Arabian Journal for Science and Engineering (2024) 49:3719–3734
https://doi.org/10.1007/s13369-023-08170-4

RESEARCH ARTICLE-COMPUTER ENGINEERING AND COMPUTER SCIENCE

Deep Contextualised Text Representation and Learning for Sarcasm Detection

Ravi Teja Gedela[1] · **Ujwala Baruah**[1] · **Badal Soni**[1]

Received: 13 June 2022 / Accepted: 24 July 2023 / Published online: 14 August 2023

Abstract
These days, the world has become a non-denominational place due to cyberspace. Digital services such as search portals and social platforms have become intricate with the regular course of daily life. As a massive number of clients' thoughts are available on the internet, sentiment analysis has become one of the most prolific investigation arenas in natural language processing. Sarcasm detection is a vital phase in sentiment analysis because of the intrinsically vague nature of sarcasm. To subsidize as a resolution to the current ever-increasing arena of attention on the ever-increasing volume of digital data, this work proposes a novel voting-based ensemble approach for sarcasm detection utilizing deep learning techniques. This paper uses Bidirectional Encoder Representations from Transformers to build contextual word embeddings and feeds those to the network, which is an ensemble of four deep learning models. Convolutional Neural Network, Bidirectional Long Short-Term Memory, and parallel and sequential combinations of both are among the models. For classification, the acquired features from each model are experimented with four machine learning classifiers, such as Support Vector Machine, Least Squares Support Vector Machine, Multinomial Naive Bayes, and Random Forest, along with Sigmoid activation function. Out of these five, the optimal classifier is selected for each model. Finally, a majority voting is performed on the outputs of all four models to differentiate between sarcastic and non-sarcastic texts. The proposed approach has been assessed using two benchmark datasets containing English-language texts, namely, news headlines and the self-annotated reddit corpus. Experiments demonstrated that the proposed methodology produces an accuracy of 94.89% on the news headlines repository, which is a gain of 2.99%, and an $F1$-score of 80.49% on the self-annotated reddit corpus, which is an improvement of 0.52% over the previous state-of-the-art methodologies.

Keywords Sentiment analysis · Sarcasm detection · Contextual word embeddings · Majority voting

1 Introduction

The views of others have a substantial influence on daily decision-making. These choices vary from purchasing a product such as a laptop, making investments, and selecting a school, all of which affect multiple aspects of everyday life. Moreover, in the online world, it becomes much easier to acquire various opinions from people around the globe. People use e-commerce sites, online review sites, and social media to get responses on how well a specific product/service is considered in the real economy. People are voicing their concerns to strangers online via blogs and social media forums [1–3].

Administrations and major corporations utilize this to assess the pulse of people on a number of topics, including images, goods, and politics. As an outcome, sentiment analysis (SA) on online data has recently piqued the public's interest. SA aims at identifying polarity by categorizing emotions or sentiments from data into positive, neutral, and negative categories. Sentiment polarity involves determining an individual's reactions to a commodity, allowing entrepreneurs to take countermeasures to meet the expectations and demands of their customers. Likewise, criticism of government services helps authorities examine public necessities and make crucial judgments [4–6]. Even if we do not like something, we can provide constructive comments, and sarcasm is one technique to convey ill feelings through positive words. Taking into consideration rhetoric literature,

U. Baruah and B. Soni have contributed equally to this work.

✉ Ravi Teja Gedela
ravi_rs@cse.nits.ac.in

1 Department of Computer Science and Engineering, National Institute of Technology, Silchar, Assam 788010, India

sarcasm is basically a kind of irony utilised with a cutting accent aimed at a victim [7, 8]. Sarcasm can frequently be seen on microblogging and online media because it encourages others to criticise or troll.

Starting with the earliest work on sarcasm detection (SD) in a speech by Tepperman et al. [9], the area has seen widespread interest. It is regarded as the most challenging and complicated issue in the natural language processing (NLP) community. Detecting sarcasm in the text takes on a variety of data representations (tweets, TV dialogues, and reviews). Sarcasm can switch the polarity of an utterance, so identifying and dealing with it accurately is critical in an automated NLP system. “Having a pricey mobile with a speedy battery drain is a lovely feeling," for example, is a sarcastic phrase expressing negative views regarding battery life. Still, affirmative adjectives like “lovely feeling" give the text a positive tone. As a result, it is critical to reveal the presence of sarcasm to enhance SA.

Previous research on forecasting sarcastic sentences has mainly concentrated on rule-based as well as statistical strategies that use (a) linguistic and pragmatic features, (b) interjections, sentiment shifts, punctuations, and so on [10]. Instead of just using hand-crafted features, a deep learning (DL) model learns the features needed automatically. DL architectures have evidenced state-of-the-art results in various NLP tasks, namely text summarization [11], machine translation [12], and question answering [13]. DL techniques have also been investigated for SD, with impressive results. Attention mechanisms aid in the performance of DL, as demonstrated by machine translation [14], text summarization [15], as well as improved reading comprehension [16]. Furthermore, the attention mechanism has been investigated for text classification [17] as well.

We attempted to convey the following shortcomings in the previous studies:

1. Previously, researchers have spent enormous amount of time on the text pre-processing phase, primarily on massive datasets [18, 19].
2. Many studies scrape online sources using APIs to develop their datasets rather than assessing their approaches on benchmark corpora. As a result, comparing and assessing their models is not justified [18, 20].
3. Many approaches look for words in significant dictionaries, which requires patience and seems impractical [21].

In this article, we introduce a novel voting-based ensemble approach using Bidirectional Encoder Representations from Transformers (BERT) contextualized word embeddings and DL models. BERT is a model that is increasingly becoming popular due to its exceptional performance in various NLP tasks [22, 23]. We use BERT as a word encoder, which has the capability to adequately represent context of each word in a sentence. By taking word embeddings (sequence output) from BERT, we propose an ensemble network that includes four DL models, i.e. Convolutional Neural Network (CNN), Bidirectional Long Short-Term Memory (BiLSTM), and parallel and sequential combinations of both, which we used to acquire more contextualised features. For each model to generate output, the acquired contextual features have been experimented with five classifiers, of which one is the sigmoid activation function and the other four are machine learning (ML) classifiers (support vector machine (SVM), least squares SVM (LSSVM), multinomial Naive Bayes (MNB), and random forest (RF)). Out of five classifiers, we select the optimal classifier for each model. Finally, a majority voting ensemble mechanism is employed among all the models presented. The below are the major contributions of the current work:

1. We proposed a voting-based ensemble approach with four models that can learn sarcastic patterns and improve accuracy. This ensemble approach aided us in developing an effective sarcasm recognition solution.
2. Using two publicly available datasets containing English-language texts for training and validation, we discovered that our proposed approach not only enhanced sarcasm identification but also offered better insights overall.
3. Four approaches of word embedding, including non-contextual (Word2vec, GloVe, fastText) and contextual (BERT), are examined for their applicability and effectiveness.
4. We presented a comprehensive analysis of the experimental results and compared the performance of the proposed approach to that of other state-of-the-art approaches for recognizing sarcasm.

The remainder of the paper is laid out as follows. In Sect. 2, we discuss the literature, and in Sect. 3, we elucidate the methodology of the proposed approach. The experimental configuration, determinations, and a comparison with prior studies are all included in Sect. 4. Section 5 concludes the paper and makes recommendations for further research.

2 Related Work

Aside from the importance of SA in industry-related applications, sarcasm and irony detection have also received considerable attention in the NLP research group, leading to a variety of approaches and benchmark repositories. There are three kinds of approaches to recognizing sarcasm in particular: rule-based, statistical, and neural (DL)-based. The subsections that follow discuss earlier research on the approaches mentioned.

Table 1 A quick recapitulation of the features used primarily in earlier research

Research paper	Features used
Davidov et al. [28]	Pattern-based and Punctuation-based (like the number of words, capital words, “!" and “?" quotes in a sentence)
Tsur et al. [29]	Pattern-based
González-Ibánez et al. [30]	Features based on lexicon and those that are pragmatic (like user mentions and emoticons)
Reyes et al. [31]	Features like ambiguity, unexpectedness, emotional scenarios, and so on
Maynard and Greenwood [26]	There are seven feature groups. These include the maximum/minimum/gap of frequency of adverbs and adjectives, the maximum/minimum/average number of synonyms, and word synsets
Barbieri et al. [32]	Features like structure, written-spoken, frequency, intensity, synonyms, sentiments, and ambiguity
Liu et al. [33]	English features include punctuations and syntactic and lexical features. Chinese features include homophonic, rhetorical, and constructional elements
Joshi et al. [34]	Incongruity-based features, both implicit and explicit
Rajadesingan et al. [35]	The amount of flips, word extensions, and readability features
Fersini et al. [36]	PoS tags and Pragmatic particles
Hernández-Farías et al. [37]	Features using similarity(Wordnet-based) are used to demonstrate semantic similarities between words
Bamman and Smith [38]	Author, Tweet, audience, and environment-related features
Altrabsheh et al. [39]	Unigram, n-gram, and other features like polarity label, hashtag count, emotion label, etc
Bouazizi and Ohtsuki [40]	Punctuation-related, syntactic-semantic combinations, sentiment-related, and pattern-related
Lukin and Walker [41]	Pattern-based

2.1 Rule-Based Approaches

These methods rely on specific evidence to determine sarcasm. This evidence is represented by rules based on sarcasm characteristics. In this section, we will go over some of the previous progress made using rule-based techniques.

