Computing and Informatics, Vol. 42, 2023, 1091–1113, doi: 10.31577/cai_2023_5_1091

ENHANCED DEEP LEARNING-BASED MODEL FOR SENTIMENT ANALYSIS TO IDENTIFY SARCASM APPEARED IN THE NEWS

Isha Gupta

School of Computer Applications
Manav Rachna International Institute of Research and Studies
Faridabad, India

Indranath Chatterjee*

Department of Computer Engineering, Tongmyong University
Busan, South Korea
&
School of Technology, Woxsen University
Telangana, India
e-mail: indranath.cs.du@gmail.com

Neha Gupta

School of Computer Applications
Manav Rachna International Institute of Research and Studies
Faridabad, India

Abstract. In the field of natural language processing (NLP), detecting emotions or sentiments can be a challenging task, and sometimes emotions can be more complex than just positive or negative. However, detecting sarcasm in textual data adds another layer of complexity. Despite this, identifying the underlying sarcasm in the text has become a recent area of interest among NLP researchers. Headlines in newspapers often use sarcasm to engage readers, but readers may have difficulty

* Corresponding author

recognizing it, leading to a misinterpretation of the news and spreading misinformation. As a result, there is an urgent need for technology that can automatically identify sarcasm with high accuracy. Recent studies in this domain have revealed a need for a robust and efficient model. Deep learning approaches have proven to be effective in sarcasm detection. In this work, we propose a novel two-stage model that uses a word-embedding technique to select relevant features followed by an advanced deep-learning architecture to classify sarcasm in news headlines. Our proposed method demonstrates promising results in identifying sarcasm in text with an accuracy rate of approximately 97 %. We have fine-tuned the hyper-parameters to increase the precision level, which enhances the efficacy of our model. Our work provides a significant contribution to the field of NLP by presenting a reliable and effective model for sarcasm detection. The comparison of our model with recent advancements indicates that our approach outperforms them. By using our model, readers can avoid misinterpretations and the spreading of misinformation. Therefore, our work can have a positive impact on society, and we believe that it can inspire future research in the field of sarcasm detection.

Keywords: Word embeddings, sarcasm detection, deep neural network, GRU, Bi-LSTM, fastText, GloVe, news community

1 INTRODUCTION

Sarcasm entails the use of terms that have a different meaning than what one truly means to express, often to offend or irritate someone or merely for pranks. Recently, studies have looked at how sarcasm is used in a range of everyday scenarios, including circumstances where social standards are broken or at work when giving critique or appreciation. When someone uses sarcasm face-to-face, one might identify it by their expressions. Equally or more helpful, the tone of their voice will likely change, too – they may sound more vibrant or carry out several phrases. However, it can indeed be challenging to decipher sarcasm in written form. In written text, there are no facial cues or vocal tones for a better understanding of the text. Sarcasm detection is a specific area of sentiment analysis [1, 2] which is again a sub-field of natural language programming (NLP) [3] study. Here, the goal is to identify the sarcasm rather than to determine if the emotion is positive or negative. This entails determining whether or if the text is ironic (sarcastic sentiment analysis) [4]. Since sarcasm is more complicated for a computer algorithm to recognize than other human emotions, this field has been fraught with difficulties.

Nowadays, social media platforms play a great role in the communication of text. Twitter and other social media sites have offered authors greater tools to help readers express their intentions. Sarcasm in the text changes a sentence's polarity and reduces the precision of a sentiment analysis task. On Twitter and other social media, people employ relatively casual language, which introduces a limited vocabulary. Whereas in news headlines, there are no spelling errors or colloquial language, as they are prepared by experts adequately. This lessens the sparsity and raises the likelihood of discovering pre-trained embeddings. News portals frequently appear to use sarcasm in their headlines to engage readers. Furthermore, readers frequently have trouble seeing the sarcasm in the headlines, which leads to them having an erroneous impression of the news and spreading that impression to their peers.

Recent studies use deep learning to recognize sarcasm. Algorithms have been developed to detect the existence of sarcasm and harshness in tweets, customer reviews, and discussion boards. The algorithms were fairly efficient in identifying explicit rude language. However, researchers discovered that algorithms require both linguistic and semantic information to be synthesized to effectively detect sarcasm. Many different models have been developed. Studies from the past indicate that deep neural networks and machine learning techniques yield accurate models.

The objective of our paper is to build an advanced deep learning-based model [5, 6] for the detection of sarcasm accurately. Alongside using a word-embedding algorithm, this work employs the use of Long Short-Term Memory (LSTM), Bidirectional LSTM, and Gated Recurrent Unit (GRU), by tuning the hyper-parameters to achieve optimal results. The major contributions of this work lie in finding the efficacy of the proposed two-stage model, using advanced deep learning techniques for sarcasm detection and comparing it with state-of-the-art algorithms. Our algorithm has a wide range of applications in NLP, including consumer research, sentiment analysis, and knowledge classification. Our algorithm has outperformed all previously developed algorithms, achieving a high level of accuracy. The focus of our proposed model is the ability of the LSTM model to capture temporal dependencies and selectively remember and forget information, making it a powerful tool for sarcasm detection. Furthermore, LSTMs are superior to other models in dealing with long-term dependencies.

We have chosen fastText over other word embedding techniques, as it generates better word embeddings for rare words or even words not seen during training. This is an advantage over other word embedding techniques. Our proposed work overcomes the limitations of individual models, such as overfitting and underfitting, which may also account for the outstanding performance of our model.

Our research demonstrates that fastText embedding improves the performance of the multi-layered LSTM model. However, the disadvantage of our model's complex architecture is that it requires more computational time to train than a simple model. On the other hand, the disadvantage of other developed algorithms is that they use traditional word embedding techniques that do not produce high-accuracy results.

Overall, our algorithm is a significant improvement over previous algorithms and offers a powerful solution for NLP applications, particularly for sarcasm detection.

The paper has been discussed in the following sections. The concept of sarcasm detection and its difficulties are covered in Section 1. The literature review was discussed in Section 2. The main ideas of various deep learning algorithms and word embedding techniques, the proposed model to identify sarcasm in the text, and the experimental results are covered in Section 3. Section 4 deals with the results obtained, discussions, and comparisons with other cutting-edge algorithms. In Section 5, we extensively analyze the proposed approach. Lastly, Section 6 provides a summary of our study and proposes potential directions for future research.

2 LITERATURE REVIEW

There is already a significant amount of research available on sarcasm detection and identification. Sarcasm is frequently employed in social networks and e-commerce platforms, hence failing to recognize it in tasks involving NLP can result in false positives, such as sentiment analysis and opinion mining. According to recent studies, the two language traits of sentiment and incongruity information are helpful for sarcasm identification. Chen et al. [7] suggested a multi-task learning approach that models context incongruity while integrating semantic data and soft sentiment labels to incorporate sentiment cues. The suggested model performs better for the sarcasm detection job with the use of sentiment hints and incongruity information, according to experimental results on datasets.

Băroiu and Trăușan-Matu [8] presented a thorough analysis of the literature on the development of the artificial sarcasm detection task from its inception in 2010 to the present. According to this study, transformer-based architectures and multi-modal techniques have grown in popularity recently. Savini and Caragea [9] reviewed the state-of-the-art and current algorithms and provided robust baselines for sarcasm detection using BERT pre-trained language models. They initially fine-tuned them on relevant intermediate tasks and then investigated a transfer learning system that uses sentiment classification. They have also investigated emotion detection as a separate intermediary task to infuse information into the target task of sarcasm detection, specifically depending on the association between sarcasm and (implied negative) sentiment and general emotions. Trial results on three datasets showed that the BERT-based models outflank numerous past models.

