ENHANCING TEST- CLASSIFICATION PERFOMANCE USING LONG SHORT - TERM MEMOERY NEURAL NETWORKS

Team Members-

K.Narasimha Rao 9921004387@klu.ac.in

R.Gouri rentalagouri2004@gmail.com

T.Kowshik 9921005230@klu.ac.in

C.Prudhvi Raj 9921005196@klu.ac.in

ABSTRACT:

Text classification typically suffers from the fundamental issues of presenting data with a very high dimensions which hinders the model developed because (1) training time grows exponentially with the number of features used and (2) model risk of over fitting increases with increasing number of features since a recurrent structures is ideal for processing long varied texts ,recurrent neural networks (RNNs) are among the model often used designs in natural language processing(NLP) RNN using the long short term memory (LSTM) architecture is one of the deep learning techniques suggest in this study in natural language for English ,one element in the subjects while another element records whether the subject is singular or plural LSTM supplies elements that are intended to be able to record an input characteristic. LSTM will discover these properties throughout the training procedure. The results of this study are most accurate when variable word sequence characteristics are taken into account.

To be able to record an input characteristic. LSTM will discover these properties throughout the training procedure. The results of this study are most accurate when variable word sequence characteristics are taken into account.

Keywords—component, formatting, style, styling, insert (*key words****)***

1.Introduction

Natural Language Processing (NLP) uses text categorization as a key component in processes including sentiment analysis, information search, and document classification. Recently, neural network-based models have gained more and more traction. The application of these models is restricted to very large datasets because, despite their very strong performance in practise, they tend to be quite sluggish both during training and testing.[7]Traditional machine learning techniques including k-Nearest Neighbours, Naive Bayes, Support Vector Machine, and Logistic Regression have been used in text categorization research. Additionally, SVM offers great efficiency and stability traits in comparison to other machine learning classification methods. The dimensions

issue is avoided since the final decision function is only based on a few support vectors When it comes to training with huge datasets, SVM also has certain drawbacks.[3]Convolutional neural networks autoencoders, deep belief networks, and recurrent neural networks (RNN) are a few deep learning techniques that have been employed for text categorization. The recurrent nature of RNN makes it one of the most often utilised architectures in natural language processing since it is ideal for analysing lengthy varied texts. The RNN architecture with the use of the Long Short-Term Memory (LSTM), which effectively increases the memory, is one of the deep learning techniques suggested in this study.[18] LSTM seeks to address the issue that classic RNN has with gradient disappearing and exploding during training. Separate memory cells that can only update and display their information when necessary are stored in LSTM. The LSTM itself uses various processing using a standard RNN model. Another distinction is the presence of a second signal, known as a context or memory cell, that is sent from one time step to the next. Context is a vector with an LSTM network designer-defined number of elements. For example, in natural language processing for English, one element stores the attribute of the subject, while another element records whether the subject is singular or plural. Each element is intended to

record a feature of inputs in these situations. LSTM will discover these characteristics during the training procedure. [9] The several table text styles are offered, however other elements like multi-levelled equations, images, and tables are not required. These components must be made by the formatter using the relevant criteria listed below.

I. II. RELATED WORK

A. *Information Classification*

To determine which label in the text content is the most appropriate is the goal of document categorization. The answer is to represent the document in the designated area and choose the label based on how relevant it is to the test document. Many NLP tasks, such as distributed learning of words, phrases, and documents, parsing, and sentiment classification, have been successfully completed using deep learning based on neural network learning models.[1] In related tasks like sentiment classification and text categorization, sentence representation of distributed learning using neural network models can produce good results with little external

domain expertise The use of improvement of global feature selection scheme (IGFSS), which modifies the final step in the general feature selection scheme to acquire a more representative feature set, is a crucial step in the selection of word features for text classification. The number of classes may even be unrelated, despite the fact that the feature set was constructed using a common word feature selection strategy to represent numerous classes.[11] The most recent methods in neural-networks deep learning are used to produce feature representation for conventional parsing procedures for both English and Chinese. The most recent methods from the literature on deep learning neural networks are used for feature representation for both English and Chinese standard parsing procedures employing feature functions based on LSTM