Riloff et al. [24] suggest a rule-based classifier that aims to detect a category of sarcasm that will be significant in tweets: “positive sentiment contrasted with a negative situation". They demonstrate a bootstrapped learning technique for attaining collections of positive sentiment utterances and negative activities, and how these collections can also be used to discover sarcastic tweets. Kreuz and Caucci [25] investigate the impact of various linguistic factors in recognition of sarcasm in narratives, like interjections (e.g. “oh" and “ooh") and punctuation (e.g. “!" and “?"). Maynard and Greenwood [26] looks into the use of sarcasm in tweets, specifically its impact on SA. They concentrate on hashtags as part of the analysis. They have created a hashtag tokenizer for GATE [27] to detect sentiment and sarcasm within hashtags. They have also formulated a set of regulations to boost SA accuracy when sarcasm exists.

2.2 Statistical Approaches

In particular, statistical models for SD differ significantly in terms of features & learning algorithms. We will go over these two within these subheadings.

Features The majority of existing approaches make use of bag-of-words as a feature. Moreover, there are some peculiar features suggested in various works. The features used in some statistical techniques are summarised in Table 1. This section focuses on features pertinent to the sentence to be classified.

Learning Algorithms Many different classifiers were tested for SD. Because of its capacity, SVM is used in the more significant part of the SD task [9, 25, 29, 34, 42–45]. Reyes and Rosso [46] employ SVM, Decision Trees (DT), and Naive Bayes (NB). They also demonstrate the Jaccard similarity of features and labels. Riloff et al. [24] contrast rule-based with SVM classifier. Sarsam et al. [47] looked at ML algorithms for SD on twitter and discovered that RF, SVM, NB, and Logistic Regression (LR) seem to be the most commonly used classifiers for forecasting sarcasm on twitter. The SVM, k-nearest neighbour (KNN), Maximum Entropy, and RF classification algorithms used by Bouazizi and Ohtsuki [40]. They noticed that a RF classifier outperformed all other models tested. Suykens and Vandewalle [48] proposed LSSVM in June of 1999. They are a least-squares variant of SVMs, a family of related supervised learning algorithms for regression and classification problems that analyse data and recognize patterns. However, little research reported using LSSVM in NLP.

2.3 Deep Learning-Based Approaches

DL-based architectures have gained popularity in NLP applications based on their ability to manage data sparsity in imbalanced datasets. A few such approaches to automatic SD have also been reported.

In order to discover sentimental content on twitter, Felbo et al. [49] present a DeepMoji framework based on emoji occurrences. To identify sarcasm in tweets, they have employed an LSTM variant of a 6-layer design, which combines BiLSTM with an attention mechanism. Mehndiratta and Soni [50] trained CNN, LSTM, and combinations of both CNN and LSTM on the reddit repository by utilizing fastText, and GloVe embeddings to examine the effectiveness using different epochs, training sizes, and dropouts on performance. Using punctuation-based features and GloVe embeddings, Kumar et al. [19] present a novel DL model called sAtt-BLSTM convNet, a mix of BLSTM based on soft attention (sAtt-BLSTM) and CNN (convNet). This design has an 8-layered architectural style and attained 97.87% accuracy on the twitter dataset and 93.71% accuracy on the random-tweet repository.

In recent SD studies, BERT has been used to give deep contextual embeddings for words. This concept is gaining traction due to its impressive performance in different NLP tasks. Shrivastava and Kumar [51] present a model that employs a variant of BERT to judge whether the text is sarcastic or not. They tested their strategy against Bi-LSTM, LSTM, attention models, LR, and SVM and observed that using transformer-based representations enhances model performance. Gregory et al. [52] use BERT+LSTM, LSTM model in conjunction with attention, and numerous different transformer models (XLNet, RoBERTa, ALBERT, and BERT) to find sarcasm in both reddit and twitter datasets. The ensemble approach using transformers outperformed all other models tested.

In addition, there were substantial repositories for performing experimentations on sarcasm recognition. Primarily, most researchers gathered their datasets mainly from twitter [20, 24, 32–34, 38, 53, 54], amazon [33, 46, 55], and reddit [56, 57]. However, few other repositories are also available, which were captured from tv shows [58], news headlines [59], etc.

3 Methodology

This paper proposes a novel voting-based ensemble approach to detect and classify sarcasm. The proposed methodology comprises three stages: pre-processing, feature extraction, and classification. A detailed description of these stages is given below:

3.1 Pre-processing

The aim of this step is to preprocess the raw textual data and arrange it for the next step of extracting useful features. We preprocessed both datasets using Natural Language Toolkit (NLTK), and the steps we took as part of that are as follows:

Handling Hashtags We detached # from the hashtag after recognising it with the pound (#) symbol. **Example** #TheDreamBoys → TheDreamBoys.

Stemming Using Snowball stemmer (called Porter2), we stripped inflectional morphemes like “ing," “ed," “s," and “est" from their token stem. **Example**: Words like “created," “creating," and “creates" are reduced to “create".

Text Cleaning We carried out this step to remove useless data. We stripped URLs, duplicate text, non-ASCII glyphs, special characters, and stop words.

As there were no hashtags in news headlines, we skip that step when we perform preprocessing on news headlines dataset. To get better contextualized word embeddings when experimenting with BERT, we did not perform stemming and stopword removal.

3.2 Feature Extraction

Because of the difficulties involved in representing both semantic and syntactic information for data points/texts, expressing utterances in an acceptable format for machines has always been an area of investigation in the domain of NLP. TF-IDF vectorization, One-Hot Encoding, and word embedding are the most popular approaches for this task. Word embedding/word vector is an input numeric vector that illustrate a word in a low-dimensional space while preserving both syntactical and semantical meaning. This is essentially a technique for gleaning features from text for use as inputs to ML/DL model. Our research employs word embedding since this surpasses TF-IDF vectorization and One-Hot Encoding.

3.2.1 Static Word Embeddings

Static word embedding techniques include Word2Vec, GloVe, and fastText.

Table 2 Configurations of BERT's base and large versions

BERT version	Number of layers	Hidden size	Attention heads	Number of parameters
BERT-base	12	768	12	110 M
BERT-large	24	1024	16	340 M

Word2vec Mikolov et al. [60] of Google introduced Word2vec in 2013. A vast corpus of datasets was used to train this unsupervised learning model. Word2vec has a relatively smaller dimension as compared to one-hot encoding.
GloVe Pennington et al. [61] demonstrated GloVe, a mathematical methodology for producing meaningful word representations by building a new matrix, i.e. the count of word-word co-occurrences, and computing a probability matrix to estimate the likelihood of two words appearing together.
fastText This method [62] uses the skip-gram concept, where a bag of character n-grams denotes each word. FastText, unlike Mikolov's embeddings, can have an embedding for misspelled words, rare words, or utterances that did not appear in the training repository.

3.2.2 Contextual Word Embeddings

Although the embeddings mentioned earlier can capture semantic information from words, a static vector in these models represents each word. As a result, words with distinct meanings in different sentences have the same embedding. Researchers have developed various word representation techniques based on transformers that can take contextual information into account when constructing embedding vectors for the same words in different situations. BERT is a multi-layer bidirectional transformer encoder framework designed by Devlin et al. [63].

The BERT developers primarily supplied two models (base and large), which vary in configurations with respect to the count of transformer blocks, hidden layer size, and attention heads (as shown in Table 2). In practice, BERT creates a grouped illustration for the text document and provides an embedding for every token, which is helpful for the classification problem. As shown in Fig. 1, the BERT input for every word is calculated by adding the token embeddings, segment embeddings, and positional embeddings. The success

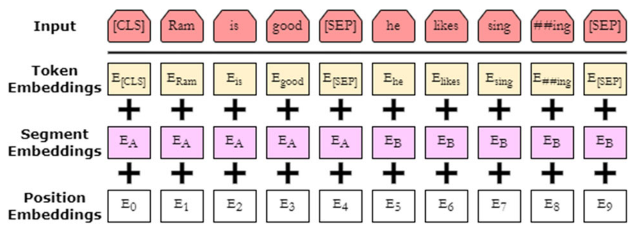

Fig. 1 BERT input embedding

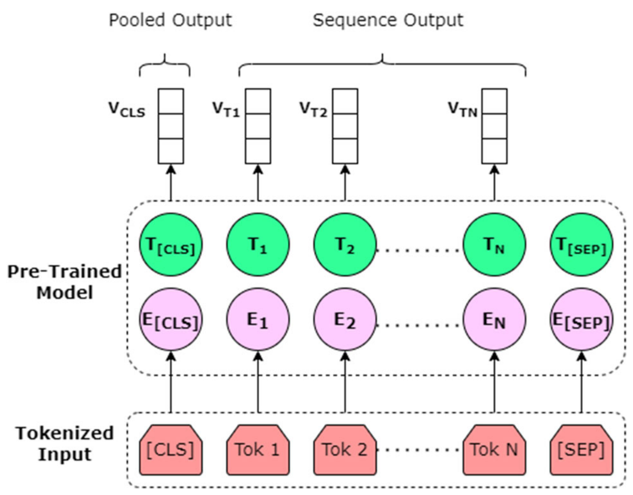

Fig. 2 Sequence and pooled outputs from BERT

of BERT in a variety of text processing tasks, namely named entity recognition [64], text classification [65], open-domain QA [66], machine translation [67], and text summarization [68], inspired us to use this framework to produce context-specific depictions in the sarcasm identification task.

