are used as input for max-pooling layer. These n-gram word vectors interact with the contextual information of the textual data and progress of the classification rate of the proposed model. Then, CNN customizes varied filters with varying window size such that it can roll over the filters of the pretrained word embeddings and perform N-dimensional convolution on them. As the filter checks on word embedding, various word sequences observations capture the semantic and syntactic category word features in the used filtered bag-of-words which are generated in vocabulary. Several such features are united

Input: x = A set of sentences, S = {S1, S2, . . ., Snd} with n-words, {w1, w2, . . ., wnd}, of dataset
Output: L ϵ {Sad:0 Hate:1 Anger:2 Neutral:3 happy:4}
(1) Assumption: Each document is a English language dataset.
(2) for each instance Si in S do
(3) Level I: Pre-Processing
(4) 1. Tokenizing the word vector
(5) wt = f(Si), Si = {w1, w2, ..., wng}ϵS
(6) Output: wt = {'w'1,' w'2 , ..., 'wn'}
(7) 2. Pad 0 to fix dimensions
(8) wp = g(wt), wt = {'w'1,' w'2 , ..., 'wn'}
(9) Output: count if words in the longest sentence = $n + k$
(10) wp = {'w'1,' w'2, . . ., 'wn', 0, 0, 0, . . .$k$times}
(11) Level II: CNN-BiLSTM Model
(12) if Si belongs to training data then
(13) initialize the data: my_model.fit (L, L_tr)
(14) else
(15) initialize the data: my_model.predict (L, L_ts)

Algorithm 1: Sentiment analysis of twitter data.

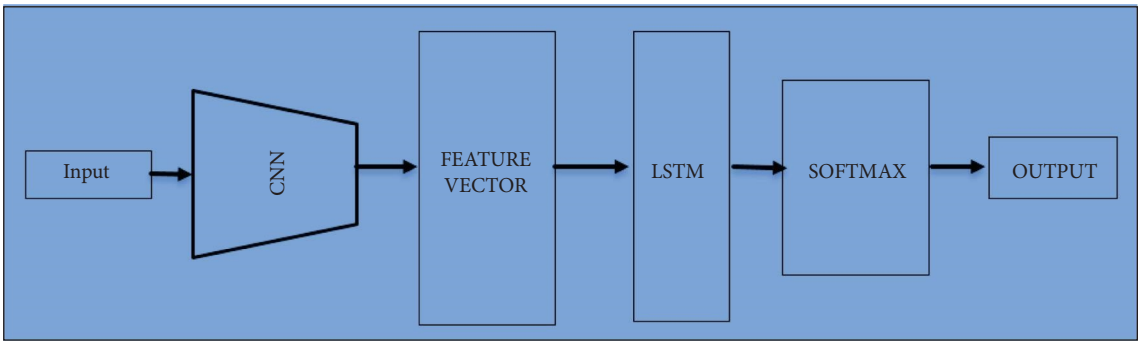

Figure 16: (a) Confusion matrix for binary classification. (b) Confusion matrix for multiclass classification.

Table 1: The hyperparameter feature settings.

Sl no.	Layers	Parameters
1	Embedding	input_dim = 5000, output_dim = 20, input_length = 30
2	SimpleRNN	Neuron units = 100
3	LSTM	Neuron units = 100
4	CNN	nb_filter = 20, filter_length = 3, activation = 'relu'
5	Max-pooling	pool_length = 2
6	Dropout(Layer 1)	Units: 0.5
7	Dropout(Layer 2)	Units: 0.3
8	Dense(Layer 1)	units = 20, activation = 'relu'
9	Dense(Layer 2)	units = 1, activation = 'sigmoid'
10	model.compile()	loss = 'binary_crossentropy', optimizer = 'Adam'
11	model.fit()	batch_size = 32, epochs = 8
12	model.predict()	batch_size = 32, verbose = 1
13	model.evaluate()	verbose = 1

together to extract feature map, and moreover, the pooling layer subsample operations are performed to enhance the performance and accuracy of the obtained results. The vast number of word embeddings generated by GloVe's pretrained embeddings is afterwards explored by the CNN network. Finally, it is noted that CNN outperforms all other deep learning techniques given modelled training data. The other models, however, were less accurate and had an $F-1$

Table 2: Summary of accuracy performance results obtained in predicting the sentiments.

