Sarcasm Identification using different Word Embedding Techniques

IDAVALAPATI VIJAY TARAKA RAMARAO
(Dept. MS in Computer Science)
700742485)

***Abstract*—The computational analysis of user opinions on websites like Twitter, Amazon, IMDB, and Snapchat is known assentiment analysis. The result will take the polarity of positive, negative, or indifferent into account. Sarcasm is one of the most difficult problems in sentiment analysis. Researchers in the field of sentiment analysis tend to deny the existence of sarcasm because it is thought to be too sophisticated. The majority of researchers considered sarcasm to be a subcategory of irony.It involves making a pleasant statement while intending to hurt someone. For both humans and machines, it might be challengingto discern the intent. To appreciate sarcasm, the listener is expected to have some prior understanding of the speaker's context or statement. The most common method for detecting sarcasm in research that considers or exclusively focuses on sarcasm is deep learning, which makes use of context other than the target word. Deep learning and context, however, both require a large number of features. Here, we will examine some sarcasm detection studies and their consensus that more than justtext is required to accurately detect sarcasm. This study also discusses the developments in sarcasm detection research andthe strategies that have been suggested. In this work, we extract features by converting the data into vectors using word2vec and ELMo (Embedded from Language Model). The ability to see the semantic relationship between words is a benefit of Word2Vec. ELMo has the benefit of deep contextualized pretraining betweenwords, or the relationship between the word that is currently being processed and the context of words that have already been processed. In this study, we improved the classification of the sarcastic tweets using LSTM(Long Short Term Memory) with CNN (Convolutional Neural Networks). Tweets were initiallygathered by crawling them using the Twitter API. To gatherpertinent tweets for this study, sarcasm-related keywords and hashtags were generated. The training uses skip-gram as the word2vec model architecture, and in the final evaluation, the best classification results will be compared. Feature extraction is done by looking at each word in the tweet and its vector scoreon the already trained word2vec model. Afterward, the features are passed to the classification step. The implementation of word embedding techniques word2vec, ELMO, LSTM-CNN, and Glove with SVM, Naive Bayes, and K-Means.**

***Index Terms*—Sarcasm Detection, Word Embedding, Data Preprocessing, Vectorization, Twitter API, Machine learning Algorithms.**

I. INTRODUCTION

A crucial technique in opinion mining, sarcasm detection has numerous uses in fields like sales, security, and health. Many organizations and businesses have expressed interest in analyzing Twitter data to see what people think about popular items, political events, or films. The industry has found the field to be quite valuable because it can notify them of what their clients are looking for at any given moment. In the areaof research, it has also quickly gained attention due to its significance and subjectivity. Every day, millions of tweets are published, greatly expanding twitter's content. However, because social media is mainly used for microblogging (tweets

can only be 140 characters long) and uses informal language,it might be challenging to gauge users' sentiment and do sentiment analysis. Sarcasm also complicates sentiment analysis and leads to incorrect categorization of people's opinions. As a result, sentiment analysis accuracy is decreased.

Sarcasm is a way for people to make fun of or express disdain in a sentence or while speaking. Positive words are sometimes used to mask depressing emotions. Sarcasm and irony are often used in social media these days, however they can be hard to spot. When evaluating social media data, the cutting-edge methods of sentiment analysis and opinion mining frequently perform poorly. Sarcasm detection during sentiment analysis, according to Maynard and others, may greatly enhance performance. As a result, the need for a reliable way to spot sarcasm develops.

On social media sites, the use of contextual features to identify sarcasm has recently gained ground. Information from microblogs is highly contextual. Because of this, us- ing content-based linguistic features to classify sentiments becomes less effective and calls for the incorporation of contextual information. Wallace and Kertz's work looked into this phenomenon by showing how conventional classifiers fall short when dealing with situations where people need more context. When there is a lack of data and reliance on solely content features for prediction, the most related techniques for sarcasm recognition generally generate low performance and outcomes that are not generalizable. But in order to overcome these problems, more data is needed, such as contextual data. In order to operate well in sentiment analysis situations, The sarcasm identification system is necessary for other reasons as well.

However, it should also carry out the more difficult duty of distinguishing it from tweets with positive and negative emotions. Thus, the majority of relevant algorithms have traditionally concentrated on content-based features that solely take tweets' contents into account. However, the majority of language studies on sarcasm contend that using contextual characteristics that take tweet context into account improves prediction performance.

Positive words are sometimes used to mask depressing emotions. In this research, we suggest a reliable technique to recognize a sarcastic tweet. Our method takes into account the different sarcasm-indicating features, including word embedding and numerous supervised machine-learning models based on extracted features. These features include lexical-based features, sarcastic-based features, and contrast-based. After the evaluation and result in the analysis sections of this paper, we propose an efficient machine learning model and feature set to improve sarcasm detection in sentiment analysis and obtain greater accuracy.