There are primarily two ways to utilize BERT. One way is to "fine-tune" BERT. We add layer(s) on top of BERT and train the entire system together afterward. We can train our additional layer(s) while adjusting (fine-tuning) the BERT weights. The other way is to use it as a "feature extractor." In other words, we utilize BERT's outcome as input to subsequent models. This way, we are "acquiring" features from text using BERT and using them in a distinct method for the actual task. The BERT "feature extraction" strategy yields two types of vectors. The first is sequence output, which is a matrix of token embeddings. Another type is pooled output, an embedding vector that describes the overall input text. Figure 2 shows both the sequence and the pooled outputs.

3.3 Classification

The proposed model has three steps in total as shown in Fig. 3.

1. In the first step, we made use of the techniques discussed in Sect. 3.1 to preprocess the raw data. The BERT's tokenizer is used to transform texts into tokens. The tokens are further fed into the BERT model, which comprises 12 layers of transformer encoders linked together. We utilize sequence output in this case, which is an embedding

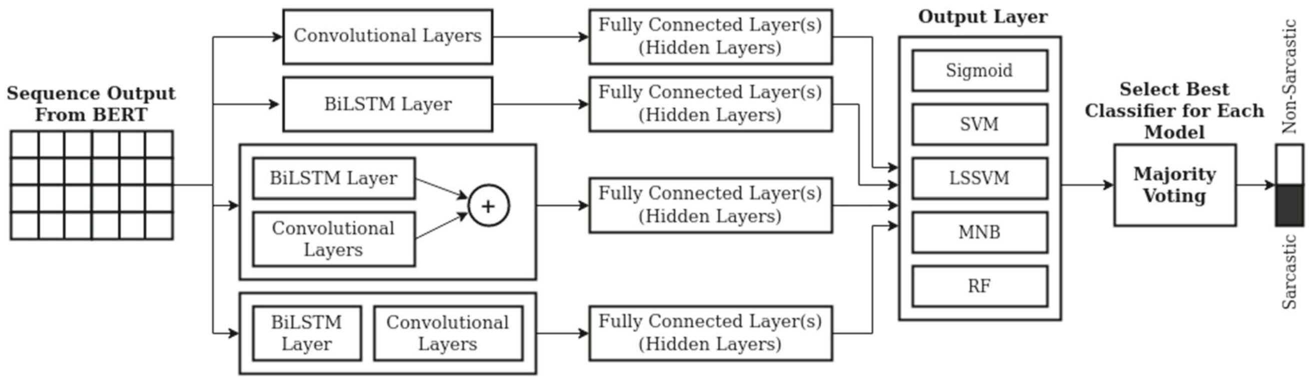
Sequence Output From BERT
Convolutional Layers
BiLSTM Layer
BiLSTM Layer
Convolutional Layers
+
BiLSTM Layer
Convolutional Layers
Fully Connected Layer(s) (Hidden Layers)
Fully Connected Layer(s) (Hidden Layers)
Fully Connected Layer(s) (Hidden Layers)
Fully Connected Layer(s) (Hidden Layers)
Output Layer
Sigmoid
SVM
LSSVM
MNB
RF
Select Best Classifier for Each Model
Majority Voting
Non-Sarcastic
Sarcastic

Fig. 3 Proposed methodology for sarcasm detection

vector for all tokens. The matrix of size n * d represents sequence output, where n is the token count in the input sequence and d is the embedding dimension. A d-dimensional embedding vector expresses a token in each row of the embedding matrix. DL models (like CNN, LSTM, GRU, etc.) can use the BERT's sequence output as input.

2. In the second step, we use an ensemble of simple (CNN, BiLSTM) and hybrid (parallel and sequential) DL models. They are:

   - BERT_CNN
   - BERT_BiLSTM
   - BERT_CNN + BERT_BiLSTM
   - BERT_BiLSTM_CNN

   In all four models, the output units of the final layer are the assessed probabilities associated with the input sample. The probability of each output is computed by an activation function. The activation function takes as its input a linear composite of trainable weights from the preceding hidden layer's outputs and a bias term. Not only do the features of the hidden layer make sense to the DL model, but they can also serve as input features for another classifier [69]. So, along with the sigmoid activation function at the output layer, we experimented with four ML classifiers (SVM, LSSVM, MNB, and RF). This lets us figure out the optimal classifier for each DL model.
3. We apply majority voting among the outputs of all DL models in the third step. A voting mechanism aims to assign a class label to a text relying on a majority vote using weights applied to the class/class probabilities.

3.3.1 Simple Models

BERT_CNN

Our first DL framework focuses on the CNN architecture. It has a powerful deep architecture, usually beginning with convolutional and then pooling/subsampling layers, transforming inputs into fully connected classification layers. CNN's broad versatility in computer vision problems encouraged the idea of applying it to NLP problems. CNN in NLP focuses on extracting major features from word collocation in a sentence. We have attached the sequence outcome of pre-trained models to a CNN to derive many features about the nearest neighbours of tokens in a sentence. We use 1D-CNN (single convolutional) in this analysis.

Our CNN framework incorporates convolution, max-pooling and average pooling layers to learn the underlying meaningful patterns. As shown in Fig. 4, our CNN architecture comprises three parallel convolutions, each having 128 filters with kernel sizes of 3, 5, and 7. Each convolutional layer's yield flows to a max-pooling layer, and indeed, the outcomes of all max-pooling layers concatenate into a single feature vector containing newly extracted features. The retrieved vector is then given to a convolution layer of kernel size 5 with 256 filters. This is preceded by average pooling, and lastly, the extracted feature vector is transmitted into the fully connected layer(s). Finally, we utilized fully connected dense layers with 0.3 dropout to prevent overfitting.

BERT_BiLSTM

BiLSTM is used in our second technique, which analyses the input text by remembering the semantics of preceding and upcoming tokens. Such a kind of RNN can grasp contextual information and long-term dependencies. The LSTM structure consists of recurrently arranged memory blocks with three multiplicative gates: an input gate, a forget gate and an output gate in each memory cell. These gates fix the vanishing gradient issue, which can use, store, and forget information for longer times.

Our BiLSTM has a hidden state dimension of 128 for both(forward as well as backward) layers, as illustrated in Fig. 5. The output of the bidirectional LSTM is sent to average pooling and subsequently to a fully connected layers. Finally, to avoid overfitting, we have used fully connected dense layers with a dropout of 0.3.