The ability to recognize contradictions is the fundamental issue with sarcasm detection. Ashwitha et al. [10] resolved the issue by compiling a list of target words that vary in interpretation depending on the context and evaluated whether a sentence contains an exact or ironic use of an objective word. Nayel et al. [11] carried out the process using a model based on the support vector machine (SVM). The ArSarcasm-v2 dataset has been used to assess the proposed model.

Abu Farha and Magdy [12] assessed the effectiveness of 24 Arabic-specific models for detecting sarcasm and sentiment in Arabic. The findings demonstrated that the models used more parameters and were trained exclusively on Arabic data performed best. The investigations using AraGPT2 variations demonstrated poor performance when compared to BERT models, suggesting that it might not be appropriate for classification tasks.

Ren et al. [5] suggested a multi-level memory network that incorporates sentiment semantics to store the characteristics of expressions of sarcasm. The first-level memory network was used to represent the sentiment semantics, and the second-level memory network represented the contrast between the sentiment semantics and the context of each sentence. They also enhanced the memory network in the absence of local information using an upgraded CNN. The experimental outcomes on the Twitter dataset and Internet Argument Corpus (IAC-V1 and IAC-V2). Majumder et al. [13] developed a multitask learning-based framework that analyses the association between sarcasm detection and sentiment analysis using a deep neural network to enhance the performance of both tasks in a multitask learning environment.

Sharma et al. [14] suggested a new autoencoder-based hybrid sentence embedding technique. Their approach employed a bidirectional encoder representation transformer, a universal sentence encoder, and sentence embedding from LSTM-autoencoder. Du et al. proposed a dual-channel CNN [15] that examines the emotional context of the target text in addition to its semantics. SenticNet was used to enhance the LSTM model with common sense. The user's expression patterns were then taken into consideration using the attention method.

Vinoth and Prabhavathy [16] developed an automated sarcasm detection and classification tool for social media utilizing an ASDC-HPTDL model for hyper-parameter-optimized deep learning. To recognize and categorize sarcasm, the attention bidirectional gated recurrent unit (ABiGRU) technique and improved artificial flora algorithm (IAFO) hyper-parameter tuning method were employed. To recognize and comprehend sarcastic behavior and patterns, Goel et al. [6] tried to close the intelligence gap between humans and machines. To identify sarcasm on the internet, the research relied on an ensemble model that combines many neural processing approaches, including LSTM, GRU, and Baseline Convolutional Neural Networks (CNN).

Barhoom et al. [17] developed a model that can recognize sarcasm in headline news using machine and deep learning to investigate how a computer can recognize sarcastic patterns. Razali et al. [18] focused on identifying sarcasm in tweets by fusing hand-crafted contextual data with information retrieved through deep learning. Feature sets were retrieved from CNN architecture and then mixed with meticulously designed feature sets.

Kumar et al. [19] built a feature-rich SVM that outperformed past models by extracting the most important characteristics. To detect sarcastic comments in a given corpus, they presented a multi-head attention-based bidirectional long-short memory (MHA-BiLSTM) network. Son et al. [20] proposed a profound learning model called sAtt-BLSTM convNet that depends on the crossbreed of soft attention-based bidirectional long short-term memory (sAtt-BLSTM) and convolution neural network (convNet), applying GloVe for building semantic word embeddings.

From the vast rigorous literature survey, we concluded that many researchers applied deep neural networks and word embedding for the detection of sarcasm, however, the limitations of the algorithm and efficacy of the approach still lag in terms of domain exchange. To brief, we designed Table 1 to showcase the performance achieved by different authors in their work.

Reference and Dataset	**Word Embedding**	**Techniques used**	**Performance**
SARC [21]	Word2Vec, fastText, GloVe	CNN, LSTM	Acc: 0.82
Sarcasm Corpus 1 [22]	Word2Vec, fastText, and GloVe	Bi-LSTM	Acc: 95.3 %
News Headlines [23]	Word embedding	HA-LSTM	F1: 0.86
SARC [24]	User embedding	CASCADE	F1: 0.86, Acc: 0.79
Real-time mash-up tweets [25]	GloVe	BiLSTM + CNN	Acc: 92.71 %, F-score: 89.05 %
Internet Argument Corpus [5]	GloVe	CNN	F1: 0.87
Eye Movement Database [13]	Sentence-level word representation	GRU with attention	F1-score: 0.87
Tweets [26]	User embedding	CNN	F1-score: 0.74
Tweets [27]	Word2Vec	CNN + Bi-LSTM	F1-score: 0.85
Hotel reservation reviews [28]	GloVe	Proposed model	Acc: 93.85 %
SST2 [29]	fastText	CNN	Acc: 84 %
SemEval 2018 [30]	–	LMTweets + CNN	Acc: 0.883
Tweets [18]	fastText	CNN	Acc: 94 %
SARC dataset [19]	Sentence embedding	MHA-BiLSTM	F1-score: 77.48 %
SemEval 2015 Task 11 [31]	GloVe	Bi-LSTM	Acc: 86.32 %
Semeval 2014 [32]	Word2Vec	CNN	F1: 97.71 %
Tweets [33]	Word embedding	RNN + LSTM	Acc: 91 %
Semeval 2018 Task 3 [34]	–	Google BERT	F1-score: 69.64 %
Ghosh Dataset [35]	Word embedding	Attentive RNN	F1-score: 95.5 %
Reddit Dataset [36]	fastText	BERT-Base-Cased	Acc: 98.25 %
ArSarcasm v2 dataset [37]	Mazajak embeddings	WANLP (AraBERT + CNN-BiLSTM)	Acc: 0.74
MUStARD and sarcasm detection Reddit track [38]	–	Fuzzy logic	Acc: 75.4 %
SemEval 2015 [39]	GloVe	CAT-BiGRU	Acc: 93 %
Rilloff Sarcastic Dataset [40]	Transformer-based contextual embedding	Attention-based BiLSTM	Acc: 93 %

Table 1. Literature showcasing the works on sarcasm detection involving word embedding and AI techniques

3 METHODOLOGY

The proposed methodology for sarcasm detection consists of mainly two phases. In the first phase, feature selection was done by word embedding technique known as fastText and the second phase mainly focuses on classification of sarcasm in news headlines with the help of deep learning techniques. Figure 1 depicts the architecture of the proposed approach.