Sr. no.	Model	Accuracy (%)	*F*1-score (%)
1	CNN	91.74	90.36
3	Bi-LSTM	88.59	87.93
4	CNN-Bi-LSTM	94.67	94.38

Table 3: Classification report stating precision, recall, and *F*1-measures values for each class.

Sentiment	Precision	Recall	*F*1-score
Neutral	0.77	0.14	0.87
Happy	0.56	0.73	0.65
Sad	0.64	0.68	0.96
Hate	0.58	0.69	0.97
Anger	0.0	0.0	0.00
Min-avg	0.89	0.84	0.95
Max-avg	0.91	0.81	0.92
Weighted-avg	0.78	0.84	0.69

Table 4: Summary of performance of models with PoS tagging.

Model	Dimension	Accuracy (%)	*F*1-score
Bi-LSTM	200	88.5	0.87
CNN	200	91.7	0.90
Bi-LSTM CNN	200	94.6	0.94
Bi-LSTM CNN + PoS tagging	200	98.6	0.97

score that indicated that the model was not properly reliable over a larger number of iterations. When necessary precision and recall are considered separately, it is shown that CNN mode has high recall score rates but still meets the requirements for precision. It implies greater false positive value in significant cases, so skipping low-level features that are more helpful in accurately forecasting the sentiment is probably the best way to prevent it. One of the main causes for combining CNN with Bi-Directional LSTM is to improve performance when sticking to low features. The entire sentence's textual information is fed to a bidirectional LSTM that has memory cells to hold up to and gates that control and hold the data, process it, or forget it, by assigning weights to the relevant data, using the procured extracted feature vectors that were obtained from the embedding layer of CNN. But if an error persists, a problem can arise, and it might be with the Bi-LSTM. The information that is lost can be prevented in Bi-LSTM and can spread further because it contains cells interconnected to other cells in addition to themselves. Thus, CNN-Bi-LSTM network handles both loss information and loss of efficiency. CNN-Bi-LSTM model performed with high accuracy of 94.67% and *F*-measure score of 94.38% as shown in Table 2. CNN and LSTM models are inevitable to each other, where CNN generates the local word invariants and LSTM is good at modelling the extracted features and helps to hold the memory of the present and pervious words in the sentence.

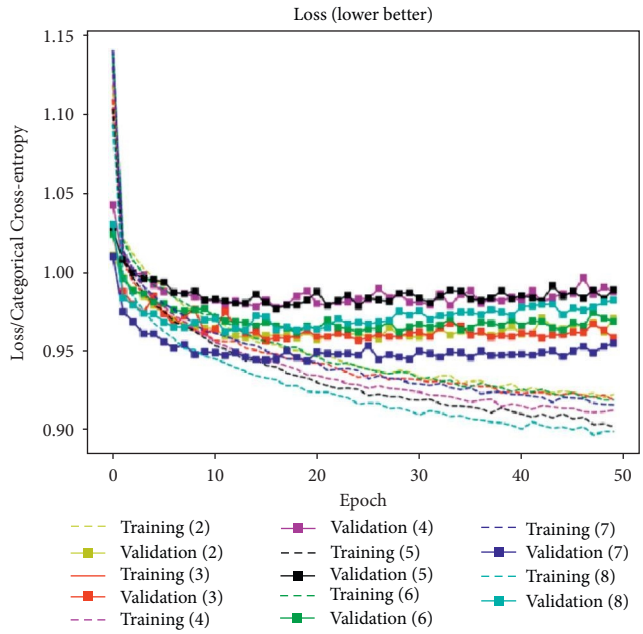

Figure 17: Error loss is predicted for Bi-LSTM CNN model with PoS tag.

Table 5: Classification report precision, recall, and *F*1-measures for Bi-LSTM CNN model with PoS tagging.

Sentiment class	Precision	Recall	*F*1-score
Neutral	0.87	0.14	0.81
Happy	0.64	0.77	0.83
Sad	0.67	0.65	0.65
Hate	0.49	0.61	0.73
Anger	0.0	0.0	0.00
Min-avg	0.99	0.91	0.96
Max-avg	0.93	0.85	0.88
Weighted-avg	0.95	0.98	0.99

5.2. Results of CNN-Bi-LSTM with PoS Tagging Approach. For CNN-Bi-LSTM PoS tag model, fine-grained Penn Tree Bank (PTB) tags are defined. From Table 4 we can observe high *F*1-measure score of 97.6% for SST-1 dataset and precision is average throughout. The reason for high *F*1-Score is elaborated use of PoS tag to the sentiments. However, proposed PoS tag algorithm overcomes the issue of using multiclass labels and increases the overall accuracy compared to the other models implemented. The best result obtained through the proposed algorithm of PoS tag CNN-Bi-LSTM is 98.6%. Table 5 shows the classification report.

