Jurnal Teknik Informatika Vol. 15 No. 2, 2022 (188-198)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v15i2.28417

JURNAL TEKNIK INFORMATIKA
Homepage : http://journal.uinjkt.ac.id/index.php/ti

Sarcasm Recognition on News Headlines Using Multiple Channel Embedding Attention BLSTM

Azika Syahputra Azwar[1], Suharjito[2]

[1,2]Computer Science Department, BINUS Graduate Program
[1,2]Bina Nusantara University Jakarta
E-mail: [1]azika.azwar001@binus.ac.id, [2]suharjito@binus.edu

Article:
Accepted: October 17, 2022
Revised: September 22, 2022
Issued: November 15, 2022

***Correspondence Address:**
azika.azwar001@binus.ac.id

ABSTRACT

Sarcasm is a statement that conveys an opposing viewpoint via positive or exaggeratedly positive phrases. Due to this intentional ambiguity, sarcasm identification has become one of the important factors in sentiment analysis that make many researchers in natural language processing intensively study sarcasm detection. This research is using multiple channels embedding the attention bidirectional long-short memory (MCEA-BLSTM) model that explored sarcasm detection in news headlines and has different approach from previous research-developed models that lexical, semantic, and pragmatic properties. This research found that multiple channels embedding attention mechanism improve the performance of BLSTM, making it superior to other models. The proposed method achieves 96.64% accuracy with an f-measure of 97%.

Keywords: *Sarcasm detection, BLSTM, Natural Language Processing, News Headline.*

I. INTRODUCTION

Text is the most effective and sophisticated means of expression transmission. One of the types of writing that are common in today's world is sarcasm. Sarcasm is characterized by the use of words and phrases that are not intended to be taken in their literal sense by the speaker or which include a deeper meaning that is assumed to be grasped by the listener [1]. Sarcasm is usually used to express a negative criticism presented with an intentional positive opinion that is often employed to satirize or ridicule. It is easier to identify sarcasm in indirect communication with the interlocutor due to gestures, voice tone, and the context of the dialogue between speakers. The lack of gestures and vocal inflection, which are necessary for understanding sarcasm, is challenging in the text [2]. In general, sarcasm consists of two main factors: the opposite meaning in the text or exaggeration and the contrast of the words conveyed with the actual situation [3]. To explain these two factors, please consider the two examples below:

Example 1. I love the song. Forgive me for the headache.

In this example, there is a contrast in the word's meaning. In example 1, the implications of love and headache are the opposite because we cannot love unpleasant headaches. The difference in the situation with these circumstances creates sarcasm.

Example 2. Her voice is lovely for sleeping.

In this example, there is a contrast in the word's meaning. In example 1, the purpose of wonderful and sleep can mean positive, but when listening to someone else's voice, it is not a good feeling; therefore, the contrast of the situation with this state creates sarcasm.

Today, text data spread on the internet is extensive because it is easily accessible worldwide, one of which is sarcasm text used on social media, news, and other written works.

Text classification is widely used in various applications, including news categorization, sentiment analysis in social media, and chatbots [4]. One often used is news headlines in the form of clickbait, whose content contains news containing satire that can be misinterpreted by readers who do not understand the context. This news can be considered the truth even though not all content includes facts. Therefore, detecting sarcasm, especially in the information, is challenging to improve understanding and detection of sarcasm in the text.

Two methods are often used in the sarcasm research part of NLP, namely CNN and LSTM[5]. This paper presents multiple Channel Embedding Attention BLSTM for sarcasm detection, which uses multi-attention and BLSTM. This paper compares the accuracy of this research with that of various other deep learning models. This approach is used to evaluate sarcasm in news headlines.

The model is crucial for identifying sarcasm because, after the data has been collected, a model approach must be developed that corresponds to the current data. Deep learning is one of the machine learning subfields in which learning algorithms are designed to mimic the structure of human brains. These systems are often referred to by their formal name, Artificial Neural Networks, or ANN. It may also take the shape of a neural network with three or more layers of artificial neural networks, which is its most basic form (ANN). It can learn and adapt to enormous amounts of data and solve complex problems for traditional machine learning algorithms to manage. Deep learning has a wide variety of applications, one of which is the identification of sarcasm used extensively within the processing domain of natural languages (NLP).

Kumar et al. [6] suggested a sarcasm identification method that merges CNN and LSTM with hyperparameters tuning on the Reddit Corpus. To evaluate the efficacy of their algorithms, the researchers assessed a total of 533 million data, which included 1.35 million sarcastic comments posted on Reddit. This technique removes punctuation, regular expressions, and special characters from raw data to standardize the preparation of the data. After that process, the processed data is stemmed to determine the word's original form, tokenized with GloVe and fastText. Then vectors are sent to CNN and LSTM along with 2-8 epochs, training size 10, and 300 embedding dimension, the algorithm considers an extra dropout value that may range from 0.15-0.35. This value is used to increase the accuracy of the classification.