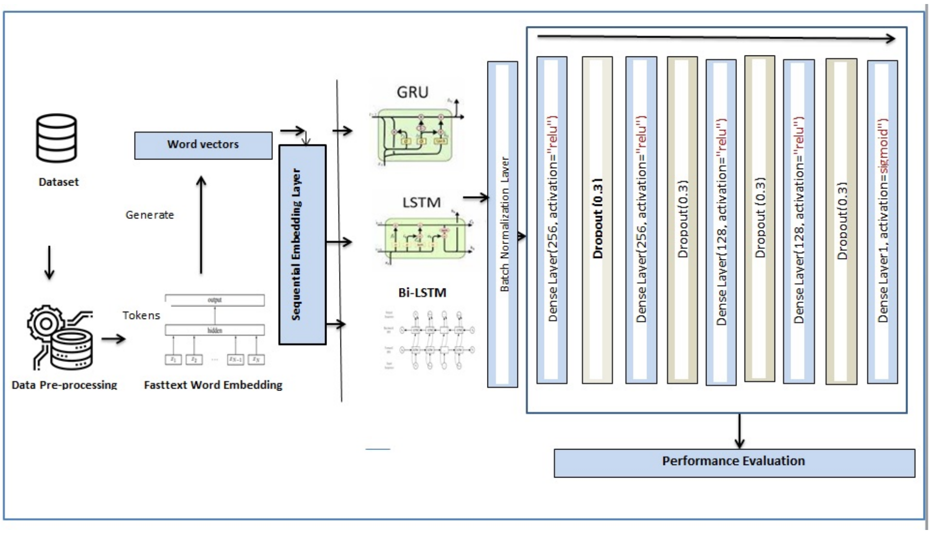

Dataset
Data Pre-processing
Tokens
Generate
Word vectors
Fasttext Word Embedding
Sequential Embedding Layer
GRU
LSTM
Bi-LSTM
Batch Normalization Layer
Dense Layer(256, activation="relu")
Dropout (0.3)
Dense Layer(256, activation="relu")
Dropout(0.3)
Dense Layer(128, activation="relu")
Dropout (0.3)
Dense Layer(128, activation="relu")
Dropout(0.3)
Dense Layer1, activation=sigmoid")
Performance Evaluation

Figure 1. Architectural design of the proposed algorithm

3.1 Word Embedding

Word embedding is a method of portraying words and written materials. It is a lower-dimensional numeric vector input that represents a word, which enables similar representations for words with similar meanings. Word embedding [41] is a technique for taking the features from text and putting them into machine-learning models, which are used for dimension reduction, word prediction, and inter-word semantics detection. While extracting the features, it makes an effort to maintain semantic and syntactical information within the text. The word count in a sentence is used by techniques like BOW, Count Vectorizer, and TF-IDF, although no syntactical or semantic data is saved. The number of vocabulary elements in these methods determines the size of the vector. If the majority of the elements are zero, we can have a sparse matrix. Large input vectors will result in plenty of weights, which will increase the amount of training computation. Word embedding serves as a solution to these issues. The popular word embedding techniques available are Word2Vec by Google, GloVe by Stanford University, and fastText by Facebook.

In this study, we have employed the usage fastText algorithm as a tool for word embedding technique.

3.1.1 FastText

FastText is a library that makes learning word representations and sentence classification quick and easy [42]. It allows multiprocessing while being trained and is written in C++. The user can train supervised and unsupervised word and phrase representations with fastText. Data compression, features in extra models, candidate selection, and fabricators for transfer learning are a few uses for these embedding. Skip-Gram and Continuous Bag of Words (CBOW) are two of the word representation computation models that fastText offers. By using a word that is close by, the Skip-Gram model can learn to anticipate a target word. The CBOW model, on the other hand, makes predictions about the target word based on the context. The target word's fixed-size windows that make up the context are depicted as a bag of words. FastText allows one to train CBOW or Skip-Gram models using softmax, hierarchical softmax, or negative sampling methods.

The distribution of fastText can be defined by the equation as shown in Equations (1) and (2):

$$P(w) = \sqrt{\frac{t}{f(w)}} + \frac{t}{f(w)}, \tag{1}$$

$$f(w) = \frac{count_w}{totalno.oftokens}, \tag{2}$$

where $t = 0.001$ is the chosen threshold, $f(w)$ is the frequency of occurrence for word w.

3.2 Deep Learning Techniques

3.2.1 LSTM (Long Short-Term Memory)

LSTM is a unique class of recurrent neural network (RNN), which is capable of learning long-term dependencies mainly in sequence prediction problems [43]. It can solve the RNN's vanishing gradient problem. LSTM has feedback connections, so it can process all of the data in a sequence.

The architecture of the LSTM model (shown in Figure 2) is different from RNN. In an LSTM model, a memory cell referred to as a "cell state" is vital, as it retains its state over time. Figure 2 shows the cell state as a horizontal line at the top. The cell state can be thought of as a conveyor belt where data passes through without alteration. The gates in LSTMs regulate the addition and removal of information from the cell state. These gates enable data to enter and exit the cell. The mechanism is supported by a sigmoid neural network layer and a point-wise multiplication

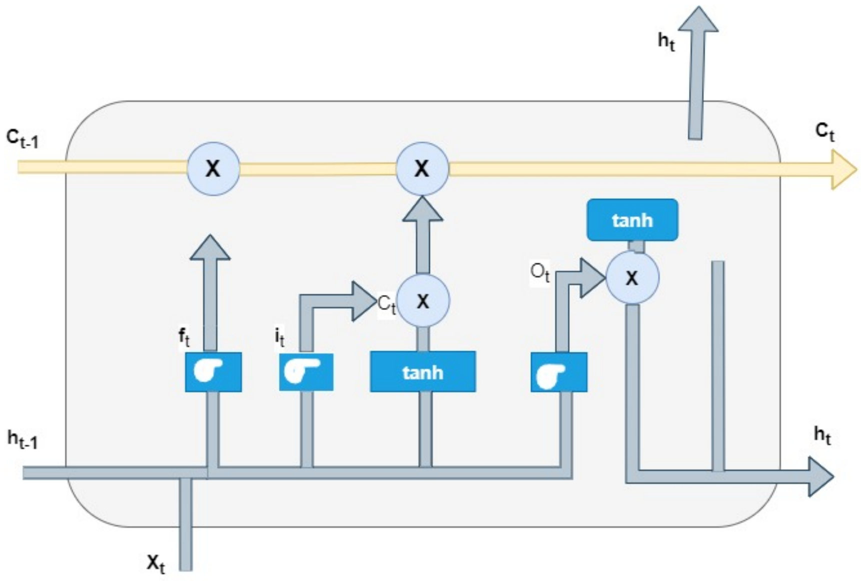


Figure 2. Architecture of LSTM

operation. An LSTM has three gates, to protect and control the cell state. The various equations (as stated in Equations (3), (4), (5), (6), (7) and (8)) associated with LSTM are:

$$f_t = \sigma(W_f.[h_{(t-1)}, x_t] + b_f), \tag{3}$$

$$i_t = \sigma(W_i.[h_{(t-1)}, x_t] + b_i), \tag{4}$$

$$\tilde{C}_t = tanh(W_c.[h_{(t-1)}, x_t] + b_c), \tag{5}$$

$$C_t = f_t * C_{(t-1)} + i_t * \tilde{C}_t, \tag{6}$$

$$O_t = \sigma(W_O.[h_{(t-1)}, x_t] + b_o), \tag{7}$$

$$h_t = O_t * tanh(C_t), \tag{8}$$

where x_t is the input vector, f_t is forget gate activation vector, i_t is input gate activation vector, o_t is output gate activation vector, h_t is the hidden state vector, c_t is the cell state vector.

3.2.2 Bi-LSTM (Bidirectional LSTM)

Bidirectional LSTM resembles an improvement over the generic LSTM algorithm. Each training sequence is presented both forward and backward in bidirectional LSTMs to differentiate recurrent nets. To maintain both past and future information, bidirectional input can be made to flow in both ways. The same output layer is shared by both sequences. Bidirectional LSTMs [44] are fully aware of every point in a given sequence, as well as everything that came before and came after it. Speech recognition, text categorization, and forecasting models can all employ this type of network. The structure of Bi-LSTM is shown in Figure 3.

