International Journal of Information Science and Management
Vol. 20, No. 4, 2022, 1-15
https://dorl.net/dor/20.1001.1.20088302.2022.20.4.1.0
Original Research

Sarcasm Detection with and without #Sarcasm: Data Science Approach

Rupali Amit Bagate
Department of Computer Science & Engineering,
Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai, Tamilnadu, India
Corresponding Author: rupali.bagate@gmail.com
ORCID iD: https://orcid.org/0000-0002-7538-5414

R. Suguna
Department of Computer Science & Engineering,
Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai, Tamilnadu, India
drsuguna@veltech.edu.in
ORCID iD: https://orcid.org/0000-0002-8930-9092

Received: 06 December 2021
Accepted: 05 May 2022

Abstract

Natural languages usually contain context, which is difficult for a machine to understand. Sentiment analysis is a contextual mining technique often used in NLP to identify, understand and extract subjective information in texts, such as people's comments, feedback, reviews, and opinions. Sentiment analysis is a useful tool for finding the polarity of a sentence. Sarcasm detection is one of the complex areas of sentiment analysis. Sarcasm flips the polarity of the sentence identified by sentiment analysis. Thus, sentiment analysis results may get biased if people use sarcasm in their text. Hence, to understand the sentence's real meaning, we proposed a system of sarcasm detection on tweets using an ensemble approach. We performed sarcasm detection with and without #sarcasm. After training a model and observing earlier studies, We found that the presence of #sarcasm gives a better result. Therefore the author tried implementing a model where #sarcasm is removed from the tweets, and the model is trained. After comparing both models' presence and absence of hashtags, it is found that the lack of the hashtag model works well, which can be used on any plain text without any clue of sarcasm.

Keywords: Sentiment Analysis**,** Sarcasm Detection, Hashtag, Machine Learning, Deep Learning, Twitter, #sarcasm.

Introduction

Natural language processing is a cross-disciplinary and broad-ranging field of computer science and linguistics whose goal is to analyze and understand human language, namely speech, and text, using machines. Sentiment classification in NLP is a complex problem due to the context present in texts which is difficult for machines to understand. Due to this, it has a huge commercial value for enterprises such as E-commerce sites like Amazon, Flipkart etc., attempting to understand user behavior by analyzing feedback, reviews, comments, etc. But sometimes fail in doing so due to sarcasm in the text. Sarcasm can flip the polarities of the text. Hence, the classification results might be incorrect, resulting in biased insights into the enterprises. Sarcasm on its own is challenging to identify for humans, let alone machines (Bagate & Suguna, 2019). Humans use sarcasm in their daily conversations while arguing, criticizing, or engaging in humor. People use tonal stress and different facial and hand gestures

to depict sarcasm, but machines cannot understand these while analyzing texts.

Previous works (Tarigan & Girsang, 2018) use word similarity scores as an augmented feature to determine a sentence's polarity, identifying and boosting the sarcasm detection process. The authors have used word similarity as a feature and deep learning, producing up to 85.65% accuracy in sarcasm detection. Online textual data such as reviews, feedback, and comments often contain emotions such as anger, sadness, excitement, happiness, boredom etc. which further make the text classification difficult but contribute hugely to the results. Hence, only word scores are not enough to determine the polarity; exploring these features while detecting sarcasm is essential.

Therefore author proposed a model incorporated with support vector machines (SVM), logistic regression, extended short-term memory networks (LSTM), and various other tree-based models previously used to detect sarcasm. Research work mainly focuses on identifying tweets as sarcastic or non-sarcastic from the Twitter dataset, where maximum tweets are present with clues such as #sarcasm, #nothappy, #sad etc. In this work, the authors monitor how the presence and absence of "#sarcasm" affect the accuracy of sarcasm detection. We aim to improve the performance of previous models using ensemble learning approaches such as bagging, boosting, and stacking. We performed a comparative analysis of sarcasm detection with the presence and absence of #sarcasm in tweets. After verifying the results, we found our model performs far better than existing models.

As described earlier in the introduction segment, many researchers have tried finding sarcasm in text with clues in tweets. Our novelty of work is that we implemented our system to find sarcasm in the text without a clue, such as #sarcasm present in tweets. The presence of #sarcasm in tweet make the machine learning model easy to learn and predict sentence as sarcastic or non-sarcastic. The other section shows how systems are trained on a dataset where we removed #sarcasm from tweets and inputted this text to different models as input for training, and still, systems are predicting better than present techniques.

This paper is divided into different parts. Section 2 describes a literature survey that supports how our work is better than the existing system. Section 3 discusses the proposed methodology, describing the implemented architecture and how the model works on the tweets dataset. Section 4 describes different methodology implementations and results performed on the Twitter dataset. Section 5 talks about the conclusion, followed by the future scope.