B.Short-Term Memory with RNN

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For

example, the head margin in this template measures proportionately more than is customary.[2] This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations. Gradient issues might arise when employing an RNN model to investigate long-distance correlation sequentially. Recurrent neural networks' hidden vectors are replaced by memory blocks with gates in LSTM. [12] By learning the proper gating weights, this is theoretically capable of maintaining long-term memory and has shown to be highly helpful in obtaining state-of-the-art for a variety of issues, including voice recognition. In 1997, Hochreiter and Schmid Huber made the long-term memory network (LSTM) proposal in order to overcome this issue of learning long-term dependencies. Separate memory cells are kept in LSTM and are only able to update and show their contents when absolutely necessary. The LSTM common architecture used in this study is depicted in Fig. 1.

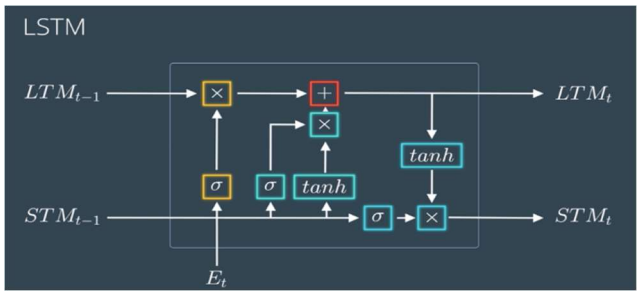


Fig. 1. Architecture of Long Short-Term Memory (LSTM)

The LSTM has also demonstrated to be an RNN model version. Each LSTM unit has a memory cell, and each memory cell represents the status at time t as ct. An arbitrarily lengthy sequence can be processed using a recurrent neural network (RNN), which applies a transition function to the internal hidden state ht vector of the input order. [15]

An LSTM model's foundational cell is seen in Fig.1. Although LSTM and RNN both have a chain-like structure, LSTM regulates the amount of information that may enter each node state using several gates. Below is a step-by-step explanation of the LSTM cell and its gates:

1) *Forget gate*

$$ft = \sigma(Wf.[ht-1, xt]+\text{bf}) \qquad (1)$$

where xt represents input on step time right now, specifies the logistics Sigmond's function, and displays how the items are multiplied. Intuitively,

the input gate regulates how much each unit is updated, the forget gate controls how many memory cells are removed per unit, and the gate outputs govern how much of the internal memory state is exposed.

II. EVALUATION AND OPTIMIZATION

A. *Evaluation.*

This paper used confusion matrix for multi-label classification as shown below:

$$Acc = (TP + TN)/(TP + FN + TN + FP) \quad (2)$$

$$Prec = TP/(FP + TP) \quad (3)$$

$$Recall = TP/(TP + FN) \quad (4)$$

$$F1\ Score = (2 * Prec * Recall)/(Prec + Recall) \quad (5)$$

B. *Optimization*

For deep learning models, there are two different types of optimizers: Adam and RMSProp. Adam was used in this study to train the data.

The Adam Optimizer can handle problems with sparse gradients. For deep learning applications like

Natural Language Processing, a variation of stochastic gradient descent is presently more often used.

$$mt = \beta 1mt - 1 + (1 - \beta 1)gt \qquad (6)$$

$$vt = \beta 2vt - 1 + (1 - \beta 1)g\hat{}2t \qquad (7)$$

Where m and v correspond to the average of the first two gradient moments, and g refers to the gradient on the current mini-batch.

III. EXPERIMENTATION

A. *Dataset*

We analyse a dataset of text messages that have been classified as spam or not in this article. 5,572 messages with binary labels and several other details, including the message's text, date, and time, are included in the dataset, which is titled SPAM text message 20170820 - Data.

The bulk of the messages (86.6%) in the dataset are classified as spam, according to our first analysis of the dataset's distribution. Then, in order to comprehend the traits of spam communications better, we conduct a number of analysis. When compared to non-spam communications, we see that spam messages are often shorter and contain more symbols and digits as shown in Fig.2.