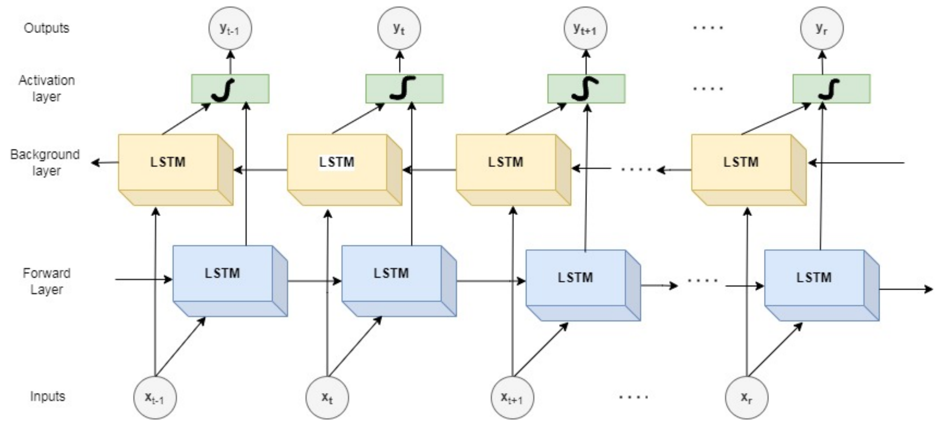


Figure 3. Structure of Bi-LSTM Model

3.2.3 Gated Recurrent Unit (GRU)

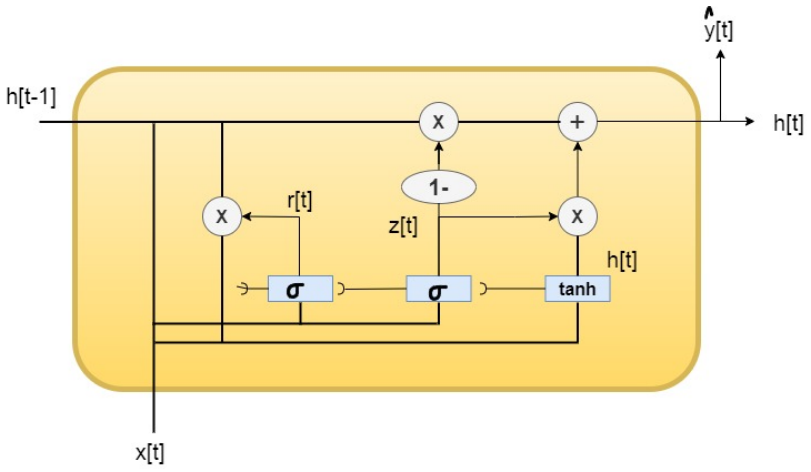


Figure 4. Architecture of GRU

GRU belongs to the RNN family of variations [45]. It only has three gates, unlike LSTM, and it does not keep track of the internal state of the cell. The architecture of GRU is shown in Figure 4. The data that is kept in the internal cell state of an LSTM recurrent unit is incorporated into the hidden state of the GRU. The next GRU receives this group of data. The internal operation of each recurrent unit in GRU networks, which includes gates that modify the current input and the prior hidden state, is the primary distinction between GRU and RNN. The update gate and reset gate are used to determine the data that should be sent to the output.

The mathematical equations (as shown in Equations (9), (10), (11) and (12)) involved are:

$$z_t = \sigma_g(W_z x_t + U_z h_{(t-1)} + b_z), \tag{9}$$

$$r_t = \sigma_g(W_z x_t + U_r h_{(t-1)} + b_r), \tag{10}$$

$$\tilde{h_t} = \phi_h(W_h x_t + U_h(r_t^\circ h_{(t-1)}) + b_h), \tag{11}$$

$$h_t = z_t^\circ \tilde{h_t} + (1 - z_t)^\circ h_{(t-1)}, \tag{12}$$

where $^\circ$ denotes the Hadamard product, x_t and h_t are the input and output vectors respectively, z_t is the update gate vector, r_t reset gate vector, $\tilde{h_t}$ is candidate activation vector, W, U, and b are parameters.

3.3 Experimental Setup

All the experimentations were performed in a High-Performance Computing facility having AMD Ryzen Threadripper PRO 3945 WX processor with 12 cores and 64 GB DDR4 Quad channel RAM. The deep learning architectures were supported with an NVIDIA RTX A5000 graphics card having 24 GB DDR6 memory.

3.3.1 Dataset Details: News Headlines Dataset

The majority of earlier research on sarcasm detection relied on Twitter datasets gathered under hashtag supervision, although these datasets are messy in terms of labels and language. The availability of contextual tweets is also necessary because many tweets are answers to other tweets, making it difficult to discern sarcasm in them. Here, we have used a news headlines dataset [46, 47]. This headlines dataset for sarcasm detection is compiled from two news websites, The Onion, and the HuffPost, to get over the noise constraints in Twitter datasets. Compared to other datasets, these datasets have significant advantages. First headlines are composed with professionalism. There are not any grammatical or spelling errors. Unlike tweets, they carry only text. The only ironic news is printed in The Onion. A total of 28 619 fresh headlines are included in the dataset. The records that fall under the category of sarcastic remarks amount to 13 635. There are 14 984 non-sarcastic records. The collection includes news headline titles as well as labels designating sarcastic and non-sarcastic headlines. The label value “0” indicates that the headline is not sarcastic, whereas the label value “1” indicates that it is. 13 635 records are included in the category of sarcastic remarks. 14 984 records are not ironic.

3.3.2 Data Pre-Processing

At first, the dataset of news headlines has been pre-processed. Raw data were transformed into a format that computers and deep learning algorithms can comprehend and analyze during the data pre-processing stage of the proposed process. The data cleaning stage includes the removal of stopwords, punctuation, and noise.

3.3.3 Application of Word Embedding Technique

The next step was to create word vectors of the pre-processed dataset. This was achieved using the fastText word embedding technique. FastText package facilitates effective word representation and sentence classification learning. As its name implies, it is a quick and effective way to complete the aforesaid task, and because of the nature of its training process, it also ends up learning specific morphological information. FastText is exceptional in generating word vectors for unknown words or from vocabulary words. This is possible because it takes into account the morphological properties of words while generating the word vector for an unknown word. Rare words function well with fastText. To obtain its embeddings and sentence categorization, a word can still be broken down into n-grams, even if it was not observed during training. In our algorithm, around 2 000 000 word vectors are generated from fastText.

3.3.4 Application of Deep Learning Algorithms

In our second phase, the deep neural network (DNN) comes into role. An artificial neural network having more than two layers between the input and output layers is called a deep neural network. The network consists of neurons, synapses, weights, biases, and functions. Keras is a deep-learning API created in Python. One of the layers in Keras is the embedding layer. This is primarily utilized in NLP-related applications and can also be applied to other neural network-based tasks. We utilized Keras embedding layer to train our embeddings. The neurons, synapses, weights, biases, and functions are the components of the embedding layer. So, the first layer of the second phase of our algorithm starts with embedding a sequential model. The value of the embedding dimension was 300.

The input length is the length of input sequences, and it remains constant. In our case, its value was 25. The embeddings_initializer is the embedding matrix which is the output of fastText word embedding. We kept the parameter trainable as 'False'. All of the layer's weights are converted from trainable to non-trainable when layer trainable is set to False. The state of a frozen layer will not be updated throughout training.