Cai, Yitao, Huiyu Cai, and Xiaojun [7] Wan also use a similar approach called multi-modal sarcasm detection that uses BLSTM to extract image, attribute, and text features representation and merge the representation of these three features to create one feature vector prediction on Twitter dataset.

Misra, Rishabh, and Prahal Arora [8] created a Hybrid Neural Network to detect sarcasm on 29,709 news headlines dataset using BLSTM, attention, and CNN module. This technique removes punctuation, regular expressions, and special characters from raw data to standardize the preparation of the data. After that process, the processed data is stemmed to determine the word's original form, tokenized with GloVe, and the vector is processed using BLSTM. After that, the data weight is calculated using the softmax approach, and the vector is concatenated using the CNN module, which outputs probability sarcastic or non-sarcastic.

Hiai and Shimada [9] proposed a Relational Neural Network (RNN) model. It was based on a Relationship Vector that applied 42,000 twitter data consisting of 21,000 non-sarcastic and 21,000 sarcastic labeled twitter data. This technique removes punctuation, regular expressions, and special characters from raw data to standardize the preparation of the data. After that, the data is stemmed to recreate the original format of the sentence. Following the tokenization of the input using word2vec, the results are processed to generate a role pair relation vector. This allows the relationship between the features to be established. The outcomes of this research are vector-based and were processed utilizing the RNN technique and BLSTM type. The time step is 30 epochs, and the vector dimensions are 200 and 150 hidden. In this instance, the results are vector-based and BLSTM-processed, using historical training data.

Kumar et al. [10]proposed a model called sAtt-BLSTM convNet that uses soft attention, BLSTM, and CNN to detect sarcasm consisting of 40,000 random and 15,000 sarcastic Twitter data. This model was trained on 15,000 sarcastic tweets and 40,000 random (convNet). The data are cleansed during the preparation phase by removing punctuation, regular expressions, and special characters. This helps to ensure that the data is consistent. After that process, the processed data is stemmed to determine the word's original form, tokenized with GloVe. After that, the information is sent to the BLSTM approach, which combines the results of two outputs backward and forward into a single output using a technique known as soft attention. After the information has been stemmed, it is transmitted to the attention layer, which organizes the findings into several categories. This layer is used to modify the nonlinearity of the data produced by the prior approach. Before proceeding to the layer pooling phase, the feature map is reduced at this point by removing all except the most influential features. After that, processed data are sent into a fully linked softmax, which determines the sarcastic or non-sarcastic of each tweet based on its overall sentiment. The representation layer is the very last step in the process, but it is by no means the least important.

Xiong, Tao, et al. [11] constructed the two vectors using self-matching attention and BLSTM, then concatenated them through low-rank bilinear pooling on 28,504 Reddit, 6,326 IAC, and 56,887 Twitter data. The researchers' conclusions were shown with the help of this data collection. It is possible to normalize the data by removing punctuation, regular expressions, and special characters during data preparation. After that process, the processed data is stemmed to determine the word's original form, tokenized with GloVe. After that, the self-matching network evaluates the data to get incongruity information to build a self-matching attention vector and BLSTM vector to retain compositional details. Incoming data is combined using low-rank bilinear pooling to get a probability that accounts for both sarcastic and nonsarcastic data.

Jain, D., Kumar, A., & Garg, G. [12] proposed that bidirectional long short-term memory, the softmax attention layer, and the convolution neural network are the three components that make up the real-time sarcasm detection model. Both sarcastic and nonsarcastic tweets written in Hindi and English were included in their data collection. These tweets were separated into two distinct groups. The removal of preliminaries such as punctuation marks, memorable characters, and common phrases from the data is an activity that must be carried out for this method to succeed. The information is tokenized (in English, GloVe, Hindi, Hindi-SentiWordNet). An English vector is formed once it has been stemmed (returned to its original form). As part of the evaluation of semantic closeness between each word and its function in the interrogative sentence, the BLSTM algorithm sends its results to the soft attention layer for semantic correspondence by estimating the degree of similarity between each word and its function in the interrogative sentence and mixing the English and Hindi vector outputs of the layer then the convolution process of the layer creates a feature map. During the pooling layer procedure, the feature map is decreased in size to incorporate just the required properties. Calculating the chance of each word in the tweets and classifying them as either sarcastic or nonsarcastic, respectively, involves the use of a fully linked softmax. When a cycle is completed, the outputs from that cycle are transferred to the representation layer stages of the current process.