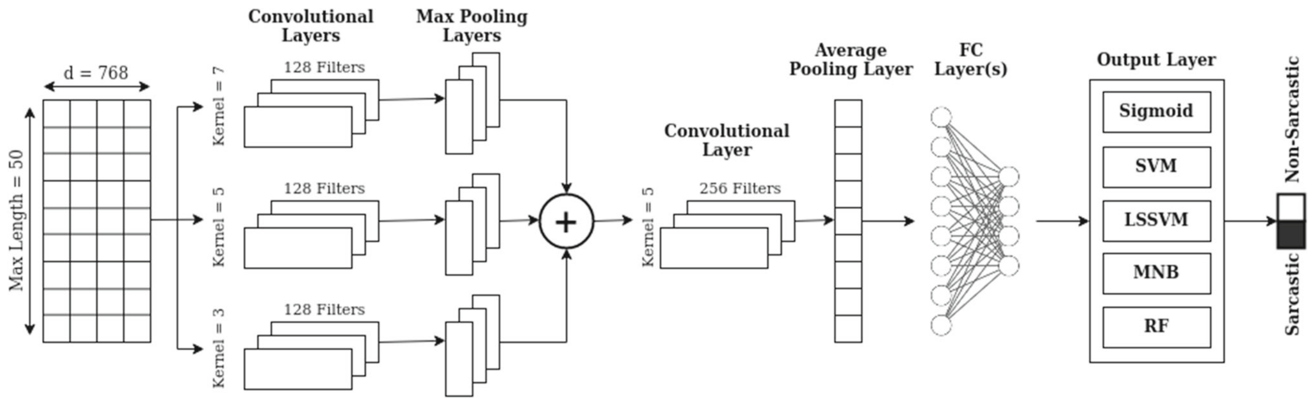

Fig. 4 Overview of BERT_CNN architecture

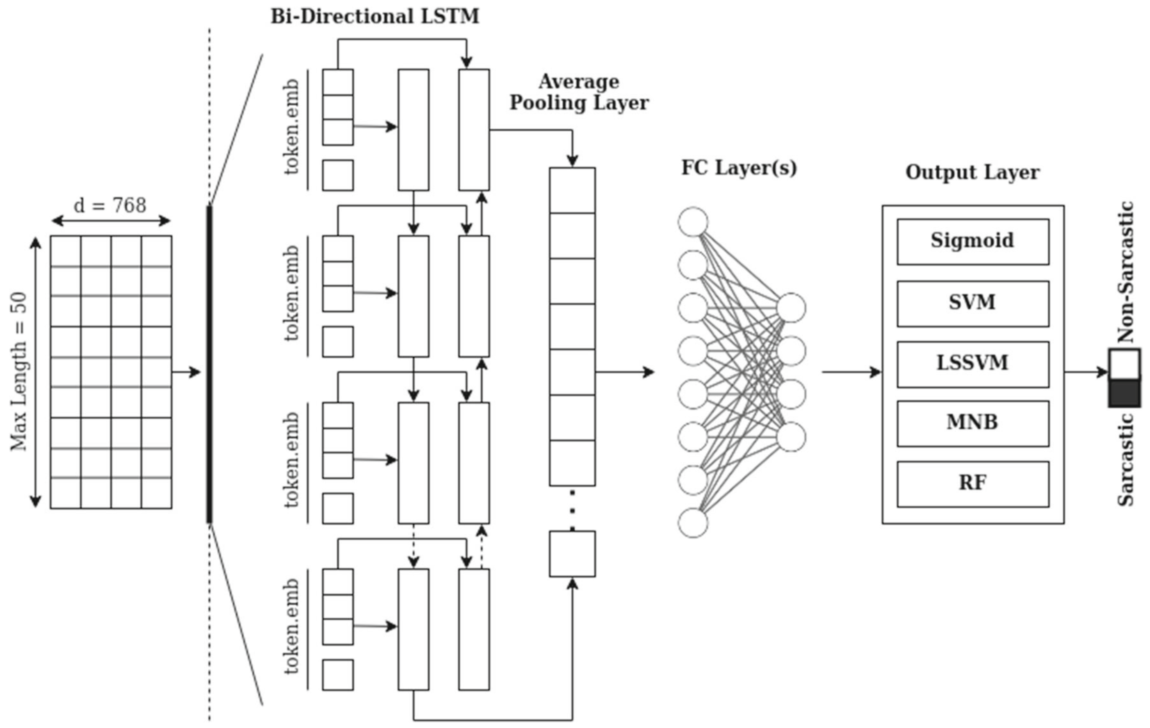

Fig. 5 Overview of BERT_BiLSTM architecture

3.3.2 Hybrid Models Combining CNN and BiLSTM

The hybrid models are a combination of the simple models discussed above. CNN deals with spatially relevant data, whereas RNNs excel at coping with temporal signals. LSTM can remember and forward information in a sequence, whereas multi-layer CNN can grab and learn enough local information. So, the combination uses the best of both worlds, the spatial and temporal worlds. The mixture of these features yielded from both CNN and BiLSTM provides more valuable information for classification [70].

BERT_CNN + BERT_BiLSTM

To have both local and contextual features from the text, we used a parallel mixture of the convolutional and BiLSTM layers within our third approach. With this parallel composition, we can effectively capture and concatenate both sets of features from the word embedding interpretation of the text.

Figure 6 depicts our structure, consisting of convolution layers of 128 filters with window sizes of 3, 5, and 7, a max-pooling layer, and the concatenation of all extracted features. The outcome is then passed through a convolution operation of 256 filters with a window size of 5, preceded by an average

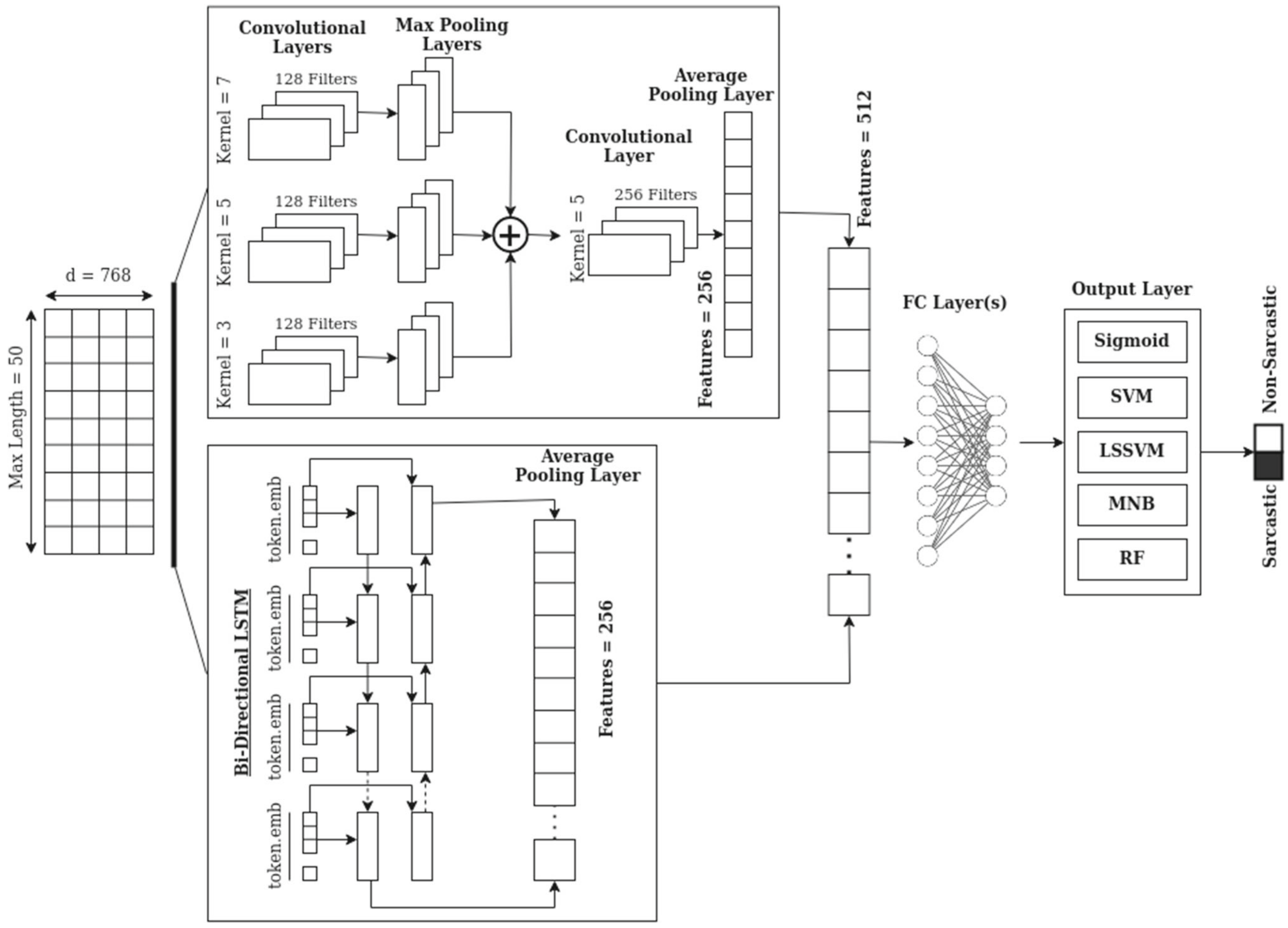

Fig. 6 Overview of BERT_CNN + BERT_BiLSTM architecture

pooling layer to yield the local features. Our architecture also includes a BiLSTM layer with 128 neurons, preceded by an average pooling layer, to gain temporal features. As a result, 512 features (256 features each from both BiLSTM and CNN) are retrieved from both models and fed into the subsequent layer. Finally, we utilized fully connected dense layers with 0.3 dropout to prevent overfitting.

BERT_BiLSTM_CNN

We combine the BiLSTM and CNN models sequentially in the final hybrid model. We have made use of a framework intending to retrieve local features from an abstract representation offered by a BiLSTM layer to accomplish this. Context information from the texts is included in this abstract representation.

As shown in Fig. 7, we feed each text's word embedding into a BiLSTM layer with a 128-state hidden dimension. This layer yields the sequence derived from the previous time step and the entire sequence of features with information from each time step. After acquiring the whole sequence of hidden states, we have employed a CNN to gather local information between various time intervals. Then, 128 filters with window sizes of 3, 5, and 7 are used to collect the local features and use a max-pooling layer to furnish more pertinent features. The features extracted from each max pooling layer are merged. A convolution layer of 256 filters precedes this with a window size of 5 and an average pooling layer. Finally, to avoid overfitting, we have used fully connected dense layers with a dropout of 0.3.

4 Experimental Results and Discussion

In this section, we discuss the results of the experiments carried out to compare the efficacy of the proposed approach with baselines and other state-of-the-art approaches. They are all tested using the benchmark datasets described in Sect. 4.1 and preprocessed using the text processing techniques described in Sect. 3.1. Section 4.2 discusses the system configuration for running our experiments. Also, this subsection provides a detailed explanation of the model parameters and their impact on model performance. Section 4.3 presents, discusses, and interprets the findings.