The second layer of DNN is set to LSTM, BiLSTM, and GRU one by one. The dimensionality of output space was kept at 256. The next layer is the batch normalization layer. It was used to normalize each mini-batch. As a result, the learning process is stabilized, and the quantity of training epochs needed to train deep neural networks is dramatically reduced throughout training. There is an addition of a dense layer that is closely coupled to its previous layer, meaning that every neuron in the layer is connected to every other neuron in the preceding layer. The activation function is set to 'ReLu' in the dense layer. To avoid overfitting, the dropout layer of 0.3 (30 %) that randomly sets input units to 0 with a frequency of rate at each step during training. Three more consecutive layers of dense and drop layers are added to the model. Lastly, a dense layer of output unit 1 with the activation function 'Sigmoid' is added to the model. Sigmoid is used for binary classification. Table 2 shows the parameter list exercised in the model.

Parameter Name	**Best Value**
Epochs	50
Batch size	16
Activation function	Sigmoid, ReLu
Embedding dimension	300
Optimizer	Adam
Dropout	0.3
Filter	256, 128

Table 2. List of hyper-parameters and their optimal values of the model

The entire dataset was cross-validated keeping 80 % of the set for training purposes and the rest for the validation set. The hyper-parameters were tuned to find the best value at which our algorithm outperforms other algorithms. The model was evaluated with an epoch ranging from 20 to 200 at an interval of 10. We found that the model reached its optimal capacity at the value of 50 epochs. The value of batch size has been kept in 8, 16, and 32. Two types of activation functions have been used – Sigmoid and ReLu. The embedding dimension has been kept at 300 as the fastText library of 300 dimensions has been used. Adam and Gradient Descent optimizer has been used for the model. Adam has been chosen as it has several advantages that combine the benefits of Gradient with Momentum and RMSProp, such as limited computational requirements, adaptability for non-stationary objectives, and efficient computation with large data and parameters. Different dropout values 0.1, 0.2, and 0.3 has been applied. The best value achieved on the hypertuning of parameters has been shown in Table 2. Here, we have used the following performance metrics to calculate the performance of our model.

3.3.5 Performance Metrics

For calculating the performance of our proposed model, the following metrics (Equations (13), (14) and (15)) are used:

Accuracy: The accuracy of a classifier is calculated by dividing the total number of correctly predicted samples by the total number of samples in the dataset.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}. \tag{13}$$

Precision: Precision refers to the proportion of true positives (TP) to the combined number of TP and false positives (FP).

$$Precision = \frac{TP}{TP + FP}. \tag{14}$$

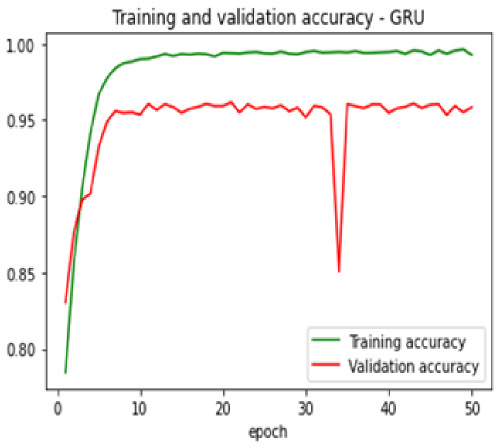
Recall: Recall is the ratio of true positives (TP) by the sum of true positives (TP) and false negatives (FN).

$$Recall = \frac{TP}{TP + FN}. \tag{15}$$

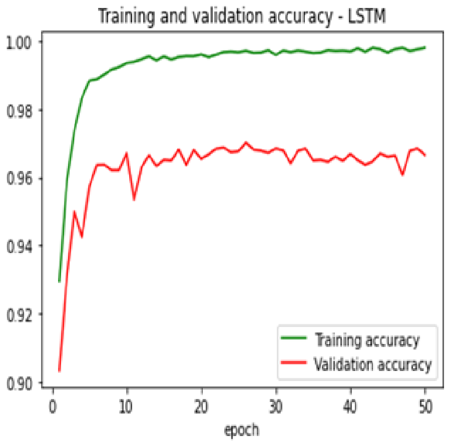
Where TP refers to true positives, TN is true negatives, FN is false negatives and FP is false positives.

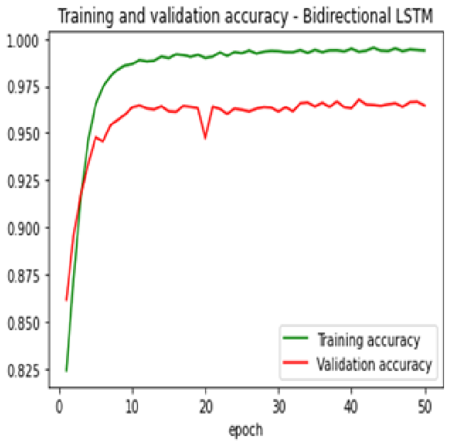
4 RESULTS

After the application of the proposed approach incorporating the word-embedding technique and deep learning algorithms, we obtained promising results for all three algorithms, in terms of performance metrics. Figure 5 shows the training v/s validation curves in terms of accuracy results at epoch 50 of LSTM, Bi-LSTM, and GRU. Figure 5 depicts the training and validation accuracy of all three deep learning techniques. This feature highlights one of the advantages of the suggested algorithm. All three techniques achieve a training accuracy of over 99 % and a validation accuracy of over 95 %. Figure 6 displays the performance metrics, including accuracy, precision, and recall of GRU, LSTM, and Bi-LSTM for both training and validation data. The results indicate that the LSTM technique outperforms the Bi-LSTM and GRU models when fastText word embedding is employed in terms of accuracy. Table 3 shows the performance metrics in terms of accuracy, precision, and recall of the proposed algorithm with each of the variants of RNN being used, namely, GRU, LSTM, and Bi-LSTM. We achieved the highest validation accuracy with LSTM, which is 96.70 %. GRU achieved a validation accuracy of 96.06 % and Bi-LSTM achieved a validation accuracy of 95.77 %. The result also indicates the LSTM achieved 99.79 % training accuracy which is the highest value among the other two models. Discussing the validation precision values, GRU has the highest value of 97.71 %, succeeded by a Bi-LSTM with a value of 97.43 %, and LSTM with a value of 97.17 %. Although the highest training precision value was obtained by LSTM, i.e., 99.79 %. LSTM has the highest validation recall value of 97.85 % and training



a) GRU



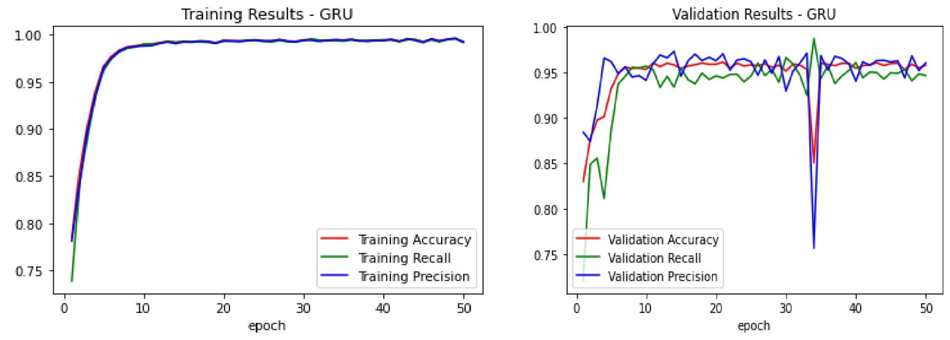


b) LSTM c) Bi-LSTM

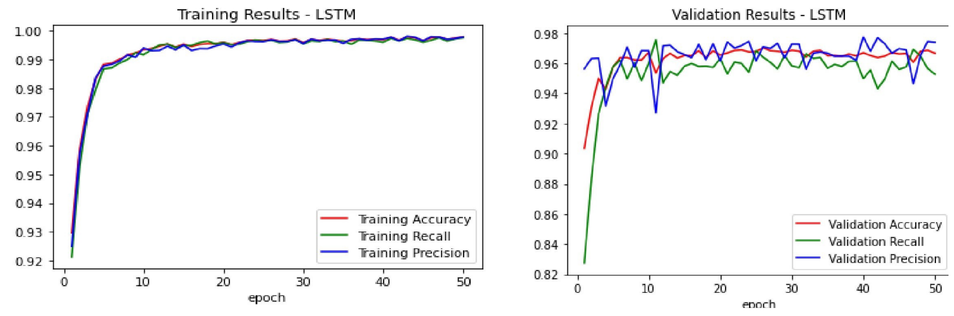
Figure 5. Training accuracy vs validation accuracy for each of the stated algorithms

