Bulletin Cognizance Scorn Analysis using NLP

Ms. Preeti Arora
Assistant professor
Department of Computer Science
BPIT,New Delhi,India
preetiarora@bpitindia.com

Ms. Tanisha Madan
Assistant professor
Department of Computer Science
BPIT,New Delhi,India
tanishamadan@bpitindia.com

Ms. Nikita Nijhawan
Assistant professor
Department of Computer Science
BPIT,New Delhi,India
nikitanijhawan@bpitindia.com

Abstract - Sarcasm is one of the sentiments that mean something positive with negative intentions. Nowadays Sarcasm is widely utilized on social platforms such as Twitter, FB, Instagram and comedy etc. Therefore we need a system to detect sarcasm using a effective model. Through the exhaustive study we conclude that the main issue of sarcasm detection by the general expression of sarcasm - "positive sentence address to a negative situation." Our proposed model works effectively on LSTM approach and provides 88% accuracy.

Keywords: LSTM, Sarcasm Detection, Sentiment analysis, SVM(Support Vector Machine), KNN, Naive Bayes, Accuracy

INTRODUCTION

Sentiment analysis is the natural language processing (NLP) and text analysis tools to associate and elicit the feelings of a writer from the text [1]. These emotions can be expressed as either positive, negative, or neutral. Sarcastic writing is common in many online texts[7]. Sarcasm can be identified when a person says the opposite of what they mean [3]. However, due to the restrictions of the unauthenticated language and characters used by News headlines, it is hard to understand the opinions of users. In addition, the appearance of sarcasm in any sentence is even more difficult. Therefore we need a system that Predict the emotions accurately thus, we proposed a model that uses Long short-term memory algorithm Our model can predict whether the sentence is sarcastic or non-sarcastic. We Used Dataset from Kaggle and News Paper headlines are used as Dataset. The main advantages of our model are as follows: (i) Sarcasm detection model can automatically differentiate a sarcastic text from a non-sarcastic one. (ii) We have compared our result with many existing algorithms such as SVM (Support Vector Machine), Naive Bayes, and KNN and Our proposed LSTM algorithm has the highest accuracy among all the pre existing algorithms and provides 88% accuracy.

II. LITERATURE STUDY

Long Short Term Memory (LSTM) is modified version of recurrent neural network and it is better than RNN in terms of memory, which makes task to remember past data in memory easy. LSTM has multiple hidden layers and information passes through every layer, the meaningful information is kept and all the non meaningful information is discarded in every cell. LSTM provide us with larger range of parameters such as learning rates and input/output. Therefore, no need for fine adjustments. The complexity to update each weight is reduced to O(1)with it, Non Decaying Error back propagation.

III. PROPOSED FRAMEWORK

Here, We use an LSTM approach to our proposed model and framework as shown in Fig. 2. Our approach model provides efficient results and improves accuracy when compared with the existing algorithm. Each

stage of the framework is explained as follows: In the first stage, we use the dataset of the news headlines from: https://www.kaggle.com/datasets/rmisra/newsheadlines-dataset-for-sarcasm-Detection. Dataset consists of 28617 headlines, with sarcastic headlines from the Onion and non-sarcastic headlines from HuffPost. The average of each headline is 12 words. The dataset consists of three columns, as shown in fig

1. And those fields are headline(headline of the news article), is_sarcastic(if the headline is sarcastic then the value is 1 otherwise, it is 0), and article link(link of the original news article).