Mandal and Mahto improved their performance in identifying sarcasm by combining the CNN-LSTM model technique with word embedding [13] and applying a carefully classified dataset of 11.725 sarcastic and 14,984 non-sarcastic data headlines. The data are cleaned during the pretreatment step by removing any punctuation marks, regular expressions, and special characters. A word-embedding dictionary containing 10,000 entries from the data is used to tokenize and stem a tokenized word. This enables the tokenized term to be reverted to its most basic form. After that, the processing vector is applied to a one-dimensional convolution layer with 32 filters and seven different kernel sizes, which ultimately results in a filter capable of performing seven-word combination filtering. The 1-D top pooling layer will get all of the results in this instance. By the grain with the highest value, a single output will be generated by integrating the outcomes of each grain into a single production. The results are then reprocessed in the 1-D convolution layer, followed by the CNN-LSTM method, which has a delay of 0.001 seconds. In the last stage of the technique, an evaluation of the significance of the results is carried out using a binary cross-entropy loss function.

Kumar et al. [14] improved the accuracy of sarcasm recognition by training their BLSTM-based Multi-Head Attention method on a dataset of 111,914 satirical and 173,003 non-satirical Reddit comments. When testing their algorithms, the researchers looked at a dataset that included both sarcastic and non-sarcastic comments from Reddit, totaling 110,914 and 173,003, respectively. This technique removes punctuation, regular expressions, and special characters from the data to standardize the data's preparation. The data is then stemmed to recover the phrase in its original form, tokenized via GloVe, and then reconstructed utilizing the results. An additional dropout value of 0.5 is included in the computation.

Using BLSTM with 100 vector dimensions and 100 hidden units, 100 vector dimensions and 100 hidden units are used to process the data. The results are then examined at the Sentence Level Multi-Head Attention Layer, where they are used to determine the importance value of each word based on the semantic component that is being used in the process. In the third stage, Auxiliary Characteristics Concatenation, the findings are processed again by obtaining the original data and concatenating it with the previously derived semantic, sentiment, and punctuation characteristics. The output of the processing layer is delivered to the softmax layer, which then assesses the probability of each word and categorizes the data as being sarcastic or not to generate a new representation of the information. The following contributions are made by this research, which uses an end-to-end network consisting of the phases of the model given below: Preprocessing, BLSTM, word embedding, token vectorization, Attention Layer, Relu Layer, Pooling, and Representation Layer are examples of what is feasible. Our results are much superior to those of the other two deep learning methods.

II. METHODOLOGY

1) Dataset

We used the News Headline dataset [8], which separated headlines in the news into sarcastic and non-sarcastic categories. The dataset includes 56.418 news headlines, of which 25.846 include sarcasm and 30.752 do not contain sarcasm. We divided our dataset into a training set consisting of 45.352 news headlines and a testing set composed of 11.066 news headlines. Training required the utilization of eighty percent of the dataset, whereas testing required just twenty percent. The classifier was trained, and its parameters were optimized with the help of the datasets used for training. On the other hand, the test dataset is one that the model does not have access to, and its only purpose is to evaluate the quality of the trained model via testing. In addition, we attempted to divide the data into 70 percent for training and 30 percent for testing; nonetheless, it produced the same results for us, 97.84 percent.

2) Preprocessing

Before the data are sent to the input layer, they go through a phase known as preprocessing, which prepares the data for feature extraction [15].
These processes were carried out in their entirety:

a. We are transforming a whole document's text into a standard format (in this case, lowercase).
b. Punctuation marks are eliminated from the text, and the content is tokenized.
c. It entails removing unnecessary words and phrases from the body of the text, such as "or", "and", "with", "he", "she" and other phrases.
d. Stemming is the process of restoring a word to its original form.

3) Parameter Setting

When trying to get outcomes with a high level of performance, it is vital to choose the most proper parameters. The validation data are put to use to get the best results that may be achieved by adjusting the parameters presented in the table that can be found below.

Table 1. Parameter List of Proposed Model

Parameter	Value
Word Embedding	GloVe and fastText
Regularization	Dropout Operation
Activation Function	ReLu
Dimension of Embedding	300
Hidden Units	64
Batch Size	128
Dropouts	0.2, 0.5
Learning Rate	0.005
Epochs	100

4) Proposed Multiple Channel Embedding Attention BLSTM

The proposed model comprises six layers, which are referred to as the input, embedding, BLSTM, ReLu, attention, max pooling, concatenation, and representation layer as representation layer. The design of the multi-layer technique that has been proposed may be seen in Figure 1. In the following parts of this article, the specifics of each layer are broken down and discussed in further depth.