Fig. 2. Dataset of Text Classification

	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

The data visualisation snippet that generates a bar plot of the quantity of instances for every category in a pandas DataFrame column named "Category". The first line of the code uses the pandas [5] 'value_counts()' function to compute the count for each category and saves the result in a variable called 'cnt_pro'. The second line produces a figure that is 12 by 4 inches in size. The 'barplot()' function from the Seaborn library is used in the third line to plot a bar chart, with the x-axis denoting the number of distinct categories in the 'Category' column and the y-axis denoting the frequency of each category. The transparency of the bars is adjusted to 0.8 via the 'alpha' option. The labels for the x and y axes are provided in the fourth and fifth lines, respectively. The x-axis labels are rotated by 90 degrees using the 'rotation' argument in the sixth line. The plot is finally shown in the final line using the'show()' function. [4] The distribution of the categories in the 'Category' column is depicted in the following graphic, which sheds light on the most prevalent

categories and their relative frequencies as shown in Fig.3.

Fig.3. Categorisation of Dataset

B. *Preprocessing*

Using the clean Text function, which accepts a string of text as input, the original text is cleaned up. The programmed first removes any HTML tags from the supplied text using Beautiful Soup. Any instances are then replaced with a single space character. The function then eliminates all instances of the character "x" and changes the entire text to lowercase. The cleaned text is then given back. The column 'Message' in a pandas dataframe df is then subjected to this function. As a result, each element in the dataframe's 'Message' column will receive a call to the clean Text method, and the cleaned text will be returned there. It uses the apply method to apply the clean Text function previously specified to the 'Message' column of the df dataframe. By deleting HTML elements, replacing URLs with the string "," changing the text's case, and eliminating the letter "x," this cleans the content. Using the train_test_split function from scikit-learn with a test size of

0.000001 and random state of 42, it divides the df dataframe into training and test sets. It imports the stop words corpus and the library from nltk. corpus. Stop words includes a collection of typical words that can be eliminated from text since they are unhelpful for NLP tasks. Each row of the training and test data frames is subjected to the tokenize text function using the apply method, which generates a Tagged Document object for each row that contains the list of tokens and their respective categories. It specifies several parameters, such as the maximum number of words to be utilised (max_fatures) and the maximum length of a sequence (MAX_SEQUENCE_LENGTH), for tokenizing the text using the Keras Tokenizer class. The fit_on_texts function is used to generate a Tokenizer object and fit it to the cleaned text in the df dataframe's 'Message' column. This creates a dictionary of original words and gives each word a number index.By utilising the texts_to_sequences method of the Tokenizer object and the pad_sequences function from Kera’s, it turns the cleaned text in the 'Message' column of the df dataframe into a series

of numbers. The length of the resultant X variable, which holds the padded sequences, represents the number of distinct tokens in the text. The text data in a pandas Data Frame’s 'Message' column is preprocessed using the Kera’s Tokenizer and pad sequences methods. The text is first transformed into integer sequences using the Tokenizer. Each word in the text is given a distinct integer ID, and each message's words are represented by a series of these IDs in the sequences.

The text data is fitted with the tokenizer before being converted into sequences using the texts_to_sequences function. The pad sequences function is then utilized to guarantee that each sequence is the same length. This is required because input sequences for neural networks generally have a set length. Any sequences that are greater than the maximum length specified by the maxlen parameter are terminated, while shorter sequences are padded with zeroth number of messages in the Data Frame and the maximum sequence length after padding should be displayed

in the shape of the resultant data tensor, which is printed last.