II. Motivation

Nowadays, most people utilize social media channels to express their feelings on products or real-life incidents. For example, if an individual buys a product from Amazon and then utilizes it, he or she can express their opinions on the product. Many people are likely to have a viewpoint on the same good or service. These reviews can assist com- panies in increasing their sales or improving the quality of their products. Humans reading all these reviews are now a time-consuming task. As a result, business organizations use machine learning techniques to boost productivity. Sarcastic text purposely misleads sentiment analysis and causes wrong assumptions.

A few people use productive language to express themselves negatively. These positive words produce a positive sentiment analysis result, which is an incorrect prediction. We will detect sarcasm in text using a machine learning algorithm and evaluate the effectiveness of methods on data sets using metrics such as accuracy, precision, F1-value, and recall. Although many previous works on irony and sarcasm detection have obtained good results while training ML models on datasets, these works use various feature engineering approaches such as n-grams, hashtag features, lexical features, and so on while neglecting the context. While word embeddings use neural network architectures to learn distributed word vectors by utilizing context-specific information in large textual data. Word embedding methodologies like LSTMs, ELMo, BERT, Word2vec, CNNs, and GloVe are fast and accurate at producing language patterns and other downstream tasks, such as comprehension, syntactic and semantic attributes, and the strong correlation between words.

III. Main Contributions and Objectives

- We propose learning sentiment embeddings that encode the sentiment of texts in continuous word representation.
- First, we will get the Twitter labeled data from Twitter API. then we perform several preprocessing steps to clean the data.
- We suggest developing embeddings that continuously represent words while encoding the sentiment of texts.
- we use Glove, ELMo, word2vec, and hybrid LSTM-CNN embedding models to transform words into vectors.
- we develop several classification models like SVM, Naive Bayes, and K-Means to learn sentiment embeddings and to distinguish between sarcastic and nonsarcastic tweets.
- We verify the effectiveness of sarcasm analysis by experimenting with each type of embeddings with all the classification algorithms.
- And then check which algorithm is outperforming the other models and consider it the best model for sarcasm analysis.

IV. Related Work

Sarcasm identification task has been studied by employing different methods, including lexicon-based, conventional machine learning, deep learning, or even a hybrid approach. Besides, several reviews on sarcasm detection have also been conducted. For instance, Eke, et al. [1] performed SLR on sarcasm recognition in text data. The study was carried out by considering 'dataset collection, preprocessing techniques,

feature engineering techniques (feature extraction, feature selection, and feature representation), classification algorithms, and performance measures.' The study revealed that content-based features are the most employed features for sarcasm classification. The study also revealed that the study also showed that the most used measures for assessing the performance of classifiers. Moreover, the study also revealed that when there is an imbalance in the class distribution of the dataset, the AUC performance measure is the right choice due to its robustness in resisting the skewness in the dataset. The review concludes with recent challenges and suggestsopen research directions that offer solutions to the sarcasm recognition research problem.

In a different study, ONAN, et al. [2] suggested a technique for identifying satire in Turkish news articles. By taking into account the language and psychological feature sets, the authors used linguistic inquiry and word count tools to extract features. In this work, a word embedding scheme, five deep learning architectures, and ensemble learning were takeninto consideration. The significance of the suggested methods was demonstrated by the experimental study of the proposed methodology, which revealed that the deep learning approach outperformed other approaches.

The punctuation has been deleted, and the words have been tokenized. The subsequent process involves simply keeping the elements required for lemmatization. Additionally, the tokens that are regularly utilized but carry little value are eliminated. A sentiment lexicon is being used to label the complicated data that was gathered for analysis. Using polarity labeling, polarity encoding, compound scores, and value scores, sentiment analysis can be utilized to do this. Using Word2Vec's CBOW and Skip-gram [3] methods, the Features are retrieved.

This article presents a scenario of applying deep learning models based on transformers and transfer learning to categorize a given text as sarcastic or not. BERT, RoBERTa, and the RNN-based models LSTM, BiLSTM, and BiGRU [4] are the models that are used. The ensembling method is used. The study also illustrates how the performance of the model will be impacted by the various class distributions.

Many language models have been built recently to address the problem with LSTM and convolutional neural networks. All of these models make use of the'self-attention mechanism'. Through the use of this technique, the words in a phraseare digested continuously as opposed to one at a time. The simultaneous processing enables the words to recognize the relationship in which they are arranged in a phrase as well as the context in which they are used. These models' efficiency is significantly higher than that of LSTM-based models because the words are processed concurrently. Additionally, they are more accurate at performing linguistic tasks like sarcasm and sentiment analysis [5].