Literature Review

The authors have studied various articles and research work in a similar area. The research by Ghosh, Fabbri & Muresan (2017) has inspired this Research Work. Their work explores various LSTM models and the conversational context in sarcasm detection. Previously, much work has been done using self-annotated Twitter data and Reddit corpus in sarcasm detection (Khodak, Saunshi & Vodrahalli, 2018). Many initial studies focused on using traditional regression-based approaches for sarcasm classification, using only the comment's text to be assessed. Guo and Shah (n.d.) and Gonzalez-Ib´anezet, Muresan and Wacholder (2011) used SVM and logistic regression over a feature set of unigrams, pragmatic features such as signs and symbols, e.g., emoticons and dictionary-based lexical features. The result of the classifier was then compared with that of humans to gain an insight into the performance of the classifier. Reyes and Rosso (2012) used short texts on Twitter to recognize irony at the linguistic level. They used a set of textual features in their work and constructed a model assessed along two

dimensions: relevance and representativeness. Riloff, Qadir, Surve, De Silva, LGilbert & Huang (2013) studied the negative situations or phrases where positive sentiments were used in close proximity and then used them as a feature for detecting sarcasm. Buschmeier, Cimiano & Klinger (2014) used different classifiers on the Amazon review dataset and compared the results of these classifiers using the difference between the sentiment expressed by their view and the user-given star rating. The survey performed by RupaliBagate, Saini, Sethi, Tomar and Singh (2021) also supports the methodology used in the proposed work far better than other classification models. One more challenge they found is to explain the ability of sarcasm detection by the model is also important. Handoyo and Suhartono (2021) studied and implemented model on various datasets referred by multiple authors for sarcasm detection. They have used a robustly optimized BERT approach along with glove on a big dataset. The authors have worked on the dataset referred by Ptácek, Habernal and Hong (2014), Ghosh and Veale (2016), Oprea and Magdy (2020), Van Hee, Lefever & Hoste (2018). Out of which, two are automatically collected, and the remaining are manually annotated, which results in a big dataset for BERT. They considered context along with the sentiment of the tweeting user. Venkatesh & Vishwas (2021) used an ensemble approach for sarcasm detection. The authors have tried various machine learning models for sarcasm detection. Using the ensemble approach achieved an f-score of 97%. Our proposed model is performing better than their models. Razali, Halin, Ye, Doraisamy & Norowi (2021) have extracted features from CNN and handcrafted features set such as incongruity, hyperbole, temporality, and dislike detection. The author used different classification algorithms SVM, KNN, LR, DT, and DISCR. comparedd to all classification algorithms Logistic regression produces better accuracy of 94%.

As we surveyed, many researchers have referred to the deep learning approach with many classification algorithms. All have referred to #sarcasm as the main feature for sarcasm detection. But in our proposed methodology, we dropped #sarcasm features while predicting the tweet classification, which makes our proposed system more nonvulnerable to any complicated text.

Proposed Methodology

The following research aims to detect sarcastic language or content from social media platforms, specifically Twitter, where people identify irrelevant and sarcastic views of people. These opinions can be used as reviews for effective decisions. To achieve high classification accuracy, the author used machine learning algorithms and multiple CNN models, which are ensemble together, which results in a very decent F-Score. Using n-Fold cross-validation, the validity of the model is enhanced. Figure 1 describes the proposed architecture of the research work; as shown in the diagram author divided the tweeter dataset into training and testing datasets. Initially extracted tweets need to be cleaned using different data pre-processing techniques. Techniques are described in the following subsection. After pre-processing data, it needs to be vectorized to feed other machine learning models. Many available vectorization techniques include word2vec, tfidf, glove, etc. Our model has chosen glove as the embedding technique because it produces a better result than other techniques. These processed vectors are fed to ensemble classifiers. How these classifiers work is explained in the subsection. Further, ensembling is used to improve the result of different classifiers.

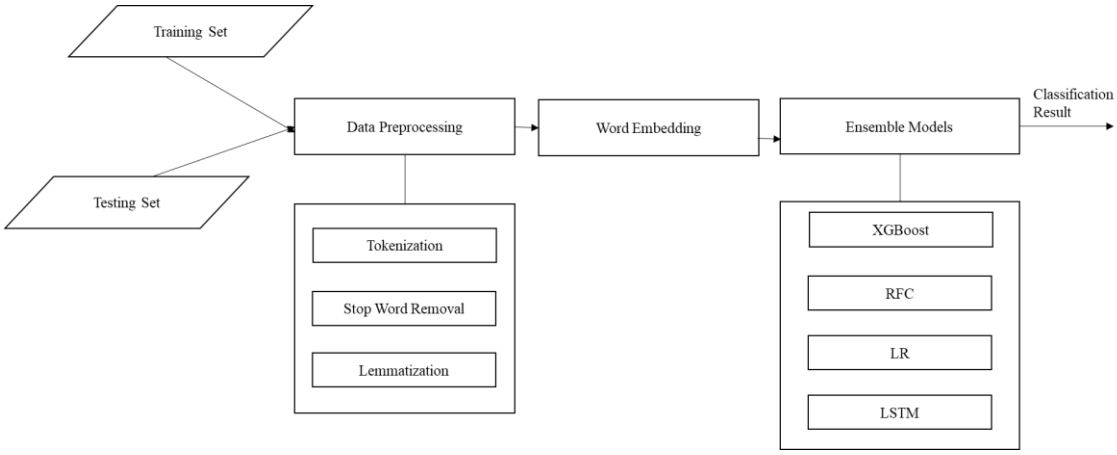

Figure 1: Proposed System Architecture

Dataset Pre-processing

The gathered data has to be cleaned and processed before it can be made available for the classification phase. We used different techniques for pre-processing, tokenization, stop word removal, and stemming, as described below.

Tokenization

Tokenization is performed on tweets to break them down into meaningful modules. The text is divided into several parts known as tokens which may contain words or other elements (Liu, 2012). Example- A sentence "we are students of #veltech" can be tokenized as "we", "are", "students", "of", "#","veltech".