A matrix created by an embedding a Doc2Vec model. The initial weights for an embedding layer in a neural network will be taken from the embedding matrix, which is a NumPy array. Each word in the Doc2Vec model's vocabulary has its own row in the embedding matrix, and there is also a row for terms that are not part of the model's vocabulary. Each word will be represented as a 20-dimensional vector in the embedding space since there are 20 columns in the matrix. The for loop fills the relevant row in the embedding matrix by iterating over each row of the embedding matrix for the Doc2Vec model. The while loop within the for loop determines if the key_to_index dictionary of the Doc2Vec model has the current row number. The word vector is 1000 characters or fewer in length. The function assigns the word vector to the i-th row of the embedding matrix if both requirements are satisfied. But there are some problems with the execution. First off, if

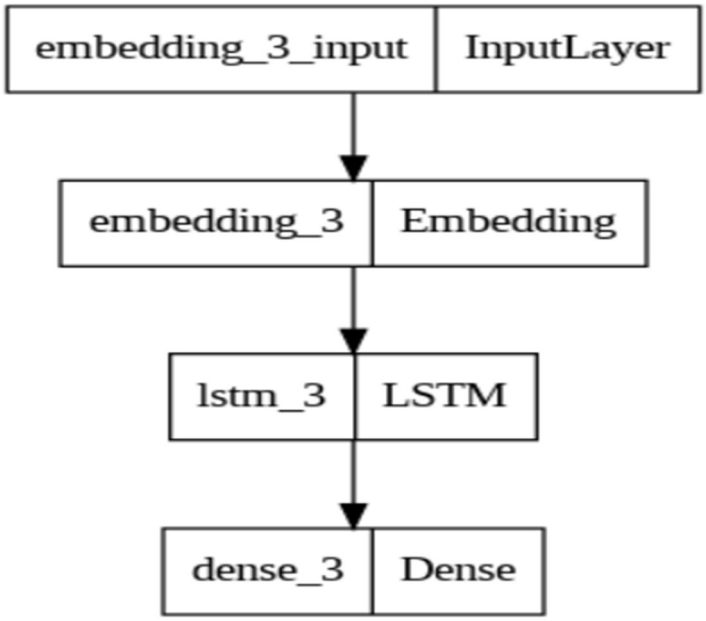
any value of vec meets the criteria i in vec = 1000, the while loop will result in an infinite loop. Second, the commented-out print statements' role is unclear, and they can get in the way of the loop's operation. The model docvecs variable has not been specified in the code, therefore it is unclear to what it references. The cosine similarity of two vectors and the measurement of distance between them based on the trained d2v_model.wv.most_similar method returns the top-n most similar words to the specified word(s). In this instance, the search function is seeking synonyms for the word "cherish."

c. *Model Development*

A binary classification task with the Kera's model. An LSTM layer, an Embedding layer, and a Dense layer make up the model architecture.[6] The LSTM layer analyses the sequence of input vectors to create a fixed-size output vector after the Embedding layer translates the input words to low-dimensional vectors. In order to create a probability distribution over the two classes, the Dense layer then applies a soft max activation

function to the output vector. The Embedding layer is added to the model by the parameter indicates the input dimensionality of the embedding layer, which is the number of distinct words in the vocabulary plus one for unknown words. The embedding vector's size is specified by the 20 parameters. Each input sequence's length is specified by the input length parameter. The trainable parameter defines whether the weights should be modified during training, whereas the weights argument specifies the pre-trained embedding matrix.[17] The input function is designed to divide an input sequence into two sequences, one with the last element removed and the other with the initial element removed and rearranged to have the shape. LSTM layer with 50 units to the model indicates that the layer only provides the results of the most recent time step. For binary classification tasks, line adds a Dense layer to the model with 2 units and SoftMax activation. A overview of the model architecture is printed by the model. The builds the model with the optimizer, loss function, and metrics to be utilised during training. Here, accuracy and binary

cross entropy loss are employed as the assessment metrics with the Adam optimizer as Shown the Fig.4.