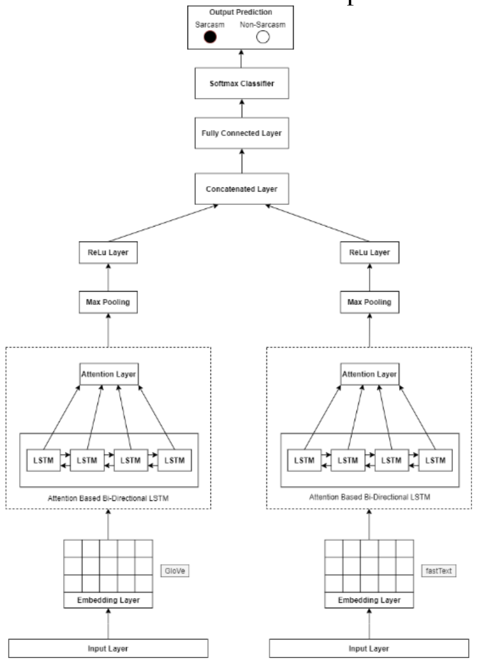

Figure 1. Proposed model architecture Multiple Channel Embedding Attention BLSTM

1. Input Layer
After completing the preprocessing step, the headline is sent to the input layer of the processing system. This process is linked to the embedding responsible for the generation of word embeddings and does so with the assistance of GloVe and fastText, respectively.

2. Embedding Layer
Through lookup table encoding, the data transform into real-valued vectors by embedding layer to make subsequent processing more straightforward. Word embeddings, which provide a visual representation of the word, help learn word representations. Word embeddings may be used to extract satirical features from the text, and this approach has only just been demonstrated to be successful in recognizing sarcasm [16]. To create word embeddings and construct a word vector table, the applications GloVe and fastText are used. Both the GloVe and fastText models are count-based and use feature vectors to represent words. Because of the use of feature vectors that make GloVe and fastText embedding able to define individual words accurately. Counting the number of times two phrases are employed together inside the same linguistic context is one of the steps that this log-bilinear model takes to figure out the link between two terms. Because of this, our model can assist by modeling the tokenized words that are included within every news headline to the expression vector tables that correspond to those words by utilizing the information contained within each news headline. In other words, our model can do this by using the information included within each news headline. To successfully combine the feature vector matrix, it is necessary to conduct suitable padding. They suppose that Z is the total number of tweets that have been provided and that there is at least one tweet X that contains t tokens. GloVe and fastText are the tools used throughout the process of building a unidirectional vector representation with a dimension of d within the unidirectional word representations. As a direct result, every t in X has been mapped to the proper V in the appropriate V-space for each Z. Each X was presented as a word

embedding vector, which equals the concatenated direct result of the mapping described before (E). The feature vector matrix may be represented by the formula given below as a direct result.

$$F = Z + X + V + E \qquad (1)$$

Since C has evolved into a concatenation operator for vectors, the amount of text in each instance may change in length from one occurrence to the next. This may cause the size of the reader to change. To standardize the feature word vector of news headlines, the threshold value is chosen by picking the headline from the available corpus that is the longest in length. This ensures that the standard is applied consistently across all headlines. The bulk of the time, these are done to adjust the size of the headline news matrix shown. As a direct result of this, no padding was added to any of the news headlines under the allotted length. After that, this matrix was sent into the BLSTM layer as an input, denoted by the letter F, where it transformed.

3. BLSTM Layer

The type of RNNs known as Long Short-Term Memory (LSTM), which has three additional gates, is often considered the most effective. LSTMs have been used in various text-mining applications, including sentiment analysis [17], phrase classification [18], and other text-mining applications. As proven in [19], when employed throughout a lengthy text, LSTM training is unstable and underperforms usual linear predictors, resulting in a worse overall performance. The training and testing of LSTMs also take up a significant amount of time and money for such a big text. It is shown in [20] how to stabilize LSTM training via pre-training LSTMs as sequenced auto-encoders or recurrent feature extraction methods before commencing the training. This should be done before starting the actual exercise. This should be done before beginning the training. We can circumvent the previously mentioned problem by using LSTMs for label sequence predictions. This is possible because a label sequence is often shorter than a document. Within the parameters of this discussion, the term "label sequence" refers to the operation of applying successive labels to a section of text. The traditional LSTM is used, even though there are numerous different LSTMs. An additional layer of word embedding is used to distinguish the various brands from one another.