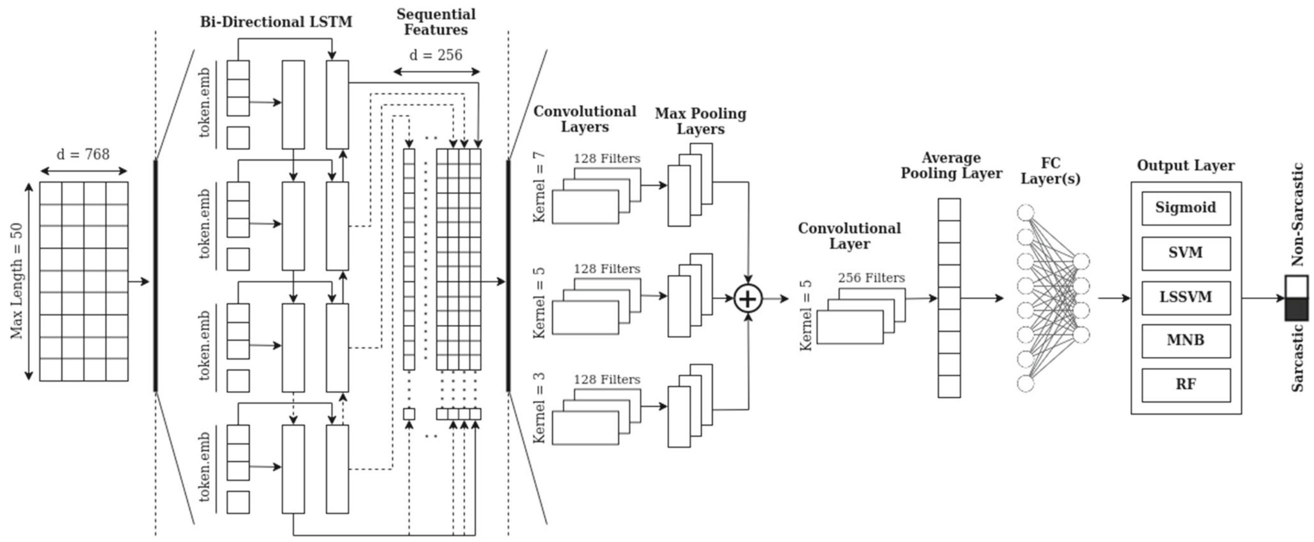

Fig. 7 Overview of BERT_BiLSTM_CNN architecture

4.1 Datasets

For this work, we have used two independent benchmark corpora containing texts in English: one with news headlines and the other with user comments from the social website, Reddit. We provide an overview of these two repositories in this subsection.

4.1.1 News Headlines Dataset

To overcome the downsides of tweets, Misra and Arora [59] developed news headlines repository from two journal websites: HuffPost and TheOnion. The repository is in JSON format and has been retrieved from Kaggle.[1] It has 28,619 headlines, 13,635 of which are sarcastic and 14,984 of which are not, and is represented by three columns. One comprises the labels 1 and 0 that indicate if the headline is sarcastic/non-sarcastic. The other two contain news headlines and the URL for the headline. The intention is to assess whether or not a headline is sarcastic; hence, we omitted the URL. The repository does not contain any null values, indicating complete data. All of the sarcastic headlines are from TheOnion, not HuffPost.

4.1.2 SARC Dataset

Khodak et al. [57] built SARC, a gigantic repository for the task of sarcasm identification. The dataset, which can be downloaded from Kaggle,[2] is a subset of Reddit comments posted from January 2009 to April 2017. There are ten columns in the dataset, out of which we have worked with two: comment and label (sarcastic or not). The dataset includes 1.3 M sarcastic comments and many more non-sarcastic comments. For experiments, we considered only the data points that had a comment length of at least 10 words. Therefore, the total number of comments is 4,41,637, of which 2,25,974 are sarcastic and 2,15,663 are not.

4.2 Experimental Setup

The proposed method is programmed in Python and uses TensorFlow, the most well-known interface for developing high-quality DL models [71]. The system configuration is Ubuntu 20.04.4 LTS (a 64-bit operating system) with an Intel Xeon (R) W-2133 processor, 64 GB RAM, and an NVIDIA Corporation TU104GL (Quadro RTX 4000) GPU.

One of the fundamental steps of any DL approach is the selection of desirable hyperparameters. There are two basic approaches to selecting the best numbers: automatic and manual selection. Both strategies are equally credible, but manual selection requires a thorough understanding of the model. Automatic selection necessitates a high computational cost. Table 3 shows the values of the hyperparameters we used in all of our experiments. We chose these values manually after seeing how the models worked with different combinations of hyperparameters.

The proposed models employ BERT-Base version, which has 12 attention heads and a hidden size of 768. Furthermore, we set the maximum sequence length as 50. To ensure that all comments/headlines are exactly 50 words long, sufficient zeros have been appended to the end of those that are less than 50 words. Similarly, comments/headlines containing over 50 words were truncated to the initial 50 words.

1 https://www.kaggle.com/datasets/rmisra/news-headlines-dataset-for-sarcasm-detection.

2 https://www.kaggle.com/datasets/danofer/sarcasm.

Table 3 Hyperparameters used for all the experiments

Hyperparameter	Value
Optimizer	Adam
Loss function	binary_crossentropy
Learning rate	5e–5
Batch size	32
Number of Epochs	30
Dropout	0.3
ModelCheckpoint	Yes
EarlyStopping	Yes
Patience	5

As a result of these modifications, the performance overall was unaffected, as there were only a few comments/headlines that had a word count of more than 50. Various learning rates (0.00002, 0.00003, 0.00004, and 0.00005) have been tested with the Adam optimizer, of which 0.00005 has been found to be the optimal value. We also tried with different dropouts, including 0.2, 0.25, 0.3, 0.35, and 0.4. The optimal value in this case is 0.3. We divide the preprocessed repository into training (80%), validation (10%), and testing (10%) and used the same sequence of training, validation, and testing data points across all the experiments.

4.3 Results and Discussion

The predictive performance of the built model has been assessed using accuracy (ACC), precision (PR), Recall (RC) and $F1$ score (FS). The equations (1)–(4) are used for calculating these metrics.

$$\mathrm{ACC} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}} \tag{1}$$

$$\mathrm{PR} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \tag{2}$$

$$\mathrm{RC} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \tag{3}$$

$$\mathrm{FS} = \frac{2*\mathrm{TP}}{2*\mathrm{TP}+\mathrm{FP}+\mathrm{FN}} \tag{4}$$

Where,

- **True Positives (TP)** is the number of sarcastic cases that were correctly identified.
- **False Positive (FP)** refers to the number of cases incorrectly classified as sarcastic.
- **True Negative (TN)** is the count of instances correctly predicted as non-sarcastic.
- **False Negative (FN)** is the count of instances incorrectly predicted as non-Sarcastic.

We have used static word representations such as Word2vec, GloVe, and fastText as a baseline. For this, we just replaced the BERT embeddings (sequence output) with Word2vec, GloVe, and fastText individually in the proposed model shown in Fig. 3. Two simple models: CNN and BiLSTM, and two hybrid models: CNN+BiLSTM and BiLSTM_CNN, respectively, are called parallel and sequential combinations. At the output layer for every model, we experimented with five classifiers separately, of which four are ML classifiers and one is a sigmoid activation function. This yields the best classifier at the output layer for each model, and the same combination has been used for the performance comparison with baselines.

For each model, the experiments have been carried out four times: once with sequence output from BERT, once with Word2vec, once with GloVe, and once with fastText. Tables 4, 5, 6, 7 detail the experimental findings of four DL models with BERT and static word embeddings. If only static embeddings are looked at, all the models with GloVe and fastText word embeddings produced almost the same results, and all models did better with GloVe embeddings. This may be due to GloVe’s emphasis on “word-to-word co-occurrences" across the entire corpus. We can observe that all four models with BERT word embeddings outperform the same models with static word embeddings by a large margin, which is approximately 5–6% on news headlines and 4–5% on SARC repositories, respectively. The reason could be due to BERT’s ability to produce the contextual embedding for each word in a text so that the semantic meaning is preserved. At the output layer, the SVM classifier for classification has produced good results in all four models on the news headline repository and also in two models on the SARC repository. Finally, majority voting has been applied to all four DL models with BERT embeddings. On both datasets, four models have shown that when BERT is used, the results are much better and even more sound when majority voting is applied. The findings of the proposed voting-based ensemble approach, along with four DL models (with BERT), are shown in Table 8.