Stop Words Elimination

Stop words are generally common words like pronouns such as 'he', and'she' or conjunctions such as 'and', 'or', 'but' etc. Such words either do not have much effect or add little or no value to the categorization process. NLTK library provides a list of stop words or commonly repeated features, and a feature is removed whenever it matches any feature in the list.

Stemming

Stemming is the process of reducing words to their stem or roots. The basic idea behind stemming is grouping the words with common or close meanings together and trying to enhance the efficiency of Natural Language Processing. For example, the words depressing, depressed, depress, and depression have different suffixes such as 'ing', 'ed', 'ion', but they all have the same stem or root i.e. 'depress'. We have used Snowball stemmer for suffix stripping.

Word Embedding

Word embedding is a critical technique to represent a document's vocabulary. Word embedding identifies the context of words from a given document along with its syntactic and semantic relation with present words in documents. Word vector is vector representations of a given word from a document. Aditya, Joshi, Tripathi, Patel, Bhattacharyya & Carman (2016) have experimented with four-word embedding techniques LSA, Glove, Dependency weight, and Word2Vec. After observing the results, Glove has performed better than other techniques.

So we decided to go with Glove, which finds and relates vector differences of two contrast words with closely possible meanings. Therefore in the proposed research pre-trained glove model is used for vectorization.

Classifiers

Authors have combined two approaches, machine learning, and Deep learning, for research work. Each methodology is explained in detail below section.

Machine Learning Approaches

Machine learning is a part of Artificial Intelligence in which a machine learns from its past data and performs an action without human assistance. Machine Learning is broadly classified as Supervised and Unsupervised learning. Few used approaches are explained further how it contributes to our work.

Naïve Bayes

Naive Bayes is from the family of Bayes techniques in which it is assumed that each attribute/feature is independent of the others. This is a probabilistic classifier; hence the probability is calculated for each category using Bayes theorem, and the category with the highest probability will be the output.

$$P(c \mid d) = [\, P(d \mid c) \times P(c) \,] / [\, P(d) \,] \quad (1)$$

XGBoost

XGBoost is an optimized gradient boosting library which is highly efficient, portable, and flexible. It implements machine learning algorithms under the Gradient Boosting framework. GBDT and GBM provide better results than many algorithms to solve machine learning problems accurately and faster.

Logistic Regression

It is a classification algorithm for supervised learning used to predict the probability of target variables. There are two classes, 0 and 1, used for binary classification.

Deep Learning Approaches

Convolutional Neural Network

Convolutions Neural Networks have several layers that take input data and weights, respectively, then send them for further processing. It consists of various layers, including convolution layers, Pooling Layers and Fully connected ones. Also, different activation layers are applied to the layers, including ReLU, signmoid, and tanh. Activation functions are mathematical functions that produce neural network output. The mathematical function determines whether this function should be fired or not. This is based on whether each neuron's input is relevant for the prediction of the model. Activation functions also help output to be normalized in different ranges depending on the activation function, which include ranges from -1 to 1 or 0 to 1. Figure 2 shows the basic Architecture of CNN.

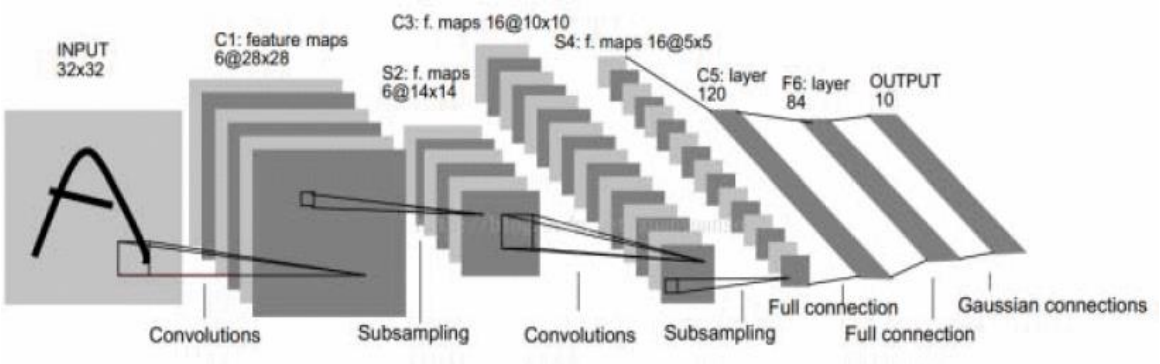

Figure 2: Basic Architecture of CNN (LeCun, Bottou, Bengio & Haffner, 1998)

Recurrent Neural Network

A Recurrent Neural Network is a Neural Network in which the previous step forwards output to the next step. The Hidden state is the most precious state of RNN. RNN appeared when previous words were used to predict the next words. RNN is very useful in natural language processing, especially in sarcasm detection.

Long Short Term Memory Networks

LSTM is a refined module of RNN with a chain-like structure with a four-layer structure interacting in an extraordinary way instead of a single tanh layer. Figure 3 shows a simple LSTM structure