The recurrent hidden layer of the LSTM, a kind of recurrent neural network, uses memory blocks [20]. The LSTM uses memory blocks as its special units. Every memory cell has three gates: first input gate, second output gate, and third forget gate. These gates allow memory cells to store information. LSTM hidden layer circuit is referred to as the LSTM cell, which is a name used to refer to this layer. It is possible to examine long-term dependencies using the LSTM if every memory cell is defined, including a collection of gates d, where d will be the LSTM hidden state [20]. This makes it possible to view the LSTM. This is accomplished by charting the results once the visualization process is complete.

Word embedding E represents the words in the news headline F so that they are not reliant on the other words in the headline. This layer generates a new representation for each instance of the news headline by summing contextual information from both directions. LSTM (2) and LSTM (3) are used to form the bidirectional LSTM, which reads the comment from x_n to x_1 in the following manner:

$$\overrightarrow{h_t} = \overrightarrow{LSTM}\,(w_t, \overrightarrow{h_{t-1}}) \qquad (2)$$

$$\overleftarrow{h_t} = \overleftarrow{LSTM}\,(w_t, \overleftarrow{h_{t-1}}) \qquad (3)$$

To derive the hidden state representation h_t for a given word, we first concatenate the forward and backward hidden states for that word's x_t representation.

Then, using (4) [21], h is calculated:

$$h_i = \overrightarrow{h_t} \odot \overleftarrow{h_t} \qquad (4)$$

It is a concatenation function, and it is used to mix the two outcomes. The variable hi represents the $\odot$ output of the i-h_t word. There are several merging mechanisms available for combining the BLSTM layers' results. The default operation is concatenation, multiplication, total, and finally average. h_t is the forward layer sequence output that is continuously generated within a positive series of inputs starting at time t-n and continuing until time t-1. The sequence begins at the moment t-n and continues until t-1. It is calculated starting at the moment t-n and continuing until t-1 using input in an opposite manner that creates outcome sequences of the backward layer called ht. With this strategy, information may be gathered more easily from the entire phrase that surrounds each word x_t. $H \epsilon \mathbb{R}^{N\times 2p}$ is a notation representing all of the hidden states of the terms x_t.

Where p indicates the length of h in both the forward and backward directions.

$$H = (h_1, h_2, h_3, \dots, h_n) \qquad (5)$$

4. Attention Layer
When it comes to the work of analyzing text, the attention model presents an illustration of the relationship among words and phrases in a text with the output result. This model will be first put to use to do text processing jobs. The standard attention model is simplified in an obvious way by using the feed-forward attention mechanism [22], which was used in this research. This method streamlines the process by creating a single vector c from the entire sequence shown in the accompanying illustration:

$$a_{i,t} = \frac{\exp\,(e_{i,t})}{\sum_{k=1}^{T} \exp\,(e_{i,k})} \qquad (8)$$

Where a is a learning function, which at this stage is solely being dictated by the constant ht, and calculating a responsive average value of a states h and then constructing an embedded layer c of the input sequence as a result of this computation becomes a way to solve the problem concerning attention mechanism that was explained in the formula that was presented before this one. When the following phrases are combined, the final sentence-pair form that is utilized for classifying is formed, which results in the following:

$$h^* = \tanh(c) \qquad (9)$$

5. Max Pooling
Data processed in the previous step will be next subjected to a maximum pooling treatment in a single dimension. This layer selects the most significant feasible value visible in each kernel. It utilizes it as its input to combine the kernel sizes of the information toward a single output in the end. This is done so that the result will be as uniform as possible. This allows the kernel sizes to be merged into a single output at the end. The practice of pooling is done to eliminate any possibility of overfitting taking place. Because of this, it will be feasible to add additional layers to the suggested architecture of the neural network, which will, in turn, make it possible for the neural network to extract characteristics at a higher level.

$$f([\overrightarrow{h_t}, \overleftarrow{h_t}]) = \arg Max\left([\overrightarrow{h_t}, \overleftarrow{h_t}]\right) \quad (10)$$

6. ReLu Layer
ReLU Layer [23] is responsible for addressing the non-linearity of the model. It begins by producing a corrected feature map, then is sent into the concatenated layer. This causes the two matrices, Glove and FastText, to be combined into a single feature map.

$$f = [f_a, f_t] \qquad (11)$$

7. Concatenated Layer
The data that has been processed by two ReLu Layers that utilized a different word embedding strategy will be concatenated together. Because of this, a more extensive matrix of Sall will be formed, which will then be included in the building of an even more extensive matrix of Sall.

$$S_{all} = [f_g, f_s] \qquad (12)$$

These outcomes are then transmitted to the ultimately linked layer. The vector outcome of the ReLu operation that uses GloVe word embedding will be called f_g, and the ReLu operation that uses fastText word embedding will be called f_s.