In our deep investigation, we found that out of all the models shown on both datasets, the sequential combination on top of BERT embeddings (BERT_BiLSTM_CNN) with an SVM classifier at the output layer gives the best results. This combination produces an accuracy of 94.61% on the news headlines dataset, which is an improvement of nearly 1%, and 79.42% on the SARC dataset, which is a gain of 0.65% over all other models presented. On two datasets, both simple models built on top of BERT word embeddings (BERT_CNN and BERT_BiLSTM) produced almost similar outcomes. However, the parallel combination (BERT_CNN+BERT_BiLSTM) has improved accuracy, although not much. When we apply majority voting to all four models presented, the accuracy improves further by 0.28% on news headlines and 1.17% on SARC datasets,

Table 4 Results (%) of CNN with contextual (BERT) and static embeddings

Word embedding	News headlines			SARC		
	Classifier	Accuracy	*F*1-score	Classifier	Accuracy	*F*1-score
	Sigmoid	93.08	92.63	Sigmoid	76.46	76.80
	SVM	**93.15**	**92.74**	SVM	76.80	77.19
BERT	LSSVM	92.97	92.50	**LSSVM**	**77.05**	**77.43**
	MNB	93.04	92.58	MNB	76.74	76.70
	RF	93.08	92.62	RF	76.67	76.53
Word2vec	SVM	83.05	82.93	LSSVM	71.63	71.37
GloVe	SVM	86.93	86.27	LSSVM	72.84	72.69
fastText	SVM	86.40	85.22	LSSVM	72.53	72.20

The best results are shown in bold

Table 5 Results (%) of BiLSTM with contextual (BERT) and static embeddings

Word embedding	News headlines			SARC		
	Classifier	Accuracy	*F*1-score	Classifier	Accuracy	*F*1-score
	Sigmoid	92.90	92.31	Sigmoid	76.55	76.18
	SVM	**93.18**	**92.78**	**SVM**	**77.61**	**78.01**
BERT	LSSVM	93.15	92.75	LSSVM	77.33	77.46
	MNB	92.87	92.27	MNB	77.28	77.68
	RF	93.15	92.67	RF	77.13	76.80
Word2vec	SVM	83.75	83.52	SVM	71.87	72.02
GloVe	SVM	87.52	86.48	SVM	73.47	73.33
fastText	SVM	87.24	86.59	SVM	72.98	73.27

The best results are shown in bold

Table 6 Results (%) of CNN + BiLSTM with contextual (BERT) and static embeddings

Word embedding	News headlines			SARC		
	Classifier	Accuracy	*F*1-score	Classifier	Accuracy	*F*1-score
	Sigmoid	93.29	92.69	Sigmoid	77.52	77.24
	SVM	**93.57**	**93.05**	SVM	78.38	78.54
BERT	LSSVM	92.90	92.33	LSSVM	78.16	78.07
	MNB	93.29	92.69	MNB	77.76	78.65
	RF	93.46	92.92	**RF**	**78.77**	**78.90**
Word2vec	SVM	84.90	84.21	RF	72.37	71.03
GloVe	SVM	87.87	87.42	RF	74.01	73.87
fastText	SVM	87.49	86.47	RF	73.80	74.28

The best results are shown in bold

Table 7 Results (%) of BiLSTM_CNN with contextual (BERT) and static embeddings

Word embedding	News headlines			SARC		
	Classifier	Accuracy	*F*1-Score	Classifier	Accuracy	*F*1-Score
	Sigmoid	94.51	94.19	Sigmoid	78.99	78.69
	SVM	**94.61**	**94.39**	**SVM**	**79.42**	**79.37**
BERT	LSSVM	94.02	93.78	LSSVM	79.10	79.28
	MNB	94.58	94.26	MNB	78.40	79.24
	RF	94.54	94.34	RF	79.04	79.33
Word2vec	SVM	84.97	84.21	SVM	74.14	74.40
GloVe	SVM	88.74	87.95	SVM	75.20	75.05
fastText	SVM	88.57	87.60	SVM	74.85	75.06

The best results are shown in bold

Table 8 Results (%) of deep learning models along with proposed voting ensemble

Model combination	News headlines		SARC	
	Accuracy	*F*1-score	Accuracy	*F*1-score
BERT_CNN	93.15	92.74	77.05	77.43
BERT_BiLSTM	93.18	92.78	77.61	78.01
BERT_CNN + BERT_BiLSTM	93.57	93.05	78.77	78.90
BERT_BiLSTM_CNN	94.61	94.39	79.42	79.37
Proposed method	**94.89**	**94.60**	**80.59**	**80.49**

The best results are shown in bold

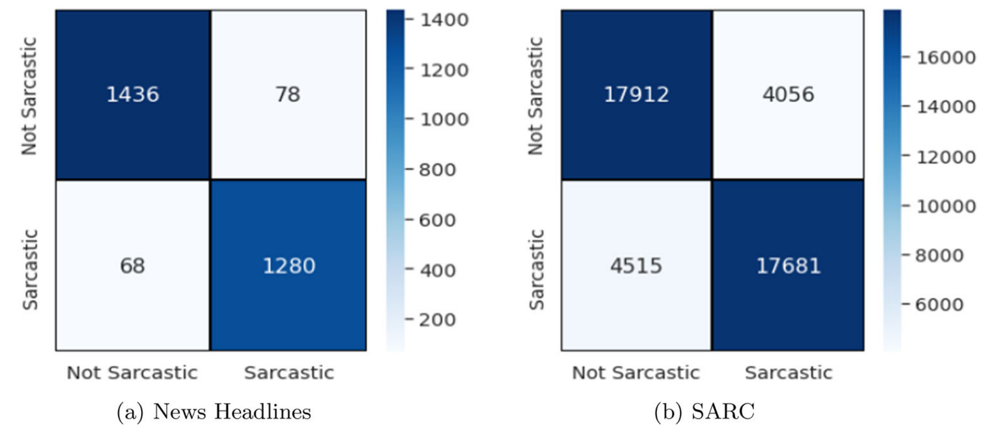

Fig. 8 Confusion matrices of proposed voting ensemble on both datasets

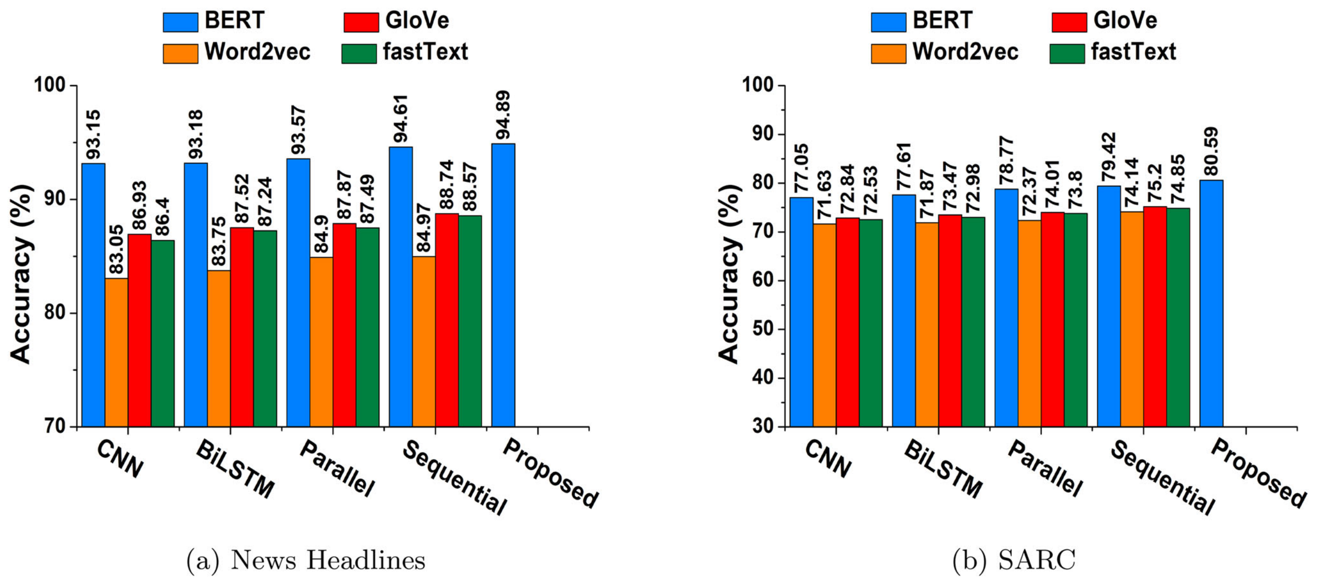

Fig. 9 Comparative performance of the proposed voting ensemble with deep learning models on both datasets

respectively. With the majority voting while making a prediction, we can lower the risk of a single model producing an incorrect prediction by having several models that are capable of producing accurate predictions. Figure 8 depicts the confusion matrices of the proposed voting-based ensemble approach on both datasets.

The performance of the proposed voting-based ensemble approach is compared to a few state-of-the-art strategies for identifying sarcasm. Table 9 presents the results of several earlier studies that used the same datasets that we did in our experiments.

We performed an error analysis by looking at a small number of false positives and false negatives that our method gave us to figure out where it fails the most. The few texts that our method misclassified are listed below, with their actual label ('NS' indicates non-sarcastic, and 'S' indicates sarcastic) shown in parentheses.

Table 9 Performance(%) comparison of the proposed method with other state-of-the-art methods

Research paper	Dataset	Accuracy	Precision	Recall	$F1$-score
Misra and Arora [59]	News headlines	89.7	–	–	–
Shrikhande et al. [72]		86.13	84.4	86.8	85.6
Pandey et al. [73]		–	88	88	88
Jamil et al. [74]		91.6	91	91	91
Akula and Garibay [75]		91.9	91.8	91.8	91.6
Sharma et al. [76]		88.9	91.1	88.67	89.87
Proposed method		**94.89**	**94.25**	**94.95**	**94.60**
Hazarika et al. [77]	**SARC**	78	–	–	77
Ilic et al. [78]		77.3	–	–	–
Mehndiratta and Soni [50]		72.75	–	–	–
Savini and Caragea [79]		–	–	–	76.3
Kumar et al. [80]		–	81.01	78.96	79.97
Savini and Caragea [81]		–	–	–	77.53
Proposed method		**80.59**	**81.34**	**79.66**	**80.49**

The best results are shown in bold

1. *cat with sunglasses is next-level chill* (NS)
2. *obama narrowly misses quarterly performance bonus* (S)
3. *angelina jolie coming for your baby* (S)
4. *Trading by these kind of quotes is a good way to lose money* (NS)
5. *I love you so much Julia, you are really ridiculous* (NS)
6. *And the bills sure were a good team last year* (S)

We took the first three texts from the news headlines repository, and the remaining three are from SARC. Our approach predicts the first headline as sarcastic. This could have been due to the absurdity of the headline, which may have led the model to view it as a humour. This is owing to the absence of context-specific knowledge of the approach. So, discerning amongst humour and sarcasm is essential to achieving more accurate findings for the task. Our method predicts the headline “Hillary Clinton vs. Herself" is sarcastic. This may be due to the lack of contextual information in the headline. Our approach predicts the second headline as non-sarcastic. The reason may be due to the model’s poor contextual handling of Obama’s identity and reputation. A similar mistake happened with the third headline, which was incorrectly classified as non-sarcastic. It could be beneficial to identify essential subjects in texts and acquire context from them so that models can make more precise predictions.