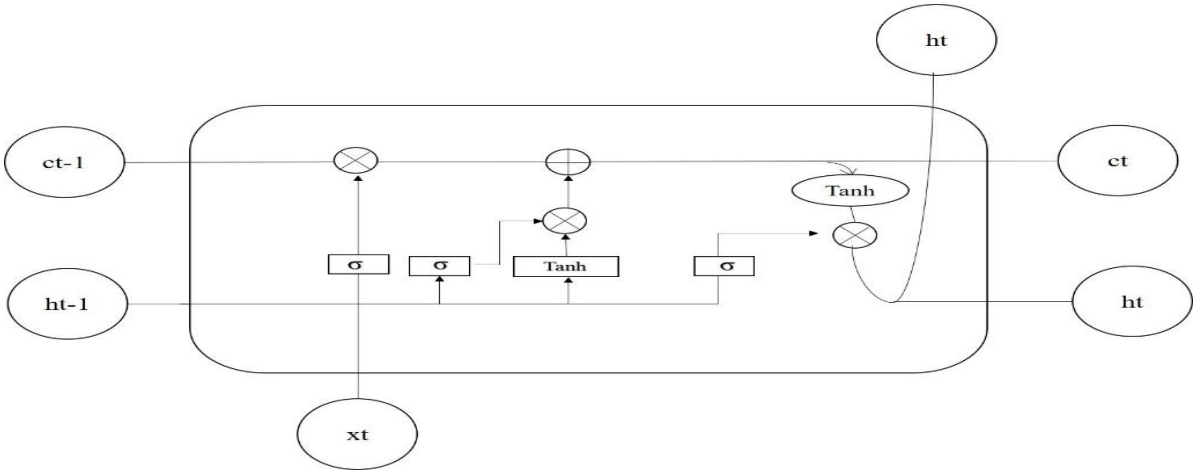

Figure 3: LSTM Structure

Gated Recurrent Unit

Gated Recurrent Unit tends to remember long-term dependencies. They are a particular type of RNN. Figure 4 shows the Architecture of GRU used in research.

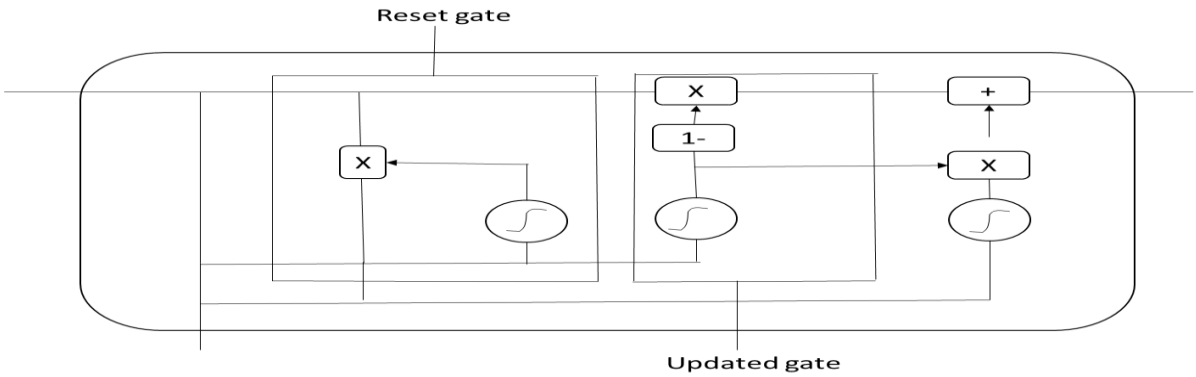

Figure 4: General Architecture of GRU

The update gate decides which information to keep and which to remove. It is similar to forget gate of LSTM. The Reset gate helps to forget past information.

Ensemble Methods

Soujanya Poria, Cambria, Hazarika &Vij (2016) ensemble four models, including Emotion, Personality, Sentiment, and Text. The F-Score was predicted to be 90.7% when all four fully connected features were combined. Figure 5 shows the Ensemble Model of various CNN models.

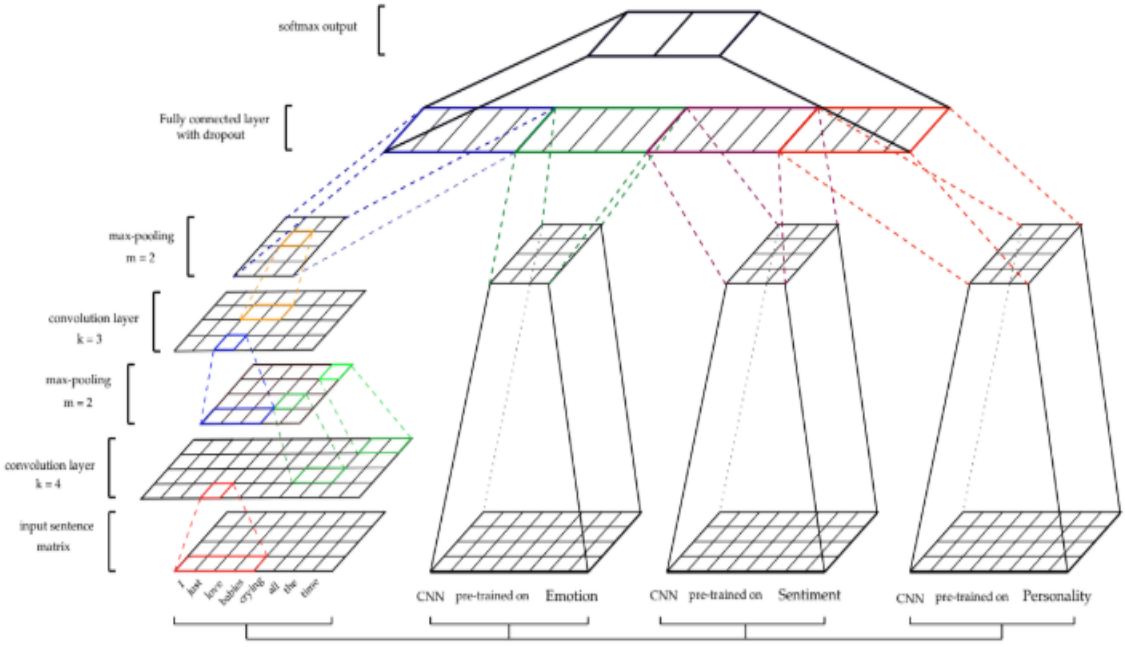

Figure 5: Ensemble method using various CNN models (Poria et al., 2016)

As mentioned in the above section, we have described the different classification models used in our research. Many authors used all these models on a dataset with #sarcasm. But our proposed methodology is different as it is trained and tested without #sarcasm. Without clues present also proposed system is generating outstanding accuracy.