	Accuracy		**Precision**		**Recall**	
DL technique	**Training**	**Validation**	**Training**	**Validation**	**Training**	**Validation**
GRU	99.28 %	96.06 %	99.20 %	97.71 %	99.23 %	96.08 %
LSTM	99.79 %	96.70 %	99.79 %	97.17 %	99.76 %	97.85 %
Bi-LSTM	99.38 %	95.77 %	99.21 %	97.43 %	99.44 %	96.40 %

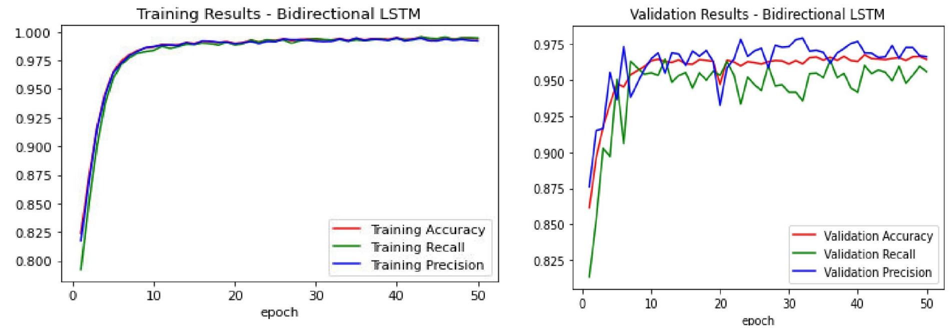
Table 3. Results of the proposed approach



a) Training and validation results for GRU



b) Training and validation results for LSTM



c) Training and validation results for Bi-LSTM

Figure 6. Performance metrics showing the results of training and validation for each of the stated algorithms

Publication	**Word Embedding**	**Deep Learning Model**	**Accuracy**	**Precision**	**Recall**
[6]	GloVe	LSTM	96 %	NA	NA
[36]	fastText	BERT-Base-Cased	92.27 %	92.2 %	92.35 %
[46]	User embeddings	Bi-LSTM + Attention	89.7 %	NA	NA
[48]	Word embeddings	Multi-head attention + GRU	91.6 %	91.9 %	91.8 %
[14]	Sentence embedding	LSTM	90.8 %	92 %	91.2 %
[16]	GloVe	GRU	96.63 %	96.76 %	96.82 %
[17]	GloVe	LSTM	95.27 %	94.15 %	96.62 %
Proposed Model	**fastText**	**LSTM**	**96.70 %**	**97.17 %**	**97.85 %**

Table 4. Comparison results in terms of accuracy with the relevant literature on the news headline dataset

value of 99.76 %. GRU has a 96.08 % recall value whereas Bi-LSTM has a 96.40 % recall value.

Table 4 shows a comparative analysis in terms of the accuracy, precision, and recall of the proposed approach with the recently published relevant literature. All the results were compared on the same dataset, i.e. the news headline dataset. It is observed that our proposed two-stage model comprising fastText word embedding techniques and hyper-parameter-tuned deep learning algorithms has acquired the maximum accuracy of 96.70 %, the precision value of 97.17 %, and recall value of 97.85 % which is the highest among the state-of-the-art algorithms. Our proposed model word embedding technique with fastText with LSTM deep learning technique shows that the technique is very powerful in identifying sarcasm in the News Headlines dataset.

5 DISCUSSION

As discussed so far, we proposed a two-stage model employing the fastText algorithm in the first stage followed by the hyper-parameter-tuned deep learning algorithms, such as GRU, LSTM, and Bi-LSTM. It is evident from the efficacy of our proposed approach that this approach is efficient in detecting sarcasm within the text with a high level of precision. By placing the different models, such as LSTM, Bi-LSTM, and GRU in a deep learning network and varying their hyper-parameters, we attempted to assess the effectiveness of each model. The outcomes showed that the hyper-parameter setup and the dataset we are attempting to examine can sometimes be the cause of performance loss. We have noticed some instances when a significant performance boost has been seen. This corroborates our original hypothesis, according to which we suggested adjusting the hyper-parameters to learn how the models behave. The results indicate that the LSTM model has performed the best. The LSTM model had been pre-trained using fastText word embedding that enhances the vocabulary of the model and helped in achieving the highest accuracy. In comparison to the past studies, the proposed technique performed noticeably better across all measures. This substantiates our assertion that feature sets explicitly created with fastText word embedding were helpful.

The experiment using more combinations of hyper-parameters, dense, and dropout layers produced positive results. The studies carried out in this study are highly detailed and extensive; they also demonstrate that the best outcomes come from combining several models, which supports our initial hypothesis. In the future, we intend to test the effectiveness of the suggested strategy using a sizable corpus and other domain-specific corpora. Future research will also concentrate on examining previous tweets and user behaviors to better understand their personalities and profiles and, as a result, significantly improve the disambiguation between sarcastic and non-sarcastic language.

6 CONCLUSION

NLP researchers have shown an increasing interest in comprehending textual nature and sarcasm detection, particularly in news headlines. In this paper, we introduce a novel two-stage model that classifies sarcasm in news headlines using GRU, LSTM, and Bi-LSTM and extracts features using a fastText word-embedding technique. Our method yields encouraging outcomes, with a remarkable accuracy rate of 97 %. Our model's hyper-parameters have been further tuned to increase accuracy. Notably, when compared to recent developments in this area, our approach shows higher performance. These results show that our research makes a substantial contribution to the NLP field. Our technique offers practical implications for sentiment analysis and opinion mining in news articles by successfully identifying sarcasm in headlines. Additionally, it raises the standard and dependability of automated text analysis systems. Future research should concentrate on improving our model by investigating new methodologies and using larger datasets. We want to advance the study of sarcasm detection in NLP through ongoing development. Overall, our research highlights the potency of our two-stage model for deciphering ironic text in news headlines and for advancing NLP research.

Acknowledgments

The authors are thankful to the Manav Rachna International Institute of Research and Studies, India, and Tongmyong University, South Korea for providing the platform to implement the research study.

Funding. This research work received no external funding.

Conflict of Interest. The authors declared no conflict of interest.

REFERENCES

[1] Gupta, I.—Chatterjee, I.—Gupta, N.: A Two-Staged NLP-Based Framework for Assessing the Sentiments on Indian Supreme Court Judgments. International Journal of Information Technology, Vol. 15, 2023, No. 4, pp. 2273–2282, doi: 10.1007/s41870-023-01273-z.

[2] Gupta, I.—Chatterjee, I.—Gupta, N.: Latent Semantic Analysis Based Real-World Application of Topic Modeling: A Review Study. 2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS), 2022, pp. 1142–1149, doi: 10.1109/ICAIS53314.2022.9742848.