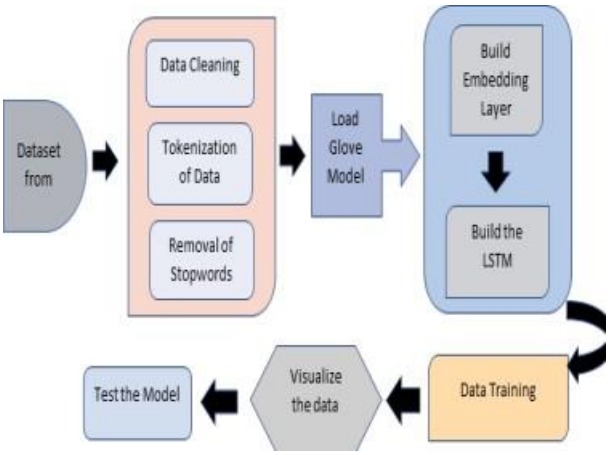

Fig 1.Proposed Framework

In the second stage, preprocessing of data is done after collecting the data. This is an essential phase of our model. Here we clean the data and remove all the unwanted information from the dataset. Such as special characters, urls, hashtags, and unnecessary symbols which do not contribute to the analysis purpose. Pre-processing represents the transformations applied to the data before feeding it to the learning algorithms. Before applying the data preprocessing task, nonEnglish words which are present in the headlines are removed. In figure 3, word cloud representations of the two datasets after applying preprocessing tasks are shown. It is clear from the figures that data preprocessing steps are very essential. Next, we convert the text data into a lower case such as "Post" and "post" are seen as two different words by the program. Therefore, it is important to normalize the case of the words so that every word is in the same case and the model does not process the same word as two different Tokens. After that stop word removal process is applied. Stop words are used more frequently than words in the language but they do not require adding more value to a sentence, hence it is safe to ignore them by removing them from the dataset. They are the most common words as "a", "an", "to", "the", and "is" in the English language. Thus after applying all the required data preprocessing tasks the total number of headlines in the dataset becomes 28617. In the third stage, the tokenizer is built based on the frequency of words occurring in the dataset. This feature converts each headline/sentence to a sequence of 25 words. Extra spaces after the 25 words would be padded. The number of unique tokens 28657 was gathered from the news headlines. The size of the vocab is 28658 words. Vocab size is taken one more than the unique token because if the word is not in the vocab, then it will give that extra space to that word. In the fourth stage, we generate the Embedding matrix. This is a vector value given to each of the unique words selected by the tokenizer in the previous step. The embedding matrix was constructed using pre-trained glove word embedding by Stanford. In the fifth stage, we sequence the embedded and tokenized words. In the sixth stage, the embedding layer is the first layer that is passed to the LSTM model. Then we add 23 more neural networks layer to it. The output is feeding to the LSTM model and then flattening all the LSTM sequences, feeding it. Our model works effectively on the following functions.

1. The sigmoid function helps to increase the accuracy of the model.
2. Binary cross-entropy is used for the loss and this loss function tells how good our

model predicts.

3. Adam optimizer is used which increases the performance of the model.

4. An Accuracy matrix is used for the performance check.

IV. PROPOSED ALGORITHM

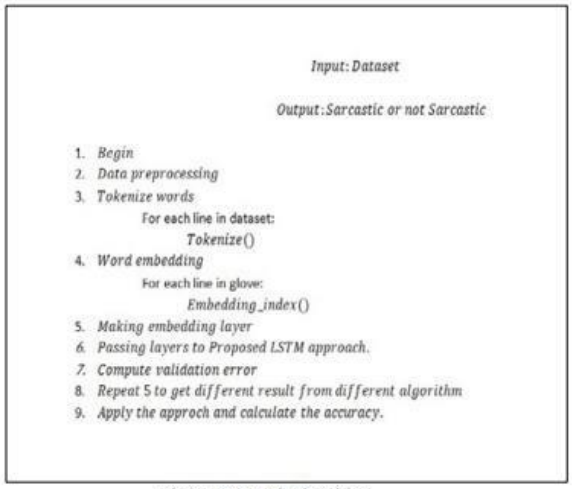

Fig2. Proposed Algorithm

V. RESULT ANALYSIS

For result analysis, news headlines were collected from the KAGGLE. Python, machine learning, and deep learning are used to implement the algorithm. For the efficient results, 24 neural networks were used for each dataset, the algorithm is executed 'n' times, and the results are stored. furthermore, Errors and accuracy are calculated on the basis of training and validating data. Our model uses dropout to reduce the overfitting by repeating the algorithm. We find minimum error from the test data to reduce the overfitting. The performance of the SVM(support vector machine), Naive Bayes, and Knn are not as good as our Proposed LSTM approach. LSTM model gave the best result in comparison to the other algorithm. The confusion matrix for each subset is calculated. It gives the performance measure of the classifier model, and how well the model predicts the correct and incorrect output values with the count. SVM gave an accuracy of 59% approx, Naive Bayes accuracy is 55% approx, and Knn accuracy is 61% approx. After training and testing the model, we see that our model predicts 88.35% accuracy and there is a loss of 0.2732. On plotting the graphs for Model Validated Accuracy and Validate loss, we get the following result for the prediction of the model. Fig 4 and fig 5 show the Model Training and the model validated

accuracy with each epoch of the data set.