Similar to hashtags on twitter, the tag “/s" is frequently used by the reddit users to convey sarcasm. It can be tough to believe that humans as well can label such comments as sarcastic when this tag is removed so as to learn a more general framework. A similar situation occurs with comment 6, as it does not carry the sarcasm sense when this tag is removed. However, when it comes to comments 4 and 5, our method predicts that they are sarcastic. Both of the comments start with a positive sense and end up with a negative sense, clearly indicating there is sarcasm in them. In some cases, the user who made the comment may have forgotten to add the “/s" at the end to indicate that it was sarcastic.

5 Conclusions and Prospects

Sarcasm identification is a critical challenge in natural language processing. A classification problem in this area aims to distinguish between sarcastic and non-sarcastic utterances. Sentiment analysis and opinion mining studies can be improved by accurately identifying sarcasm. This study concentrated on the context-based feature strategy for sarcasm identification using voting-based ensemble approach. We proposed an ensemble network that employs BERT contextual word embeddings in conjunction with four deep learning models to contribute to this study. Finally, majority voting has been performed on the outputs of all four models. The proposed method achieved state-of-the-art performance on the news headlines dataset (version-2), with an accuracy of 94.89% and an $F1$-score of 80.49% on the SARC dataset. On both datasets, the proposed voting-based ensemble method surpassed all state-of-the-art methodologies with superior results.

In terms of future work, we intend to expand the evaluation with additional sarcasm datasets to assess our frameworks’ performance on different texts. Furthermore, we will investigate the findings of multimodal sarcasm detection by connecting texts with audio to contribute to sarcasm recognition in the progressing world of virtual assistants. The user’s intonation in their speech could be evident as sarcasm. This sort of work is still in its infancy.

References

1. Pang, B.; Lee, L.: Opinion mining and sentiment analysis. Found. Trends® Inf. Retriev. **2**(1–2), 1–135 (2008)
2. Bharti, S.K.; Vachha, B.; Pradhan, R.; Babu, K.S.; Jena, S.K.: Sarcastic sentiment detection in tweets streamed in real time: a big data approach. Digit. Commun. Netw. **2**(3), 108–121 (2016)
3. Parmar, K.; Limbasiya, N.; Dhamecha, M.: Feature based Composite Approach for Sarcasm Detection using MapReduce. In: 2018 Second International Conference on Computing Methodologies and Communication (ICCMC), pp. 587–591. IEEE (2018)
4. Feldman, R.: Techniques and applications for sentiment analysis. Commun. ACM **56**(4), 82–89 (2013)
5. Gautam, G.; Yadav, D.: Sentiment analysis of Twitter data using machine learning approaches and semantic analysis. In: 2014 7th International Conference on Contemporary Computing (IC3), pp. 437–442. IEEE (2014)
6. Saha, S.; Yadav, J.; Ranjan, P.: Proposed approach for sarcasm detection in Twitter. Indian J. Sci. Technol. **10**(25), 1–8 (2017)
7. Attardo, S.: Irony as relevant inappropriateness. J. Pragmat. **32**(6), 793–826 (2000)
8. Gibbs, R.W.: Irony in talk among friends. Metaphor. Symb. **15**(1–2), 5–27 (2000)
9. Tepperman, J.; Traum, D.R.; Narayanan, S.S.: "yeah Right": Sarcasm Recognition for Spoken Dialogue Systems. In: Interspeech (2006)
10. Joshi, A.; Bhattacharyya, P.; Carman, M.J.: Automatic sarcasm detection: a survey. ACM Comput. Surv. **50**(5), 1–22 (2017)
11. Rush, A.M.; Chopra, S.; Weston, J.: A Neural Attention Model for Abstractive Sentence Summarization. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 379–389. Association for Computational Linguistics, Lisbon (2015)
12. Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; Dyer, C.: Neural Architectures for Named Entity Recognition. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 260–270. Association for Computational Linguistics, San Diego (2016)
13. Yu, X.-M.; Feng, W.-Z.; Wang, H.; Chu, Q.; Chen, Q.: An attention mechanism and multi-granularity-based Bi-LSTM model for Chinese Q&A system. Soft. Comput. **24**(8), 5831–5845 (2020)
14. Berrichi, S.; Mazroui, A.: Addressing limited vocabulary and long sentences constraints in English–Arabic neural machine translation. Arab. J. Sci. Eng. **46**(9), 8245–8259 (2021)
15. Tomer, M.; Kumar, M.: Improving text summarization using ensembled approach based on fuzzy with LSTM. Arab. J. Sci. Eng. **45**(12), 10743–10754 (2020)
16. Hermann, K.M.; Kocisky, T.; Grefenstette, E.; Espeholt, L.; Kay, W.; Suleyman, M.; Blunsom, P.: Teaching machines to read and comprehend. Adv. Neural Inf. Process. Syst. **28**, 1 (2015)
17. Rani, M.S.; Subramanian, S.: Attention mechanism with gated recurrent unit using convolutional neural network for aspect level opinion mining. Arab. J. Sci. Eng. **45**(8), 6157–6169 (2020)
18. Van Hee, C.; Lefever, E.; Hoste, V.: Exploring the fine-grained analysis and automatic detection of irony on Twitter. Lang. Resour. Eval. **52**(3), 707–731 (2018)
19. 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 **7**, 23319–23328 (2019)
20. Ptáček, T.; Habernal, I.; Hong, J.: Sarcasm detection on Czech and English Twitter. In: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pp. 213–223 (2014)
21. Sulis, E.; Farías, D.I.H.; Rosso, P.; Patti, V.; Ruffo, G.: Figurative messages and affect in Twitter: differences between# irony,# sarcasm and# not. Knowl.-Based Syst. **108**, 132–143 (2016)
22. Uddin, M.N.; Li, B.; Ali, Z.; Kefalas, P.; Khan, I.; Zada, I.: Software defect prediction employing BiLSTM and BERT-based semantic feature. Soft Comput. **1**, 1–15 (2022)
23. Yuan, L.: A joint method for Chinese word segmentation and part-of-speech labeling based on deep neural network. Soft. Comput. **26**(12), 5607–5616 (2022)
24. Riloff, E.; Qadir, A.; Surve, P.; De Silva, L.; Gilbert, N.; Huang, R.: Sarcasm as contrast between a positive sentiment and negative situation. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 704–714 (2013)
25. Kreuz, R.; Caucci, G.: Lexical influences on the perception of sarcasm. In: Proceedings of the workshop on computational approaches to figurative language, pp. 1–4. Association for Computational Linguistics, Rochester, New York (2007)
26. Maynard, D.G.; Greenwood, M.A.: Who cares about Sarcastic Tweets? Investigating the Impact of Sarcasm on Sentiment Analysis. In: LREC 2014 Proceedings (2014). ELRA
27. Cunningham, H.; Maynard, D.; Bontcheva, K.; Tablan, V.: GATE: an Architecture for Development of Robust HLT applications. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pp. 168–175 (2002)
28. Davidov, D.; Tsur, O.; Rappoport, A.: Semi-Supervised Recognition of Sarcasm in Twitter and Amazon. In: Proceedings of the Fourteenth Conference on Computational Natural Language Learning, pp. 107–116 (2010)
29. Tsur, O.; Davidov, D.; Rappoport, A.: ICWSM—a great catchy name: semi-supervised recognition of sarcastic sentences in online product reviews. In: 4th International AAAI conference on weblogs and social media (2010)
30. González-Ibáñez, R.; Muresan, S.; Wacholder, N.: Identifying sarcasm in Twitter: a closer look. In: Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies, pp. 581–586 (2011)
31. Reyes, A.; Rosso, P.; Buscaldi, D.: From humor recognition to irony detection: the figurative language of social media. Data Knowl. Eng. **74**, 1–12 (2012)
32. Barbieri, F.; Saggion, H.; Ronzano, F.: Modelling sarcasm in Twitter, a novel approach. In: Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, pp. 50–58 (2014)
33. Liu, P.; Chen, W.; Ou, G.; Wang, T.; Yang, D.; Lei, K.: Sarcasm detection in social media based on imbalanced classification. In: International Conference on Web-Age Information Management, pp. 459–471. Springer (2014)
34. Joshi, A.; Sharma, V.; Bhattacharyya, P.: Harnessing context incongruity for sarcasm detection. In: Annual Meeting of the Association for Computational Linguistics (2015)
35. Rajadesingan, A.; Zafarani, R.; Liu, H.: Sarcasm detection on Twitter: a behavioral modeling approach. In: Proceedings of the 8th ACM International Conference on Web Search and Data Mining, pp. 97–106 (2015)
36. Fersini, E.; Pozzi, F.A.; Messina, E.: Detecting irony and sarcasm in microblogs: the role of expressive signals and ensemble classifiers. In: 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. 1–8 (2015). IEEE
37. Hernández-Farías, I.; Benedí, J.-M.; Rosso, P.: Applying Basic Features from Sentiment Analysis for Automatic Irony Detection. In: Iberian Conference on Pattern Recognition and Image Analysis, pp. 337–344 (2015)
38. Bamman, D.; Smith, N.A.: Contextualized Sarcasm Detection on Twitter. In: International Conference on Web and Social Media, vol. 9, pp. 574–577 (2015)