8. Representation Layer
The activation function for softmax is connected and a component of the output layer. This feature map is then passed on as an output to the integrated softmax layer. Then the news headline will be evaluated as sarcastic or nonsarcastic depending on the results of an assessment of the likelihood of each word in the output. P_i is the softmax layer's output vector, and it can be seen when the coating is evaluated (13).

$$P_i = Softmax(W_c \cdot f + b_c) \quad (13)$$

b_c is the value that reflects the offset value, W_c is the value that specifies the weight matrix, and Pi is the number that shows the possibility that the input is sarcastic.

III. RESULTS AND DISCUSSION

The empirical research has been split into two stages to analyze the results: setting up the parameters for the proposed model and (ii) doing classification accuracy comparisons using numerous baselines.

3.1 Training and Testing Result

The proposed model was validated by applying it to a dataset consisting of 56,418 news items, which enables sarcasm detection.

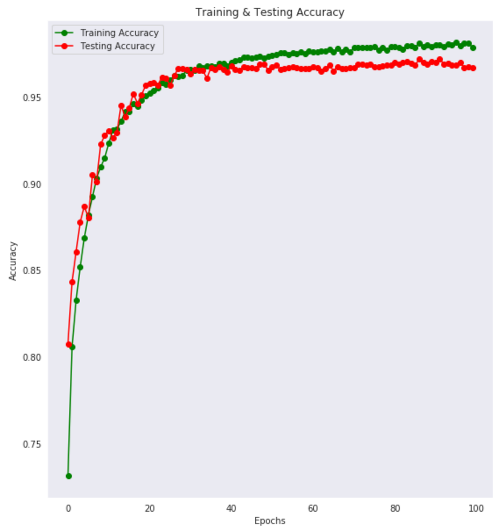

Figure 2. Training & Testing Accuracy MCEA-BLSTM

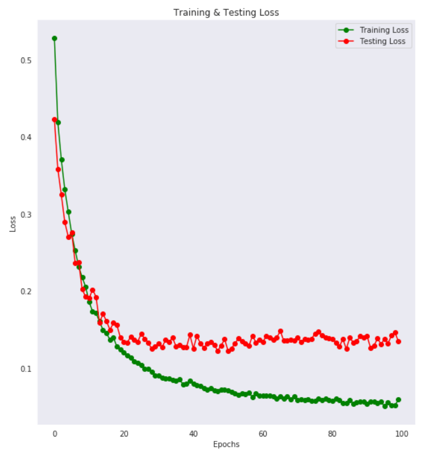

Figure 3. Training & Testing Loss MCEA-BLSTM

As the learning process continues with 100 epochs, it is possible to observe in figure 2 that there is a considerable rise in accuracy and testing as the number of epochs increases. The accuracy of the training process has reached 97.84 percent up to the most recent epoch, while the accuracy of the testing process has been 96.65 percent. 0.0591 percent when they were practicing, and 0.1343 percent while they were being tested. The fact that the model does not suffer from overfitting is another thing that can be seen clearly in Figure 3.

The findings were evaluated using key performance indicators such as accuracy, recall, and precision in addition to the F_1 ([24][25]). The outcomes of applying the suggested MCEA-BLSTM model to the datasets in question are outlined in Table 2, which can be found below.

Table 2. Result Performance of MCEA-BLSTM

Indicator	Value
Precision	97%
Recall	97%
F1	97%
Accuracy	96.64%

3.2 Compared to Other Deep Learning Performance Results

In this study, a comparison is made between the capabilities of the MCEA-BLSTM and those of two distinct deep learning systems: the CNN-LSTM approach and the Hybrid Neural Network approach. This research conducted word embedding for each baseline model using GloVe. We used the four primary performance metrics to evaluate the total performance of the two datasets. Table 3 provides the findings produced by using CNN-LSTM and Hybrid Neural Networks.

Table 3. Result Performance of Other Deep Learning and MCEA-BLSTM

Model	Indicator	Value
CNN-LSTM	Precision	85%
	Recall	87%
	F1	86%
	Accuracy	86.16%
Hybrid Neural Network	Precision	88%
	Recall	90%
	F1	89%
	Accuracy	89.7
MCEA-BLSTM	Precision	97%
	Recall	97%
	F1	97%
	Accuracy	96.64%

The MCEA-BLSTM model, which has been recommended, has a precision for the news headline dataset of 96.64 percent, which is higher than the inaccuracy of the earlier model. The CNN-LSTM has the lowest accuracy of all the news headline datasets, with an accuracy of 86.16 percent. The order in which the models are presented follows the order of CNN-LSTM, Hybrid Neural Network, and MCAB-BSLTM in terms of accuracy. With a recall rate of 97 percent for the datasets in question, the proposed MCEA-BLSTM has the highest recall for the news headline datasets. This recall rate makes it the most accurate method for predicting news headlines. The model suggests that the leading MCEA-BLSTM model also has the most remarkable accuracy value, at 97 percent. Figure 2 presents a comparison of the accuracy values that were produced by the three models that were mentioned earlier in this paragraph in graphical form, which may be seen below.