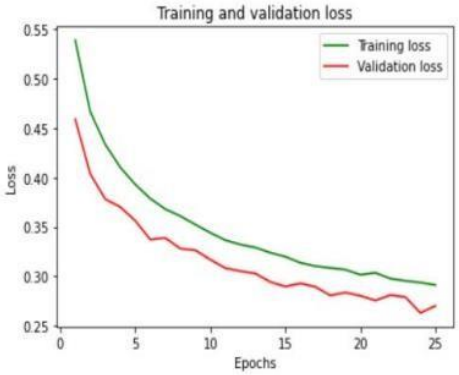

Fig4. Model Loss

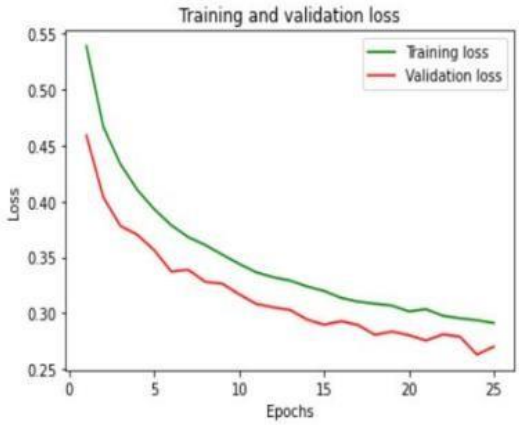

Fig4. Model Loss

VI. CONCLUSION

In this paper, LSTM approach is proposed. Proposed approach provides good accuracy and overcome the problem of Sarcasm Detection. For the adequate results, Binary entropy was used to find the loss and The sigmoid function helps to increase the accuracy of the model. Binary crossentropy is used for the loss and this loss function tells how good our model predicts. Adam optimizer is used which increases the performance of the model. An Accuracy matrix is used for the performance check. Model is trained on the basis of training & validating Data. Overall 88.35% of Accuracy is achieved by the proposed model. Our model is capable to detect sarcastic and nonsarcastic headlines with no context.

REFERENCES

[1] B. Wagh, J. V Shinde, and P. A. Kale, "A Twitter Sentiment Analysis Using NLTK and Machine Learning Techniques," Int. J. Emerg. Res. Manag. Technol., vol. 6, no. 12, pp. 37–44, 2018

[2] C. J. H. E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in Eighth International Conference on Weblogs and Social Media (ICWSM-14). Available at (20/04/16)

[3] Shihab Elbagir and Jing Yang, "Twitter Sentiment Analysis Using Natural Language Toolkit and VADER Sentiment," Proceedings of the International MultiConference of Engineers and Computer Scientists, IMECS 2019, Hong Kong, March 13 15, 2019.

[4] S. B. Mane, Y. Sawant, S. Kazi, and V. Shinde, "Real-Time Sentiment Analysis of Twitter Data Using Hadoop," Int. J. Comput. Sci. Inf. Technol., vol. 5, no. 3, pp. 3098–3100, 2014.

[5] M. Bouazizi and T. Ohtsuki, "A Pattern-Based Approach for MultiClass Sentiment Analysis in Twitter," IEEE Access, vol 3536, no. c, pp. 1–21, 2017.

[6] Wenhong Tian, Yong Zhao, in Optimized Cloud Resource Management and Scheduling, 2015

[7] Medhat, Walaa, Ahmed Hassan, and Hoda Korashy. (2014) "Sentiment analysis algorithms and applications: A survey." Ain Shams engineering journal 5(4): 1093- 1113.

[8] Pang, Bo, and Lillian Lee. (2008) "Opinion mining and sentiment analysis." Foundations and Trends® in Information Retrieval 2(1–2): 1-135.

[9] Kolchyna, Olga, and et al. (2015) "Twitter sentiment analysis: Lexicon method, machine learning method and their combination." arXiv preprint arXiv: 1507.00955

[10] Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. (2002) "Thumbs up? : sentiment classification using machine learning techniques." Proceedings of the ACL-02 conference on Empirical methods in natural language processing. Association for Computational Linguistics, 10: 79-86.

[11] Liu, Bing. (2012) "Sentiment analysis and opinion mining." Synthesis lectures on human language technologies 5(1): 1-167.

[12] https://www.researchgate.net/publication/327097647 _Scopus_Indexed_Journals.

[13] https://www.scribd.com/document/483389700/ahuja 2018-pdf

[14] Khattri, Anupam, and et al. (2015) "Your sentiment precedes you: Using an author’s historical tweets to predict sarcasm." Proceedings of the 6th workshop on computational approaches to subjectivity, sentiment and social media analysis. Lisboa, Portugal, 25-30.

[15] Joshi, Aditya, et al. (2016) "Are word embedding based features useful for sarcasm detection?." Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Austin, Texas, 1006–1011