5.3. Error Analysis. CNN-Bi-LSTM accuracy model has been benefited with part-of-speech tagging. As the improvised performance of the CNN-Bi-LSTM model is dependent on amount of data used to train them, better performance could be obtained by increasing the data. Moreover, the noise or the error is caused by the input sequence length. The input with equal length has a better

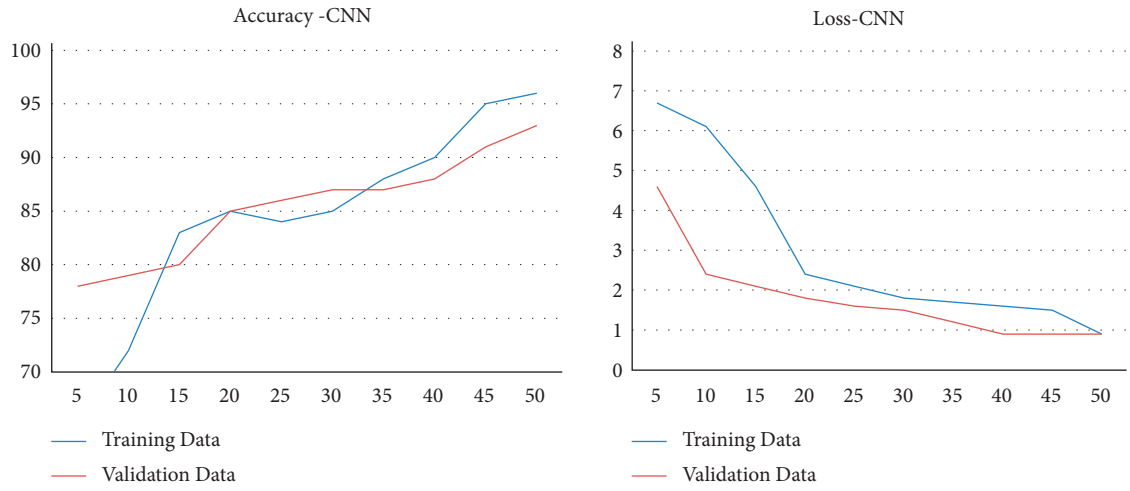

Figure 18: CNN accuracy loss plot.

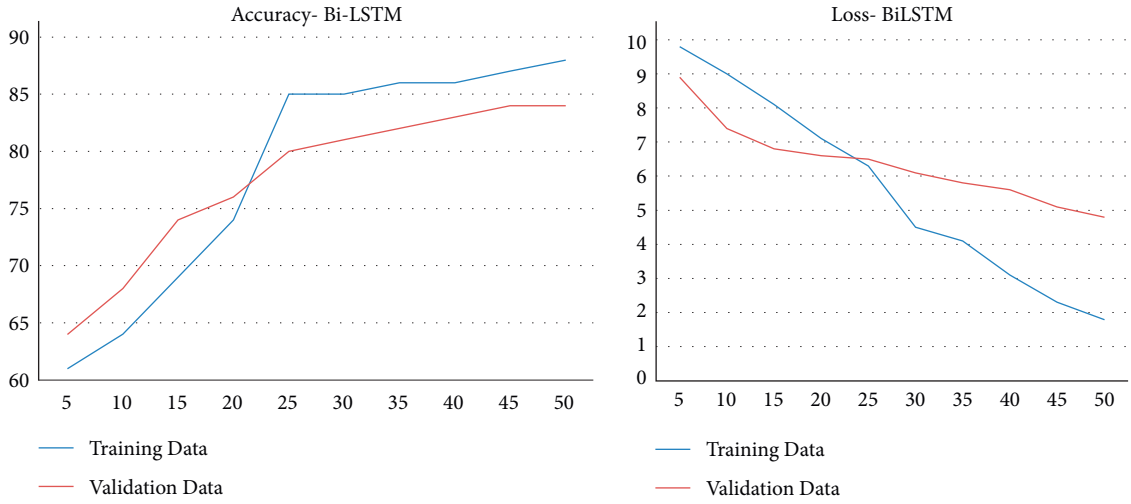

Figure 19: Bi-LSTM accuracy and loss plots.

accuracy; by this approach error can be reduced. The training and validation kernel loss is depicted at different epochs and shown in Figure 17.