The model offered is a sequential model with a distinct three-layer design. An embedding layer with 187240 trainable parameters makes up the top layer. The input sequences are transformed into a fixed size vector representation by this layer. Sequences of numbers are anticipated as the input to this layer. A tensor of the shape (None, 50, 20) is the result of the embedding layer, where 50 is the length of the sequence and 20 is the size of the embedding vector. [14] The model has an LSTM layer with 14200 trainable parameters as its

second layer. The output of the Embedding layer is sent into the LSTM layer, which successively analyses it while taking into consideration the temporal component of the input sequences. A tensor of shape (None, 50), which is equal to the number of units in the LSTM layer of 50, is the result of the LSTM layer. A Dense layer with 102 trainable parameters makes up the third layer of the model. The ultimate output of the model, which is a probability distribution over the two classes, is created by this layer. This layer uses a SoftMax activation function to guarantee that the output is a legitimate probability distribution. The model also features a layer named FC1, which is dense and has 256 trainable parameters. It is followed by a layer with a dropout rate of 0.5. In order to make the model more sophisticated and capture higher-level properties in the input data, the FC1 layer is utilised. To regularize the model and avoid overfitting, utilize the Dropout layer. The model is developed for classification problems with two classes and includes 202,310 trainable parameters in total. It generates a probability distribution across the two classes as an output

using sequences of integers as input. Plot the mdel as shown the Fig.5

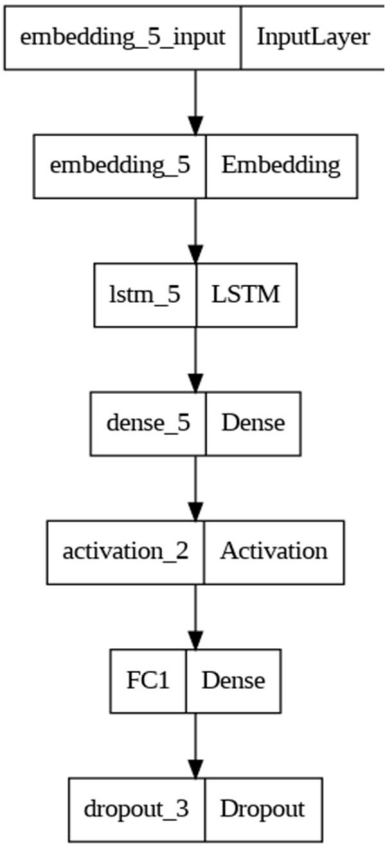


Fig.5. LSTM Model Diagram using Dropout layer

Using the dummies method, it creates a one-hot encoded representation of the 'Category' column in the pandas dataframe 'df' and assigns it to the variable 'Y'. To put it another way, each category in the "Category" column is turned into a separate binary column, and each row is represented by a different set of 0s and 1s across

these columns. The input data 'X' and the one-hot encoded labels 'Y' are then divided into training and testing sets using the 'sklearn. models' election' module. For repeatability, the 'random state' parameter is set to 42 and the 'test size' parameter is set to 0.15, which indicates that 15% of the data will be utilised for testing. The model also comprises a dense layer called FC1 with 256 trainable parameters. A layer with a dropout rate of 0.5 follows it. The FC1 layer is used to enhance the model's complexity and capture higher-level features in the input data. Use the Dropout layer to regularize the model and prevent overfitting.[8]

The model has 202,310 trainable parameters overall and was created for classification tasks with two classes. Using sequences of numbers as input, it produces a probability distribution across the two classes as an output. In general, these graphs can be helpful to track the model's performance throughout training and to spot possible problems like overfitting or underfitting as shown the Fig.6 & Fig.7.