The following section describes how different classification models and techniques are ensemble and trained to achieve accurate classification.

Results

Figure 6 shows a black box of our working model where authors feed tweets with #sarcasm and without #sarcasm as input to the proposed model. The model shown in the diagram ahead provides the result as zero or one. If the model's output is one, it is a sarcastic sentence; Otherwise, a non-sarcastic sentence.

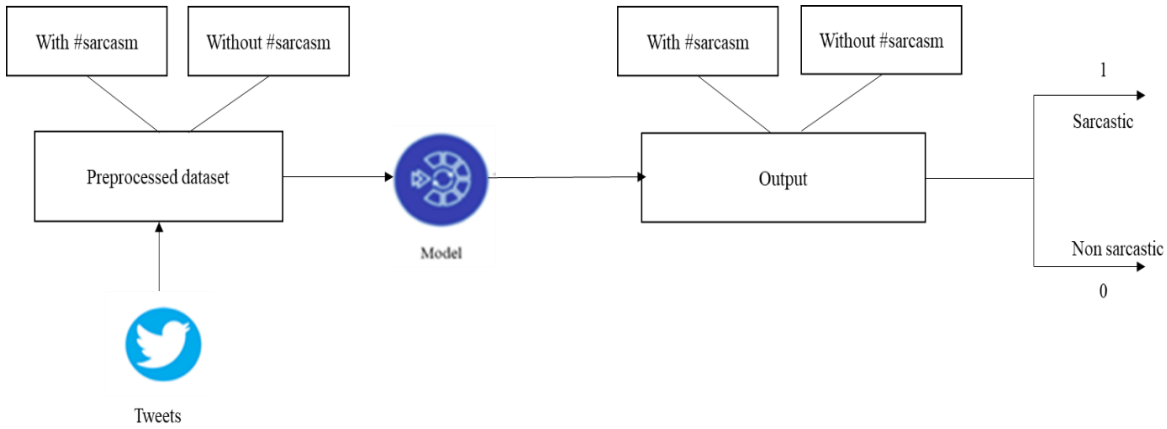

Figure 6: The Black box of the working model

The current section describes the working model. We have trained and tested model with a Random forest classifier, logistic regression, XgBoost classifier, and Neural Network. We used the Twitter dataset for training and testing (Riloff et al., 2013). The contribution factor of our research is author trained model for a given dataset with #sarcasm and without #sarcasm. Regarding a survey or earlier state of the art, researchers have worked on tweets with #sarcasm. Implementing a model with #sarcasm present in the tweet makes model prediction very easy and improves the model's accuracy.

Therefore, we removed the #sarcasm from the tweet before training and testing our model. This makes out model more robust even if a tweet contains no sarcastic hashtag or hint. Without the #sarcasm process, we remove the hashtag from the dataset and then feed it to the model while training. This makes the system more intelligent in identifying the tweets in the testing phase. It's prevalent and easy for a system to identify sentences or tweets as sarcastic or not with hints like #sarcasm, #not liked, Sarcastic. E.g., "wow, what a food #sarcasm". If the machine inputted a sentence with a hashtag while testing the model's prediction will be 1(sarcastic). It's self-evident for a model to predict tweets as sarcastic with #sarcasm present.

Therefore, to make the model more realistic and challenging, we implement the model without hashtags present in a tweet while training and testing the model. A piece of code is written in python after the dataset preprocessing to remove #sarcasm from a tweet and then fed tweets data frame to different models for training and testing purposes.

We used the Sarcasm detection model in the python environment. Python uses different packages and libraries such as Keras, nltk, pandas, numpy, and sklearn. As described earlier authors have used various machine learning and neural network models. After the data preprocessing PCA model is applied for dimension reduction. We have trained the dataset using LSTM neural network and different machine learning models. Figure 7 shows how NN uses LSTM classification model works on a dataset. Every layer described below works and contributes a significant amount of data toward sarcasm detection.

- **Embedding Matrix:** The embedding matrix is used to solve the relationship representation problem. In this model we have used an embedding matrix to quantify the relationship among words in terms of colloquial linguistic norms.
- **Sequential Model:** In this work, we used the Sequential model API from Keras. The Sequential model is a model depicting a stack of layer where each layer has exactly one input tensor and one output tensor. Its layer doesn't share any input or output and that particular feature helps us create model with much more flexibility as an interaction between layers is only limited to the next and previous layers.
- **Embedding Layer:** Keras offer an embedding layer that can be used for neural networks on text data. Tokenizer API from Keras is used to prepare data. A unique integer represents each word in this layer. A pre-computed embedding matrix as weights and 3D tensor was used in this particular case.