[3] Chowdhary, K. R.: Natural Language Processing. Fundamentals of Artificial Intelligence, Springer, New Delhi, 2020, pp. 603–649, doi: 10.1007/978-81-322-3972-7_19.

[4] Hussein, D. M. E. D. M.: A Survey on Sentiment Analysis Challenges. Journal of King Saud University - Engineering Sciences, Vol. 30, 2018, No. 4, pp. 330–338, doi: 10.1016/j.jksues.2016.04.002.

[5] Ren, L.—Xu, B.—Lin, H.—Liu, X.—Yang, L.: Sarcasm Detection with Sentiment Semantics Enhanced Multi-Level Memory Network. Neurocomputing, Vol. 401, 2020, pp. 320–326, doi: 10.1016/j.neucom.2020.03.081.

[6] Goel, P.—Jain, R.—Nayyar, A.—Singhal, S.—Srivastava, M.: Sarcasm Detection Using Deep Learning and Ensemble Learning. Multimedia Tools and Applications, Vol. 81, 2022, No. 30, pp. 43229–43252, doi: 10.1007/s11042-022-12930-z.

[7] Chen, W.—Lin, F.—Zhang, X.—Li, G.—Liu, B.: Jointly Learning Sentimental Clues and Context Incongruity for Sarcasm Detection. IEEE Access, Vol. 10, 2022, pp. 48292–48300, doi: 10.1109/ACCESS.2022.3169864.

[8] Băroiu, A. C.—Trăușan-Matu, S.: Automatic Sarcasm Detection: Systematic Literature Review. Information, Vol. 13, 2022, No. 8, Art. No. 399, doi: 10.3390/info13080399.

[9] Savini, E.—Caragea, C.: Intermediate-Task Transfer Learning with BERT for Sarcasm Detection. Mathematics, Vol. 10, 2022, No. 5, Art. No. 844, doi: 10.3390/math10050844.

[10] Ashwitha, A.—Shruthi, G.—R., S. H.—Upadhyaya, M.—Ray, A. P.—Manjunath, T. C.: Sarcasm Detection in Natural Language Processing. Materials Today: Proceedings, Vol. 37, 2020, pp. 3324–3331, doi: 10.1016/j.matpr.2020.09.124.

[11] Nayel, H.—Amer, E.—Allam, A.—Abdallah, H.: Machine Learning-Based Model for Sentiment and Sarcasm Detection. Proceedings of the Sixth Arabic Natural Language Processing Workshop, Association for Computational Linguistics, 2021, pp. 386–389, `https://aclanthology.org/2021.wanlp-1.51`.

[12] Abu Farha, I.—Magdy, W.: Benchmarking Transformer-Based Language Models for Arabic Sentiment and Sarcasm Detection. Proceedings of the Sixth Arabic Natural Language Processing Workshop, Association for Computational Linguistics, 2021, pp. 21–31, `https://aclanthology.org/2021.wanlp-1.3`.

[13] Majumder, N.—Poria, S.—Peng, H.—Chhaya, N.—Cambria, E.—Gelbukh, A.: Sentiment and Sarcasm Classification with Multitask Learning. IEEE Intelligent Systems, Vol. 34, 2019, No. 3, pp. 38–43, doi: 10.1109/MIS.2019.2904691.

[14] Sharma, D. K.—Singh, B.—Agarwal, S.—Kim, H.—Sharma, R.: Sarcasm Detection over Social Media Platforms Using Hybrid Auto-Encoder-Based Model. Electronics, Vol. 11, 2022, No. 18, Art. No. 2844, doi: 10.3390/electronics11182844.

[15] Du, Y.—Li, T.—Pathan, M. S.—Teklehaimanot, H. K.—Yang, Z.: An Effective Sarcasm Detection Approach Based on Sentimental Context and Individual Expression Habits. Cognitive Computation, Vol. 14, 2022, No. 1, pp. 78–90, doi: 10.1007/s12559-021-09832-x.

[16] Dakshnamoorthy, V.—Prabhavathy, P.: Automated Sarcasm Detection and Classification Using Hyperparameter Tuned Deep Learning Model for Social Networks. Expert Systems, Vol. 39, 2022, No. 10, Art. No. e13107, doi: 10.1111/exsy.13107.

[17] Barhoom, A.—Abu-Nasser, B. S.—Abu-Naser, S. S.: Sarcasm Detection in Headline News Using Machine and Deep Learning Algorithms. International Journal of Engineering and Information Systems (IJEAIS), Vol. 6, 2022, No. 4, pp. 66–73.

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

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

[20] Kumar, A.—Sangwan, S. R.—Arora, A.—Nayyar, A.—Abdel-Basset, M.: Sarcasm Detection Using Soft Attention-Based Bidirectional Long Short-Term Memory Model with Convolution Network. IEEE Access, Vol. 7, 2019, pp. 23319–23328, doi: 10.1109/ACCESS.2019.2899260.

[21] Mehndiratta, P.—Soni, D.: Identification of Sarcasm Using Word Embeddings and Hyperparameters Tuning. Journal of Discrete Mathematical Sciences and Cryptography, Vol. 22, 2019, No. 4, pp. 465–489, doi: 10.1080/09720529.2019.1637152.

[22] Onan, A.—Toçoğlu, M. A.: A Term Weighted Neural Language Model and Stacked Bidirectional LSTM Based Framework for Sarcasm Identification. IEEE Access, Vol. 9, 2021, pp. 7701–7722, doi: 10.1109/ACCESS.2021.3049734.

[23] Pandey, R.—Kumar, A.—Singh, J. P.—Tripathi, S.: Hybrid Attention-Based Long Short-Term Memory Network for Sarcasm Identification. Applied Soft Computing, Vol. 106, 2021, Art. No. 107348, doi: 10.1016/j.asoc.2021.107348.

[24] Hazarika, D.—Poria, S.—Gorantla, S.—Cambria, E.—Zimmermann, R.—Mihalcea, R.: CASCADE: Contextual Sarcasm Detection in Online Discussion Forums. Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018), Association for Computational Linguistics, 2018, pp. 1837–1848, `https://aclanthology.org/C18-1156`.

[25] Jain, D.—Kumar, A.—Garg, G.: Sarcasm Detection in Mash-Up Language Using Soft-Attention Based Bi-Directional LSTM and Feature-Rich CNN. Applied Soft Computing, Vol. 91, 2020, Art. No. 106198, doi: 10.1016/j.asoc.2020.106198.

[26] Naz, F.—Kamran, M.—Mehmood, W.—Khan, W.—Alkatheiri, M. S.—Alghamdi, A. S.—Alshdadi, A. A.: Automatic Identification of Sarcasm in

Tweets and Customer Reviews. Journal of Intelligent and Fuzzy Systems, Vol. 37, 2019, No. 5, pp. 6815–6828, doi: 10.3233/JIFS-190596.

[27] Ghosh, A.—Veale, T.: Magnets for Sarcasm: Making Sarcasm Detection Timely, Contextual and Very Personal. In: Palmer, M., Hwa, R., Riedel, S. (Eds.): Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, 2017, pp. 482–491, doi: 10.18653/v1/D17-1050.

[28] Onan, A.: Sentiment Analysis on Product Reviews Based on Weighted Word Embeddings and Deep Neural Networks. Concurrency and Computation: Practice and Experience, Vol. 33, 2020, No. 5, Art. No. e5909, doi: 10.1002/cpe.5909.