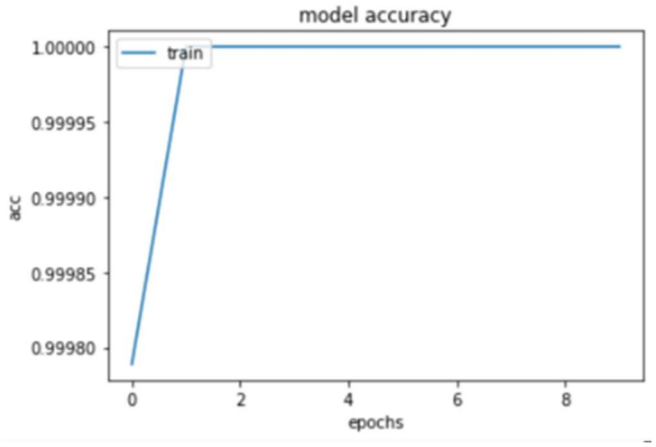


Fig.6. Model Accuracy

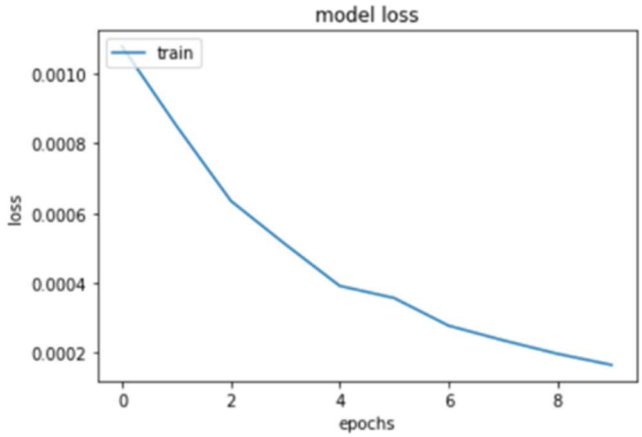


Fig.7. Model Loss

Getting input from the user and train and test the input model using the Trained LSTM Model.

IV. RESULT FOR TEXT CLASSIFICATION

The output appears to be the outcome of executing the training and evaluation phases of a machine learning model. The model was trained using 27 samples or batches, according to the first

line, and it took 0 seconds to complete the training process. The model accurately identified 99.28% of the training data; the training loss was 0.0607, and the training accuracy was 0.9928. The model's accuracy during testing and training comes last.[13] The model properly identified every piece of training data, as shown by the training accuracy of 1.0. With a testing accuracy of 0.99, the model successfully identified 99.28% of the test data. The fact that the model can accurately categories the testing data suggests that it is operating well on the testing dataset and that it has picked up on the patterns in the training dataset. To ensure the model's resilience and generalizability, it's crucial to remember that the performance of the model should be further examined and confirmed using new testing data. A machine learning model's predictions based on a set of test data appear to be the output given. Each row of the predictions corresponds to a single test case, and each column indicates the projected probability for a certain class. The predictions are presented as a 2D array.[16]

The model has produced predictions for each test instance in this example, and the output reveals that the odds for the first class are anticipated to be extremely near to 1.0 (9.9999994e-01), whilst the probabilities for the second class are predicted to be very tiny (on the scale of 1e-12 to 1e-9). The results show that the machine learning model predicts the test cases' membership in the first class with a high degree of confidence. To verify the correctness and dependability of the model, it is crucial to further analyses and assess its performance using the right metrics and testing techniques. [10] By contrasting the projected classifications of a machine learning model with the actual classifications in a testing dataset, the confusion matrix is a table that is used to assess the performance of the model. Visualization of the classification confusion matrix for the LSTM model. This might be a helpful tool for figuring out how well the model performs and finding any patterns or trends in the classifications it makes as shown in Fig.8.