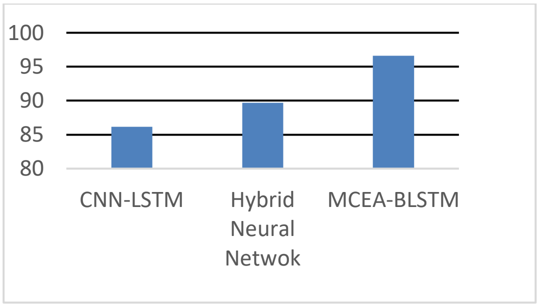

Figure 4. Accuracy result of MCEA-BLSTM and baseline model.

IV. CONCLUSION

The Multiple Channel Embedding Attention BLSTM (MA-BLSTM) is presented in this work for the sarcasm detection model that was explained before. The model is developed using text sentiment analysis characteristics as building blocks. It is essential to build up the two-word embedding at the beginning that is then supplied further into Attention-Based BLSTM and dealt with by the ReLu Layer later on. We may be able to repair the long-term reliance and gradient instability like other deep learning methods during the training process if we concatenate two output matrices for training. Specifically, this will allow us to train more effectively. This would involve putting the matrices together in a single structure. This would be useful. Experiments have shown that using this approach to detect sarcasm results in an accuracy rate of 96.64 percent, which substantially improves over using the traditional method. The findings of this article will serve as a roadmap for future research, which will focus more closely on the structure of the model, explore how the model is affected by using a variety of neural networks, and develop a model that can identify sarcasm.

BIBLIOGRAPHY

[1] P. Katyayan and N. Joshi, *Sarcasm detection approaches for English language*, vol. 374. Springer International Publishing, 2019.

[2] A. C. Pandey, S. R. Seth, and M. Varshney, *Sarcasm detection of Amazon Alexa sample set*, vol. 526. Springer Singapore, 2019.

[3] L. Liu, J. L. Priestley, Y. Zhou, H. E. Ray, and M. Han, “A2Text-net: A novel deep neural network for sarcasm detection,” *Proc. - 2019 IEEE 1st Int. Conf. Cogn. Mach. Intell. CogMI 2019*, pp. 118–126, 2019, doi: 10.1109/CogMI48466.2019.00025.

[4] H. Peng *et al.*, “Large-scale hierarchical text classification with recursively regularized deep graph-CNN,” *Web Conf. 2018 - Proc. World Wide Web Conf. WWW 2018*, pp. 1063–1072, 2018, doi: 10.1145/3178876.3186005.

[5] Z. Wang and B. Song, “Research on hot news classification algorithm based on deep learning,” *Proc. 2019 IEEE 3rd Inf. Technol. Networking, Electron. Autom. Control Conf. ITNEC 2019*, no. Itnec, pp. 2376–2380, 2019, doi: 10.1109/ITNEC.2019.8729020.

[6] P. Mehndiratta and D. Soni, “Identification of sarcasm using word

embeddings and hyperparameters tuning," *J. Discret. Math. Sci. Cryptogr.*, vol. 22, no. 4, pp. 465–489, 2019, doi: 10.1080/09720529.2019.1637152.

[7] Y. Cai, H. Cai, and X. Wan, "Multi-modal sarcasm detection in Twitter with hierarchical fusion model," *ACL 2019 - 57th Annu. Meet. Assoc. Comput. Linguist. Proc. Conf.*, pp. 2506–2515, 2020, doi: 10.18653/v1/p19-1239.

[8] R. Misra and P. Arora, *Sarcasm Detection using Hybrid Neural Network*, vol. 1, no. 1. Association for Computing Machinery, 2019.

[9] S. Hiai and K. Shimada, "Sarcasm detection using RnN with relation vector," *Int. J. Data Warehous. Min.*, vol. 15, no. 4, pp. 66–78, 2019, doi: 10.4018/IJDWM.2019100104.

[10] L. H. Son, A. Kumar, S. R. Sangwan, A. Arora, A. Nayyar, and M. Abdel-Basset, "Sarcasm detection using soft attention-based bidirectional long short-term memory model with convolution network," *IEEE Access*, vol. 7, pp. 23319–23328, 2019, doi: 10.1109/ACCESS.2019.2899260.