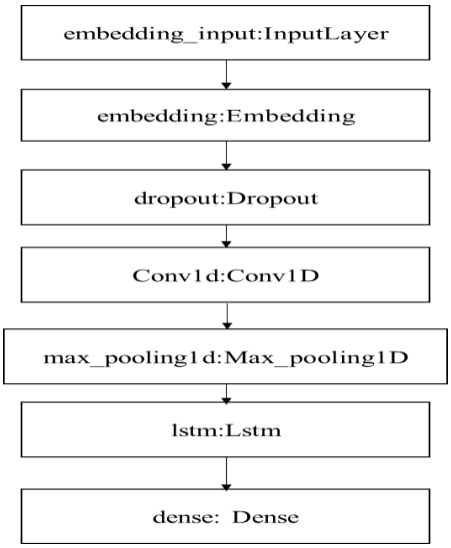

Figure 7: NN LSTM Model Architecture

- **Dropout Layer:** Neural networks are very likely to overfit a training dataset. Dropout creates an illusion of ensembles of neural networks with different model configurations.

A dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, inputs not set to 0 are scaled up by 1/(1- rate) such that the sum over all the inputs is unchanged.

- **Convo1D Layer:** Convolutional is a mathematical operation in which you suffice a tensor, matrix or vector into smaller ones. A convolution operates on the first axis, with one-dimensional space Conv1D. We have used ReLU function, a piecewise linear function that will output the input directly if it's positive; otherwise, it will output zero.
- **MaxPooling1D Layer:** MaxPooling1D is something that reduces the dimensionality of data in the output from the previous convolutional layer. When a filter convolves a given input, it gives us an output. In Max pooling, we down sample the input representation by taking the maximum value over the window defined by pool size.
- **LSTM Layer:** Long Short-Term Memory Networks are a special kind of RNN, capable of learning long-term dependencies, and as in our research dealing with prior context was one of the prime predicaments we need to overcome.

- **Dense Layer:** Dense Layer act as a normal layer in Deep Learning neural network. In this particular layer we have a dot product between input and kernel (weights). To optimize the output, bias is added along with a relatively specific function to achieve a conducive output. In this model, sigmoid function is used to ensure that the output we receive is a linear output and allow us the flexibility to use complex neural structure.

After training the NN model, we introduced various parallel machine learning models. We have trained Logistic regression, Random forest, and Xgboost model on similar tweet datasets. Once all models were trained, we tested our data set, having tweets with and without hashtags, and generated results. Table 1 and Table 2 show the results of working models with different inputs. Table 1 describes the result of sarcasm detection with hashtags, and table 2 shows the result without hashtags. And as discussed earlier, many authors have worked on it with #sarcasm, but no one has worked on it without #sarcasm. But after comparing the result of our model with #sarcasm present without #sarcasm, the absence of hashtag also gives outstanding accuracy.

Table 1 shows the result of sarcasm detection when tweets are inputted along with #sarcasm. XGBoost classier gives an accuracy of 99.63 % and an F1 score of 99.64 %, which is a pretty good score given by the model.

Table 1
Results of classifiers with #sarcasm

Model	F1 Score	Auc Score	Accuracy %
Neural Net with LSTM	0.931056	0.935183	0.934558
Random Forest Classifier	0.991951	0.992469	0.992284
Logistic Regression	0.916155	0.919773	0.913362
XGBoost Classifier	0.996345	0.996516	0.996484

But after the removal of #sarcasm from tweets, the XGBoost classifier still gives comparatively good results accuracy of 99.56 % and an F1 score of 99.58 %. Therefore, the summary in Table 2 shows that even after removing hashtags from tweets, the system is still performing well. Therefore system trained without hashtags also predicts tweets sarcastic with reasonable accuracy.

Table 2
Results of classifiers without #sarcasm

Model	F1 Score	Auc Score	Accuracy %
Neural Net with LSTM	0.925474	0.929346	0.928990
Random Forest Classifier	0.992052	0.992572	0.992381
Logistic Regression	0.912716	0.916867	0.909357
XGBoost Classifier	**0.995635**	0.995828	0.995800

To improve the classification results, we inculcate the stacking ensemble learning, which helps to combine different models and classifiers via meta-regressor. The base model Logistic Regression and NN are trained on the entire training dataset. Further, it's applied to the XGB Metamodel to improve the classification result. As a result of stacking model with #sarcasm gives a result of 99.74 % accuracy, and without #sarcasm gives an accuracy of 99.58 %.

As shown in Tables 1 and 2 results generated are far better. We have also compared the

proposed model result and earlier results to show accuracy improvement. The comparative analysis demonstrated in Tables 3 and 4 proves the proposed model is generating better results than existing models. As shown in table 3, Liebrecht, Kunneman & van Den Bosch (2013) have referred to the Dutch language for sarcasm detection. The comparison is in Table 3, as very few researchers have worked on datasets where a clue of #sarcasm is absent. Tables 3 and 4 show that our model prediction is far better than compared models. The main contribution of the proposed work is the ensemble approach with the combination of without #sarcasm.