5.4. Comparison with Traditional Approaches. Machine learning algorithms with better classification are Support Vector Machines (SVM), Navies Bayes, Decision Tree, Random Forest, and RNN which are widely used to predict the sentiments and perform classification. It is seen that 10-fold validation technique is used to assess the performance of the machine learning classifiers: The Naïve Bayes classifier, the SVM classifier, the C4.5 Decision Tree, the Random Forest classifier, and RNN with LSTM model. The classification accuracy is tabulated below in Table 6. The lexicon-based classifier using VADER attained the lowest accuracy, 66.73%. The significant classification accuracy of the Naïve Bayes model was found out to be 75.38%. The Naïve Bayes Network classifier overtook the lexicon-based classifier. The SVM classifier obtained an overall accuracy of 80.85%, the C4.5 Decision Tree accuracy is 69.16%, the Random Forest classifier attained accuracy of 73.07%, and RNN based LSTM classifier outperformed all the classifiers by gaining the classification accuracy of 89.10%.

5.5. Comparison of Accuracy and Loss. The CNN accuracy and loss function for the designed model using GloVe pretrained word embedding on SST are shown in Figure 17. The loss function is essential for compiling the "categorical-cross entropy" of the designed model. The CNN model gave a good performance accuracy of 91.7% (Figure 18).

LSTM/Bi-LSTM is rather improvised CNN. But they are good enough for classification of the data using memory cell. The Bi-LSTM obtained an accuracy of 88.5% as shown in Figure 19 which is very close to CNN model with GloVe.

This combination of CNN and Bi-LSTM model with PoS tagging feature has achieved significantly improved accuracy of 98.6% as illustrated in Figure 20; this signifies that combining the models can improve performance and it is confirmed that adding LSTM's module to CNN has improved the classification results.

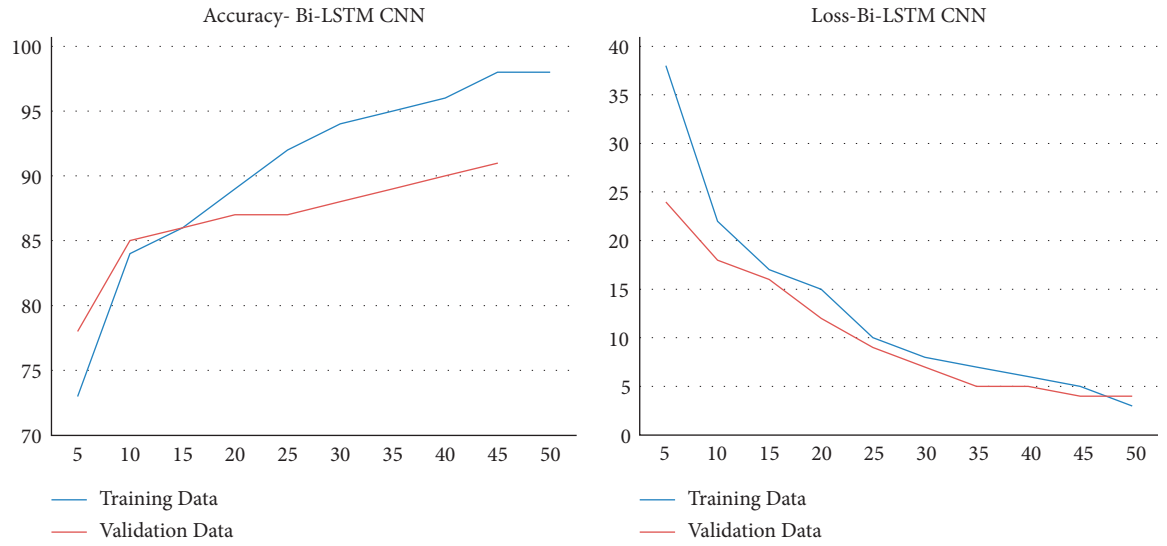

Figure 20: Bi-LSTM CNN accuracy and loss plots.