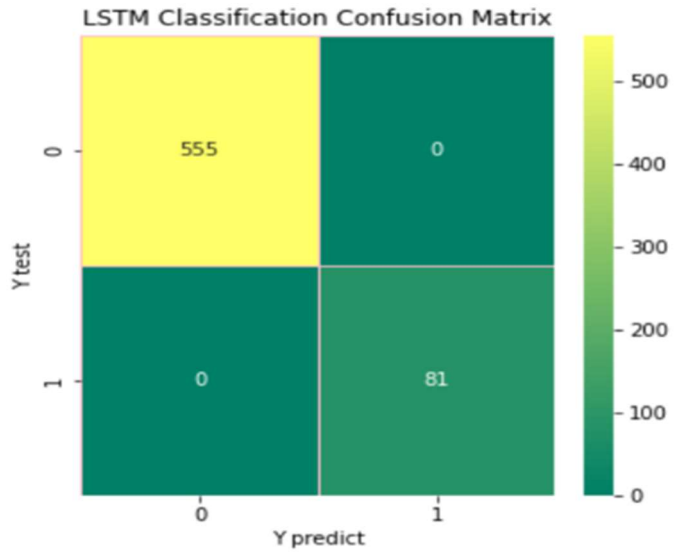


Fig.8. Confusion_Matrix for the LSTM Model

V. COMPARING RESULT FOR DIFFIREENT LSTM MODEL

S.NO	MODEL	ACCURACY
1.	SIMPLE LSTM MODEL	95
2.	LSTM MODEL WITH DROPOUT LAYER	96
3.	LSTM MODEL WITH DENSE LAYER	97
4.	LSTM MODEL DENSE DROPOUT LAYER	99.70

VII.Conclusion

The goal of this study is to categories text using LSTM using different groups of words as input characteristics. The outcomes demonstrate that the suggested model performs well in terms of accuracy. In this instance, the dropout feature with

10.000 words that most frequently appear in combinations of length 250 sequences is the best with 100% accuracy. In other words, the experiment showed that LSTM is effective for problems involving text categorization since LSTM itself takes the form of a long input sequence, indicating that texts may contain lengthy phrases. Future study will utilize a sizable dataset and add LSTM layers to two or three layers. Long Short-Term Memory (LSTM) text classification is a well-liked natural language processing (NLP) method that has demonstrated promising outcomes in a number of applications, including sentiment analysis, topic classification, and spam detection. Recurrent neural networks (RNNs) of the LSTM variety are particularly well suited for text classification tasks because they can efficiently capture long-term dependencies in sequential input. We covered the main ideas of LSTM and how it might be applied to text classification in this article. We also evaluated prior research in the area and offered a thorough evaluation of how well LSTM-based models performed across a range of text classification tasks. Finally, we reviewed the drawbacks and

difficulties of employing LSTM for text classification and suggested several lines of inquiry for further study. A sequence of words or tokens is used as the input to the LSTM network in text classification, and the predicted label or category is the output. From the input sequence, the LSTM network may learn to separate pertinent information and apply them to provide precise predictions. A SoftMax function that gives each potential label a probability can be the network's output layer. The efficiency of LSTM-based models in a variety of text categorization tasks has been shown in numerous research. For instance, it has been demonstrated that LSTM outperforms conventional machine learning methods like Naive Bayes and Support Vector Machines (SVMs) in sentiment analysis. On benchmark datasets like AG's News and 20 Newsgroups, LSTM has been demonstrated to attain state-of-the-art performance in subject classification.

References

[1] L. Li, L. Xiao, W. Jin, H. Zhu, and G. Yang, "Text Classification Based on Word2vec and Convolutional Neural Network," In International Conference on Neural Information Processing. Springer, Cham, 2018, pp. 450-460

[2] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, Ng, Y. Andrew, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," Proceedings of the 2013 conference on empirical methods in natural language processing. 2013, pp. 1631-1642.

[3] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," In Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies. pp. 1480-1489

[4] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for text classification," arXiv preprint arXiv:1606.01781. 2016

[5] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification." arXiv preprint arXiv:1607.01759. 2016.

[6] L. Jiang, C. Li, S. Wang, and L. Zhang, "Deep feature weighting for naive Bayes and its application to text classification," Engineering Applications of Artificial Intelligence, 52, 2016, pp. 26-39.

[7] Darshana Rathnayake, Ashen de Silva, Dasun Puwakdandawa, Lakmal Meegahapola, Archan Misra, Indika Perera, "Jointly Optimizing Sensing Pipelines for Multimodal Mixed Reality Interaction", 2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), pp.309-317, 2020.

[8] Wentao Wang, Chengxu Ye, Ping Yang, Zhikun Miao, "Research on Movie Recommendation Model Based on LSTM and CNN", 2020 5th International Conference on Computational Intelligence and Applications (ICCIA), pp.28-32, 2020.