Table 3

Comparative analysis of Sarcasm detection without #sarcasm (RupaliBagate & Suguna, 2021)

Authors	Dataset	Methodology	Language	Accuracy
Liebrecht et al. (2013)	Tweets	Balanced Winnow	Dutch	75%
González-Ibánez, Muresan, & Wacholde (2011)	Tweets	SMO (sequential minimal optimization) with LIWC+ _P(Linguistic Inquiry and Word Count) and LIWC+ _F	English	75.89%
Our proposed Model	Tweets	Ensemble approach	English	99.58%

Table 4

Comparative analysis of Sarcasm detection with #sarcasm

Authors	Dataset	Methodology	Language	Accuracy
Jamil, Ashraf, Rustam, Saad, Mehmood, & Choi (2021)	Multi Domain Dataset	Hybrid approach (CNN+ LSTM)	English	91.60
Sonawane & Kolhe (2020)	Tweets	Term Co-occurrence Based Sarcasm Detection	English	93.54
Our proposed Model	Tweets	Ensemble approach	English	99.64%

Discussion

The goal of this research work is to identify sarcasm from natural language written in the English language. This research paper discusses various methodologies ensemble together for better performance. Here, the researcher compared sarcasm detection on tweets using #sarcasm and without #sarcasm. From the majority of findings and literature surveys, maximum researchers have detected sarcasm with the clue of #sarcasm in tweets. Ghosh et al. (2017) referred to online interactions for sarcasm detection, whereas our proposed methodology works on tweets. If we compare both contexts, it relates to natural language processing. Finding sarcasm in NLP applies the same process, such as pre-processing the text, splitting filtered datasets for training and testing purposes, and inputting to different models.

Further, this model works on different features to identify sarcasm. In our proposed model, we did sarcasm detection with the clue of #sarcasm and without #sarcasm. Whereas in Ghosh et al. (2017) referred only to "the clue of #sarcasm with attention mechanism" to identify the weight

of different features contributing to sarcasm detection. They have generated the f1score with attention mechanism as 73.7 %, whereas our proposed model is generating better accuracy as 0.996 with #sarcasm and 0.995 without #sarcasm.

In sarcasm detection, different types of conversation datasets are referred. Such as the news headlines dataset, online conversation on the tweet, and Reddit dataset. Guo and Shah (n.d.) have used the Reddit dataset for research purposes. The author has used an unbalanced dataset for his work. Unbalanced dataset stands for the balance of sarcastic and non-sarcastic comments are uneven. The percentage of non-sarcastic comments is more than that of sarcastic comments. Whereas in our research, we used a tweet balance dataset. This makes the system more efficient and fault-tolerant. Guo and Shah (n.d.) made a system with more precession and less recall, with f1score of 0.098. Whereas our proposed system has both precision and recall with good values with accuracy as 0.996 with #sarcasm and 0.995 without #sarcasm.

Different metrics are used to measure the system's performance, such as precision, recall, F1Score, AUC score, and confusion matrix. Bagate and Suguna (2019) have mentioned different metric parameters, such as.

- Recall that it stands for the total number of positive samples classified as positive from the total positive samples.

- Precision stands for the total number of positive samples retrieved from total positive samples.

- F1Score is the harmonic mean of Recall and precision. If recall and precision are closer to one, f1 score is better.

- Accuracy is the total number of correct predictions out of the total predictions present.

- Auc Score is the area under the ROC curve. This curve helps in visualizing the true positive rate and true negative rate.

- Confusion Matrix is the representation of how model prediction has performed. It has four parameters contributing to the model accuracy and other metric parameters such as TN, FN, TP, and FP. More TP and TN better the result.

As mentioned in our research, we considered Accuracy, F1score, and Auc score as measuring parameters. Nowadays, pointing out sarcasm in conversion impacts the mentality and thinking of people a lot. This research has focused on this important topic and tried to identify the sarcasm from tweets with and without a clue present in text. We choose a tweet as a dataset to be part of the research because it is one of the popular platforms for youth to express their opinion. This study further proves combination of machine learning and deep learning models along with a feature set generates outstanding results in terms of f1score and accuracy. As well as this study proves model classifying tweets in the correct class with and without the presence of specific sarcastic clues in tweets.

Conclusion

The research paper proposed an implementation of sarcasm detection with and without the #sarcasm keyword present in tweets. Detecting sarcasm accuracy has excellent potential to improve our understanding of text processing. Various works that authors have studied have shown that there is still much research in this field. Deep learning approaches proved more efficient than their counterparts in classifying sarcasm in tweets. We experimented with models like xgboost, LR, RF, and NN. Using #sarcasm as a feature gave excellent results with high accuracy. Even without #sarcasm, the accuracy gets improved.

Further ensembling methods, such as stacking, are used to improve both models' accuracy. The classification model gives 99.58 %. Accuracy without #sarcasm comparatively with #sarcasm present is a good outcome. Thus a proposed system with and without hashtags provides good accuracy, and the still absence of hashtags in the text model is a better classification of tweets.

Future Scope

In the Natural language processing field, identifying sarcastic intent is a challenging job. More improvements in sarcasm classification can be made by using recent new methods, such as the extreme gradient boosting model. Many researchers have worked on the Twitter data set, but few have worked on domain-specific tweets. Therefore the future scope of this work, we can apply this model to domain-specific datasets to further make it more realistic and applicable to the real world.

References

Bagate, R. A. & Suguna, R. (2019, September). Different Approaches in Sarcasm Detection: A Survey. In *International Conference on Intelligent Data Communication Technologies and Internet of Things* (pp. 425-433). Springer, Cham.