39. Altrabsheh, N.; Cocea, M.; Fallahkhair, S.: Detecting Sarcasm from Students' Feedback in Twitter. In: European Conference on Technology Enhanced Learning, pp. 551–555. Springer (2015)
40. Bouazizi, M.; Ohtsuki, T.O.: A pattern-based approach for sarcasm detection on Twitter. IEEE Access **4**, 5477–5488 (2016)
41. Lukin, S.; Walker, M.: Really? Well. Apparently bootstrapping improves the performance of sarcasm and nastiness classifiers for online dialogue. arXiv preprint arXiv:1708.08572 (2017)
42. Tungthamthiti, P.; Shirai, K.; Mohd, M.: Recognition of sarcasms in Tweets based on concept level sentiment analysis and supervised learning approaches. In: Proceedings of the 28th Pacific Asia Conference on Language, Information and Computing, pp. 404–413 (2014)
43. Ghosh, D.; Guo, W.; Muresan, S.: Sarcastic or not: word embeddings to predict the literal or sarcastic meaning of words. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1003–1012 (2015)
44. Barbieri, F.; Ronzano, F.; Saggion, H.: UPF-taln: SemEval 2015 Tasks 10 and 11. Sentiment Analysis of Literal and Figurative Language in Twitter. In: Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), pp. 704–708 (2015)
45. Tungthamthiti, P.; Shirai, K.; Mohd, M.: Recognition of Sarcasm in Microblogging Based on Sentiment Analysis and Coherence Identification. J. Nat. Lang. Process. **23**, 383–405 (2016)
46. Reyes, A.; Rosso, P.: Making objective decisions from subjective data: detecting irony in customer reviews. Decis. Support Syst. **53**(4), 754–760 (2012)
47. Sarsam, S.M.; Al-Samarraie, H.; Alzahrani, A.I.; Wright, B.: Sarcasm detection using machine learning algorithms in Twitter: a systematic review. Int. J. Mark. Res. **62**(5), 578–598 (2020)
48. Suykens, J.A.; Vandewalle, J.: Least squares support vector machine classifiers. Neural Process. Lett. **9**(3), 293–300 (1999)
49. Felbo, B.; Mislove, A.; Søgaard, A.; Rahwan, I.; Lehmann, S.: Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. arXiv preprint arXiv:1708.00524 (2017)
50. Mehndiratta, P.; Soni, D.: Identification of sarcasm using word embeddings and hyperparameters tuning. J. Discrete Math. Sci. Cryptogr. **22**(4), 465–489 (2019)
51. Shrivastava, M.; Kumar, S.: A pragmatic and intelligent model for sarcasm detection in social media text. Technol. Soc. **64**, 101489 (2021)
52. Gregory, H.; Li, S.; Mohammadi, P.; Tarn, N.; Draelos, R.; Rudin, C.: A Transformer Approach to Contextual Sarcasm Detection in Twitter. In: Proceedings of the Second Workshop on Figurative Language Processing, pp. 270–275 (2020)
53. Reyes, A.; Rosso, P.; Veale, T.: A multidimensional approach for detecting irony in Twitter. Lang. Resour. Eval. **47**(1), 239–268 (2013)
54. Ghosh, A.; Li, G.; Veale, T.; Rosso, P.; Shutova, E.; Barnden, J.; Reyes, A.: SemEval-2015 Task 11: Sentiment Analysis of Figurative Language in Twitter. In: Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), pp. 470–478 (2015)
55. Filatova, E.: Irony and Sarcasm: Corpus Generation and Analysis Using Crowdsourcing. In: LREC, pp. 392–398. Citeseer (2012)
56. Wallace, B.C.; Choe, D.K.; Charniak, E.: Sparse, contextually informed models for irony detection: exploiting user communities, entities and sentiment. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on NLP, pp. 1035–1044 (2015)
57. Khodak, M.; Saunshi, N.; Vodrahalli, K.: A large self-annotated corpus for sarcasm. arXiv preprint arXiv:1704.05579 (2017)
58. Joshi, A.; Tripathi, V.; Bhattacharyya, P.; Carman, M.J.: Harnessing sequence labeling for sarcasm detection in dialogue from TV series 'Friends'. In: CoNLL, pp. 146–155 (2016)
59. Misra, R.; Arora, P.: Sarcasm detection using hybrid neural network. arXiv preprint arXiv:1908.07414 (2019)
60. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
61. Pennington, J.; Socher, R.; Manning, C.D.: GloVe: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
62. Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T.: Enriching Word Vectors with Subword Information. Trans. Assoc. Comput. Ling. **5**, 135–146 (2017)
63. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805 (2018)
64. Liu, L.; Wang, M.; Zhang, M.; Qing, L.; He, X.: UAMNer: uncertainty-aware multimodal named entity recognition in social media posts. Appl. Intell. **52**(4), 4109–4125 (2022)
65. Ahuja, R.; Sharma, S.: Transformer-based word embedding with CNN model to detect sarcasm and irony. Arab. J. Sci. Eng. **1**, 1–14 (2021)
66. Alzubi, J.A.; Jain, R.; Singh, A.; Parwekar, P.; Gupta, M.: COBERT: COVID-19 question answering system using BERT. Arab. J. Sci. Eng. **1**, 1–11 (2021)
67. Yang, J.; Wang, M.; Zhou, H.; Zhao, C.; Zhang, W.; Yu, Y.; Li, L.: Towards Making the Most of BERT in Neural Machine Translation. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 9378–9385 (2020)
68. Liu, Y.; Lapata, M.: Text Summarization with Pretrained Encoders. arXiv preprint arXiv:1908.08345 (2019)
69. Niu, X.-X.; Suen, C.Y.: A novel hybrid CNN-SVM classifier for recognizing handwritten digits. Pattern Recogn. **45**(4), 1318–1325 (2012)
70. Zhou, C.; Sun, C.; Liu, Z.; Lau, F.: A C-LSTM neural network for text classification. arXiv preprint arXiv:1511.08630 (2015)
71. Ertam, F.; Aydın, G.: Data classification with deep learning using Tensorflow. In: 2017 International Conference on Computer Science and Engineering (UBMK), pp. 755–758 (2017). IEEE
72. Shrikhande, P.; Setty, V.; Sahani, A.: Sarcasm Detection in Newspaper Headlines. In: 2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS), pp. 483–487 (2020). IEEE
73. Pandey, R.; Kumar, A.; Singh, J.P.; Tripathi, S.: Hybrid attention-based Long Short-Term Memory network for sarcasm identification. Appl. Soft Comput. **106**, 107348 (2021)
74. Jamil, R.; Ashraf, I.; Rustam, F.; Saad, E.; Mehmood, A.; Choi, G.S.: Detecting sarcasm in multi-domain datasets using convolutional neural networks and long short term memory network model. PeerJ. Comput. Sci. **7**, 645 (2021)
75. Akula, R.; Garibay, I.: Interpretable multi-head self-attention architecture for sarcasm detection in social media. Entropy **23**(4), 394 (2021)
76. Sharma, D.K.; Singh, B.; Garg, A.: An ensemble model for detecting sarcasm on social media. In: 2022 9th International Conference on Computing for Sustainable Global Development (INDIACom), pp. 743–748. IEEE (2022)
77. Hazarika, D.; Poria, S.; Gorantla, S.; Cambria, E.; Zimmermann, R.; Mihalcea, R.: CASCADE: contextual sarcasm detection in online discussion forums. In: Proceedings of the 27th International Conference on Computational Linguistics, pp. 1837–1848 (2018)
78. Ilić, S.; Marrese-Taylor, E.; Balazs, J.; Matsuo, Y.: Deep contextualized word representations for detecting sarcasm and irony. In: Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, pp. 2–7 (2018)

79. Savini, E.; Caragea, C.: A Multi-task learning approach to sarcasm detection (student abstract). In: AAAI Conference on Artificial Intelligence (2020)
80. Kumar, A.; Narapareddy, V.T.; Gupta, P.; Srikanth, V.A.; Neti, L.B.M.; Malapati, A.: Adversarial and auxiliary features-aware BERT for sarcasm detection. In: Proceedings of the 3rd ACM India joint international conference on data science & management of data (8th ACM IKDD CODS & 26th COMAD), pp. 163–170 (2021)
81. Savini, E.; Caragea, C.: Intermediate-task transfer learning with BERT for sarcasm detection. Mathematics **10**(5), 844 (2022)