Table 6: Traditional sentiment classifiers approaches accuracy.

Approaches	Accuracy
VADER-lexicon based	66.73
Naïve Bayes	75.38
Decision Tree	69.16
Random Forest	73.07
Support Vector Machine	80.85
Recurrent Neural Network with LSTM	89.10
Bi-LSTM	88.52
CNN	91.78
Bi-LSTM CNN	94.64
Bi-LSTM CNN + PoS tagging	98.61

It is observed from the plots that outcome of Bi-LSTM helps in attaining the sequence and flow among the data in specific direction which makes the context more understandable, and the CNN is categorized by its capability to extract the features of the given sentence of the textual document; this combination has long way benefited the strengths of each model together.

6. Conclusion

The necessary components involved in addressing the graph-based sentiment analysis with different text representations as well as the different model architectures were discussed. For better understanding the layers of each model, brief outlines were explained, followed by the model architectures with their respective layers and their parameters. Supervised machine learning techniques have been discussed for SSTb-1 dataset. Discussing the machine learning approaches, the advantage and disadvantage points of various approaches have been conversed. Using deep neural network sentence-level sentiment analysis is performed. Different features of neural network are used to extract significant features of the document such that it improves the performance. CNN and RNNs are powerful sentiment analysis techniques. A sequential architecture is implemented for the sentiment analysis for twitter data where CNN, LSTM, and Bi-LSTM are used to create three different sequential models. It is observed from the models that, depending upon the data size and the complexity of dataset, these help for identifying sentiments. The Stanford Standard Tree Bank dataset is used to test the proposed strategy. The experiments were carried out using the GloVe word embedding method. The best result was attained by the Bi-LSTM CNN with PoS tagging model and GloVe word embedding technique, which reached 98.6% test accuracy in the 5 different classes. On the SST dataset, the CNN model got a test accuracy of 96,7%. Additionally, the Bi-LSTM model kept its 89.5 percent test accuracy. These neural network approaches are used in this study work to forecast the emotions. Additionally, there are more social media platforms and review websites than ever before, making it harder to find relevant reviews [10].

Data Availability

Data are available on request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] Y. Goldberg, "Neural network methods for natural language processing," *Synthesis Lectures on Human Language Technologies*, vol. 10, no. 1, pp. 1–309, 2017.

[2] Z. Jianqiang, G. Xiaolin, and Z. Xuejun, "Deep convolution neural networks for Twitter sentiment analysis," *IEEE Access*, vol. 6, pp. 23253–23260, 2018.

[3] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A Convolutional Neural Network for Modelling Sentences," 2014, https://arxiv.org/abs/1404.2188?context=cs.CL.

[4] Y. Kim, "Convolutional Neural Networks for Sentence Classification," 2014, https://arxiv.org/abs/1408.5882.
[5] U. Karn, "An Intuitive Explanation of Convolutional Neural Networks," ujjwalkarn, 2016, https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets.
[6] B. Liu, "Sentiment Analysis and Opinion Mining. Synthesis Lectures on Human Language Technologies," Morgan and Claypool Publishers, San Rafael, 2012.
[7] H. Wu and X. Gu, "Towards dropout training for convolutional neural networks," *Neural Networks Elsevier*, vol. 71, pp. 1–10, 2015.
[8] C. Bear, *Kulbear/deep-learning-nano-foundation*, GitHub, 2017, https://github.com/Kulbear/deep-learning-nano-foundation/wiki/ReLU-and-Softmax-Activation-FunctionsAccessed on:.
[9] S. Ruder, "An Overview of Gradient Descent Optimization Algorithms," 2017, http://sebastianruder.com/optimizing-gradient-descent/.
[10] J. McCaffrey, "Test run - L1 and L2 regularization for machine learning," 2017, https://msdn.microsoft.com/en%20us/magazine/dn904675.aspx.
[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[12] C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith, "Transition-based dependency parsing with stack long short-term memory," *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, vol. 1, 2015.
[13] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, no. 5-6, pp. 602–610, 2005.
[14] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," *Twenty-ninth AAAI Conference on Artificial Intelligence*, vol. 32, 2015.
[15] J. Liu and Y. Zhang, "April. Attention modeling for targeted sentiment," *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, vol. 2, pp. 572–577, 2017.