[11] T. Xiong, P. Zhang, H. Zhu, and Y. Yang, "Sarcasm detection with self-matching networks and low-rank bilinear pooling," *Web Conf. 2019 - Proc. World Wide Web Conf. WWW 2019*, pp. 2115–2124, 2019, doi: 10.1145/3308558.3313735.

[12] D. Jain, A. Kumar, and G. Garg, "Sarcasm detection in mash-up language using soft-attention based bi-directional LSTM and feature-rich CNN," *Appl. Soft Comput. J.*, vol. 91, p. 106198, 2020, doi: 10.1016/j.asoc.2020.106198.

[13] P. K. Mandal and R. Mahto, "Deep CNN-LSTM with Word Embeddings for News Headline Sarcasm Detection," no. Itng, pp. 495–498, 2019.

[14] A. Kumar, V. T. Narapareddy, V. A. Srikanth, A. Malapati, and L. B. M. Neti, "Sarcasm Detection Using Multi-Head Attention Based Bidirectional LSTM," *IEEE Access*, vol. 8, pp. 6388–6397, 2020, doi: 10.1109/ACCESS.2019.2963630.

[15] A. Onan, "Topic-Enriched Word Embeddings for Sarcasm Identification," *Adv. Intell. Syst. Comput.*, vol. 984, pp. 293–304, 2019, doi: 10.1007/978-3-030-19807-7_29.

[16] Y. Diao *et al.*, "A Multi-Dimension Question Answering Network for Sarcasm Detection," *IEEE Access*, vol. 8, pp. 135152–135161, 2020, doi: 10.1109/ACCESS.2020.2967095.

[17] S. S. Salim, A. Nidhi Ghanshyam, D. M. Ashok, D. Burhanuddin Mazahir, and B. S. Thakare, "Deep LSTM-RNN with word embedding for sarcasm detection on twitter," *2020 Int. Conf. Emerg. Technol. INCET 2020*, pp. 33–36, 2020, doi: 10.1109/INCET49848.2020.9154162.

[18] S. Sangwan, M. A. Akhtar, P. Behera, and A. Ekbal, "Exploring Multimodality for Sarcasm Detection," *2020 Int. Jt. Conf. Neural Networks*, pp. 0–7, 2020.

[19] D. M. Ashok, A. Nidhi Ghanshyam, S. S. Salim, D. Burhanuddin Mazahir, and B. S. Thakare, "Sarcasm detection using genetic optimization on LSTM with CNN," *2020 Int. Conf. Emerg. Technol. INCET 2020*, pp. 3–6, 2020, doi: 10.1109/INCET49848.2020.9154090.

[20] L. Ren, B. Xu, H. Lin, X. Liu, and L. Yang, "Sarcasm Detection with Sentiment Semantics Enhanced Multi-level Memory Network," *Neurocomputing*, vol. 401, pp. 320–326, 2020, doi: 10.1016/j.neucom.2020.03.081.

[21] M. Bedi, S. Kumar, M. S. Akhtar, and T. Chakraborty, "Multi-modal Sarcasm Detection and Humor Classification in Code-mixed Conversations," *IEEE Trans. Affect. Comput.*, vol. 3045, no. c, pp. 1–13, 2021, doi: 10.1109/TAFFC.2021.3083522.

[22] A. Kamal and M. Abulaish, "CAT-BiGRU: Convolution and Attention with Bi-Directional Gated Recurrent Unit for Self-Deprecating Sarcasm Detection," *Cognit. Comput.*, vol. 14, no. 1, pp. 91–109, 2022, doi: 10.1007/s12559-021-09821-0.

[23] M. S. Razali, A. A. Halin, L. Ye, S. Doraisamy, and N. M. Norowi, "Sarcasm Detection Using Deep Learning with Contextual Features," *IEEE Access*, vol. 9, pp. 68609–68618, 2021, doi: 10.1109/ACCESS.2021.3076789.

[24] P. Verma, N. Shukla, and A. P. Shukla, "Techniques of Sarcasm Detection: A Review," *2021 Int. Conf. Adv. Comput. Innov. Technol. Eng. ICACITE 2021*, vol. 7, pp. 968–972, 2021, doi: 10.1109/ICACITE51222.2021.9404585.

[25] N. Majumder, S. Poria, H. Peng, N. Chhaya, E. Cambria, and A. Gelbukh, "Sentiment and Sarcasm Classification with Multitask Learning," *IEEE Intell. Syst.*, vol. 34, no. 3, pp. 38–43, 2019, doi: 10.1109/MIS.2019.2904691.