Buschmeier, K., Cimiano, P. & Klinger, R. (2014, June). An impact analysis of features in a classification approach to irony detection in product reviews. In *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis* (pp. 42-49). Baltimore, Maryland. Association for Computational Linguistics. Retrieved from https://aclanthology.org/W14-2608.pdf

Ghosh, A. & Veale, T. (2016). Fracking sarcasm using neural network. In *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, (pp. 161–169). San Diego, California. Association for Computational Linguistics. Retrievd from https://aclanthology.org/W16-0425.pdf

Ghosh, D., Fabbri, A. R. & Muresan, S. (2017, July). The role of conversation context for sarcasm detection in online interactions. In *Proceedings of the SIGDIAL 2017 Conference*, (pp.186 -196), Saarbr¨ucken, Germany. Retrieved from https://arxiv.org/pdf/1707.06226v1.pdf

González-Ibánez, R., Muresan, S. & Wacholder, N. (2011, June). 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). Portland, Oregon, USA. Association for Computational Linguistics. Reyrieved from https://aclanthology.org/P11-2102.pdf

Guo, N. & Shah, R. (n.d.). Finding sarcasm in Reddit postings: A deep learning approach. Retrieved from http://cs229.stanford.edu/proj2017/final-posters/5147906.pdf

Handoyo, A. T. & Suhartono, D. (2021). Sarcasm detection in Twitter--performance impact while using data augmentation: Word embeddings. https://doi.org/10.48550/arXiv.2108.09924

Jamil, R., Ashraf, I., Rustam, F., Saad, E., Mehmood, A. & Choi, G. S. (2021). Detecting sarcasm in multi-domain datasets using convolutional neural networks and long short term memory network model. *PeerJ Computer Science*, 7, e645. https://doi.org/10.7717/peerj-cs.645

Joshi, A., Tripathi, V., Patel, K., Bhattacharyya, P. & Carman, M. (2016, November). Are word embedding-based features useful for sarcasm detection? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. (pp. 1006 -1011). Austin, Texas. Association for Computational Linguistics. Retrieved from https://aclanthology.org/D16-1104.pdf

Khodak, M., Saunshi, N. & Vodrahalli, K. (2018, May). A large self-annotated corpus for sarcasm. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, (pp. 641- 646), Miyazaki, Japan. European Language Resources Association (ELRA). Retrieved from https://arxiv.org/pdf/1704.05579.pdf

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, 86(11), 2278-2324. https://doi.org/10.1109/5.726791

Liebrecht, C. C., Kunneman, F. A. & van Den Bosch, A. P. J. (2013, June). The perfect solution for detecting sarcasm in tweets# not. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, (pp. 29–37), Atlanta, Georgia. Association for Computational Linguistics. Retrieved from https://aclanthology.org/W13-1605.pdf

Liu, B. (2012). Sentiment analysis and opinion mining. In Graeme Hirst (Ed.) *Synthesis Lectures on Human Language Technologies*. 1-167. Springer Cham.

Oprea, S. & Magdy, W. (2020, July). iSarcasm: A Dataset of Intended Sarcasm. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (pp. 1279–1289), Online. Association for Computational Linguistics. Retrieved from https://aclanthology.org/2020.acl-main.118.pdf

Poria, S., Cambria, E., Hazarika, D. &Vij, P. (2016). A deeper look into sarcastic tweets using deep convolutional neural networks. arXiv preprint arXiv:1610.08815

Ptácek, T., Habernal, I. & Hong, J. (2014). Sarcasm detection on Czech and English twitter. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, (pp 213–223), Dublin, Ireland. Retrieved from https://aclanthology.org/C14-1022.pdf

Razali, M. S., Halin, A. A., Ye, L., Doraisamy, S. & Norowi, N. M. (2021). Sarcasm detection using deep learning with contextual features. *IEEE Access*, 9, 68609-68618. https://doi.org/10.1109/ACCESS.2021.3076789

Reyes, A. & Rosso, P. (2012). Making objective decisions from subjective data: Detecting irony in customer reviews. *Decision support systems*, 53(4), 754-760. https://doi.org/10.1016/j.dss.2012.05.027

Riloff, E., Qadir, A., Surve, P., De Silva, L., Gilbert, N. & Huang, R. (2013, October). 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). Retrieved from https://aclanthology.org/D13-1066.pdf

RupaliBagate, R. Suguna (2021). Sarcasm detection of tweets without #sarcasm: Data science approach. *Indonesian Journal of Electrical Engineering and Computer Science*, 23(2), 991-1001. http://doi.org/10.11591/ijeecs.v23.i2.pp993-1001

RupaliBagate, R., Saini, A., Sethi, K., Tomar, H. & Singh, A. (2021). Sarcasm Detection and Explainable AI: A Survey. In *3rd International Conference on Communication and Information Processing (ICCIP)*. Retrieved from https://ssrn.com/abstract=3911955

Sonawane, S. S. & Kolhe, S. R. (2020). TCSD: Term co-occurrence based sarcasm detection from twitter trends. *Procedia Computer Science*, 167, 830-839. https://doi.org/10.1016/j.procs.2020.03.422

Tarigan, J. & Girsang, A. S. (2018). Word similarity score as augmented feature in sarcasm detection using deep learning. *International Journal of Advanced Computer Research*, 8(39), 354-363. http://dx.doi.org/10.19101/IJACR.2018.839002

Van Hee, C., Lefever, E. & Hoste, V. (018). Semeval-2018 task 3: Irony detection in English tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, (pp 39 – 50), New Orleans, Louisiana. Association for Computational Linguistics. Retrieved from https://aclanthology.org/S18-1005.pdf

Venkatesh, B. & Vishwas, H. N. (2021, September). Real Time Sarcasm Detection on Twitter using Ensemble Methods. In *Third International Conference on Inventive Research in Computing Applications (ICIRCA)* (pp. 1292-1297). IEEE. Coimbatore, India. https://doi.org/10.1109/ICIRCA51532.2021.9544841