[29] Khasanah, I. N.: Sentiment Classification Using FastText Embedding and Deep Learning Model. Procedia Computer Science, Vol. 189, 2021, pp. 343–350, doi: 10.1016/j.procs.2021.05.103.

[30] Ahuja, R.—Sharma, S. C.: Transformer-Based Word Embedding with CNN Model to Detect Sarcasm and Irony. Arabian Journal for Science and Engineering, Vol. 47, 2022, No. 8, pp. 9379–9392, doi: 10.1007/s13369-021-06193-3.

[31] Kumar, A.—Garg, G.: Empirical Study of Shallow and Deep Learning Models for Sarcasm Detection Using Context in Benchmark Datasets. Journal of Ambient Intelligence and Humanized Computing, Vol. 14, 2023, No. 5, pp. 5327–5342, doi: 10.1007/s12652-019-01419-7.

[32] Poria, S.—Cambria, E.—Hazarika, D.—Vij, P.: A Deeper Look into Sarcastic Tweets Using Deep Convolutional Neural Networks. CoRR, 2016, doi: 10.48550/arXiv.1610.08815.

[33] Porwal, S.—Ostwal, G.—Phadtare, A.—Pandey, M.—Marathe, M. V.: Sarcasm Detection Using Recurrent Neural Network. 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), 2018, pp. 746–748, doi: 10.1109/ICCONS.2018.8663147.

[34] Shrivastava, M.—Kumar, S.: A Pragmatic and Intelligent Model for Sarcasm Detection in Social Media Text. Technology in Society, Vol. 64, 2021, Art. No. 101489, doi: 10.1016/j.techsoc.2020.101489.

[35] Huang, Y. H.—Huang, H. H.—Chen, H. H.: Irony Detection with Attentive Recurrent Neural Networks. In: Jose, J. M., Hauff, C., Altıngovde, I. S., Song, D., Albakour, D., Watt, S., Tait, J. (Eds.): Advances in Information Retrieval (ECIR 2017). Springer, Cham, Lecture Notes in Computer Science, Vol. 10193, 2017, pp. 534–540, doi: 10.1007/978-3-319-56608-5_45.

[36] Kumar, P.—Sarin, G.: WELMSD – Word Embedding and Language Model Based Sarcasm Detection. Online Information Review, Vol. 46, 2022, No. 7, pp. 1242–1256, doi: 10.1108/OIR-03-2021-0184.

[37] Hengle, A.—Kshirsagar, A.—Desai, S.—Marathe, M.: Combining Context-Free and Contextualized Representations for Arabic Sarcasm Detection and Sentiment Identification. CoRR, 2021, doi: 10.48550/arXiv.2103.05683.

[38] Zhang, Y.—Liu, Y.—Li, Q.—Tiwari, P.—Wang, B.—Li, Y.—Pandey, H. M.—Zhang, P.—Song, D.: CFN: A Complex-Valued Fuzzy Network for Sarcasm Detection in Conversations. IEEE Transactions on Fuzzy

Systems, Vol. 29, 2021, No. 12, pp. 3696–3710, doi: 10.1109/TFUZZ.2021.3072492.

[39] Kamal, A.—Abulaish, M.: CAT-Bigru: Convolution and Attention with Bi-Directional Gated Recurrent Unit for Self-Deprecating Sarcasm Detection. Cognitive Computation, Vol. 14, 2022, No. 1, pp. 91–109, doi: 10.1007/s12559-021-09821-0.

[40] Naseem, U.—Razzak, I.—Eklund, P.—Musial, K.: Towards Improved Deep Contextual Embedding for the Identification of Irony and Sarcasm. 2020 International Joint Conference on Neural Networks (IJCNN), IEEE, 2020, pp. 1–7, doi: 10.1109/IJCNN48605.2020.9207237.

[41] Almeida, F.—Xexéo, G.: Word Embeddings: A Survey. CoRR, 2019, doi: 10.48550/arXiv.1901.09069.

[42] Joulin, A.—Grave, E.—Bojanowski, P.—Douze, M.—Jégou, H.—Mikolov, T.: FastText.zip: Compressing Text Classification Models. CoRR, 2016, doi: 10.48550/arXiv.1612.03651.

[43] Hochreiter, S.—Schmidhuber, J.: Long Short-Term Memory. Neural Computation, Vol. 9, 1997, No. 8, pp. 1735–1780, doi: 10.1162/neco.1997.9.8.1735.

[44] Hameed, Z.—Garcia-Zapirain, B.: Sentiment Classification Using a Single-Layered BiLSTM Model. IEEE Access, Vol. 8, 2020, pp. 73992–74001, doi: 10.1109/ACCESS.2020.2988550.

[45] Dey, R.—Salem, F. M.: Gate-Variants of Gated Recurrent Unit (GRU) Neural Networks. 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), 2017, pp. 1597–1600, doi: 10.1109/MWSCAS.2017.8053243.

[46] Misra, R.—Arora, P.: Sarcasm Detection Using Hybrid Neural Network. CoRR, 2019, doi: 10.48550/arXiv.1908.07414.

[47] Misra, R.—Grover, J.: Sculpting Data for ML: The First Act of Machine Learning. 2021.

[48] Akula, R.—Garibay, I.: Interpretable Multi-Head Self-Attention Architecture for Sarcasm Detection in Social Media. Entropy, Vol. 23, 2021, No. 4, Art. No. 394, doi: 10.3390/e23040394.

Isha Gupta is a research scholar at the Manav Rachna International Institute of Research and Studies, Faridabad, India. She has past 11 years of teaching experience at the Vivekananda College, University of Delhi. Her research areas include sentiment analysis, data mining, computer networks, software testing, and recommender systems. She has authored and co-authored research papers in journals and conference proceedings in the area of data mining.

Indranath Chatterjee is working as Professor in the Department of Computer Engineering at Tongmyong University, Busan, South Korea. He is also serving as Adjunct Professor at the School of Technology, Woxsen University, India. He is also involved in the Collaborating Faculty at the International Centre for Neuroscience Research, Tbilisi, Georgia. He received his Ph.D. in computational neuroscience from the Department of Computer Science, University of Delhi, Delhi, India. His research areas include computational neuroscience, schizophrenia, sentiment analysis, medical imaging, and deep learning. He has published 11 books on computer science and neuroscience published by renowned international publishers. To date, he has published numerous research papers in international journals and conferences. He is a recipient of various global awards in neuroscience. He is currently serving as a Chief Section Editor of a few renowned international journals and serving as a member of the Advisory Board and Editorial Board of various international journals and Open-Science organizations worldwide. He is presently working on several projects of government and non-government organizations as PI/co-PI, related to medical imaging and machine learning for a broader societal impact, in collaboration with several universities globally. He is an active professional member of the ACM, OHBM, FENS, ACNM, INCF, etc.

Neha Gupta has done her Ph.D. at the Manav Rachna International University and has a total of 16 years of experience in teaching and research. She is a Life Member of ACM CSTA, Tech Republic and a professional member of IEEE. She has authored and co-authored 70 research papers in SCI/Scopus/peer-reviewed journals and conference proceedings in areas of web content mining, mobile computing, and cloud computing. She has published books with publishers such as Springer, Taylor & Francis, IGI Global and Pacific Book International and has also authored book chapters with Elsevier, Springer, CRC Press, and IGI Global, USA. Her research interests include ICT in rural development, web content mining, cloud computing, data mining, and NoSQL databases. She is an active reviewer for the International Journal of Computer and Information Technology and at various IEEE conferences around the world.