The author's proposed method divided the sarcasm recognition algorithm into four distinct analyses: features for the analysis linked to syntactics, sentiment, punctuation, and patterns. By classifying words into two distinct categories—"CI" and "GFI," they provided an efficient and trustworthy strategy for sarcasm identification. The GFI class focuses more on the word's grammatical function whereas the CI class emphasizes the significance of the word's contents in the sentence. For prediction purposes, the Random Forest classifier was used, and results showed an accuracy of 83.1 per and a precision of 91.1per.Accordingly, the comparative study showed that the pattern-based technique performed better than other approaches. The study, however, placed more emphasis on the expression's word patterns, which fall short of accurately conveying all sardonic undertones. The sarcastic analysis showed that the aforementioned methods worked best. The significance of background information in sarcastic utterances to resolve the ambiguity connected with a sarcastic statement, however, was overlooked. By taking into account feature fusion strate- gies for sarcasm classification using textual data, the context embedding that takes into account both local and global context information has been used in this research to develop the deep learning and BERT model features. The model has been tested using three benchmark datasets provided by Riloffet al. [6], Ghosh and Vale [7], and IAC-v2 [8].

They used a voting-based classification system and polarity-based sentiment analysis on live Twitter data to do sentiment analysis in a work on Sentiment Analysis Using Machine Learning and Deep Learning. Algorithms for machine learning are employed. Naive Bayes, MNB Classifier, Bernoulli Classifier, Logistic Regression, Stochastic Gradient Descent, Linear SVC, and NuSVC Classifier are a few examples of algorithms. techniques to identify opinions on tweets based on polarity. The feature selection and mapping processes use the Deep Learning Method LSTM-CNN [9]. When compared to machine learning algorithms, deep learning techniques produced the best results (Chandra and Jana, 2020).

Gui et alresearch .'s [10] from 2022 suggested a multitask mutual learning framework for learning multiple machine learning algorithms simultaneously. They found that utilizing IMDB datasets to test their algorithm produced better sentiment analysis findings. Results are excellent even when using full datasets and a table with fewer vocabulary terms. They employed RCNN, NSC, and convolutional neural networks as well as recurrent neural networks.

Academic feedback is essential in schools for preserving a good rapport between students, teachers, and parents. To do analysis, a VADER function from NLP was employed. The attitude of the student, or how he or she is acting in a classroom, can be determined by performing sentiment analysis on datasets and evaluating grade, attendance, and aptitude (Watkins et al., 2020) [11].

V. Proposed Framework

This describes the framework for comparing different word embedding techniques for sarcasm detection. The workflow of this framework contains five steps:

- Data collection using twitter API
- Preprocessing the data by removing punctuations, stop words, hashtags, URLs, converting the sentences into tokens and applying stemming on the dataset.

- To compare various word embedding techniques like GloVe, ELMo, Word2vec, LSTM, and CNN+LSTM we need to extract word embeddings i.e., we need to convert our categorical data into vectors to train our machine learning models.
- After extracting word vectors using embedding techniques, we need to train machine learning algorithms like SVM, Na¨ıve Bayes and k-means.
- Finally, we need to compare the values of different matrices like f-value, precision, recall, and accuracy of all the algorithms we trained on the dataset. In the next subsection we will briefly describe about the steps mentioned above:

A. Data Collection:

In this proposed framework we extract the raw data which contains sarcastic and non-sarcastic tweets by using the Twitter API. After extracting the dataset, we labeled our data based on the two categories, the tweets with sarcastic hashtags are labeled with 0 and the tweets with the non-sarcastic hashtag are labeled with 1. The dataset we are using contains 16,486 tweets which include 8,954 sarcastic tweets and 7,532 non- sarcastic tweets.

B. Data Preprocessing:

The dataset we extracted from Twitter API contains some noisy data which affects the performance of our classification process. To improve the performance of the classification mod- els we need to preprocess the data. In this framework we have done we are preprocessing data to remove the noise presentin our dataset. In the first step, we are removing the URLs, repeated words present in the tweets. We need to remove the punctuation from the data. After removing the punctuation, we need to remove the stop words i.e., prepositions present in our data by downloading the stop words file which contains the stop words we need to remove from the python NLTK module.Stop words present in our dataset will affect the performance of our model. After stop word removal we need to convertour sentences into tokens using the NLTK tokenizer. Finally, we need to perform stemming on our dataset. Stemming is an important preprocessing technique in NLP applications. To perform stemming we used the porter stemmer of the NLTK module. This stemming is used to reduce the vocabulary size of the tweet text. After stemming we need to perform another method that reduces the text size lemmatization.

C. Extracting word vectors:

In this approach to convert our categorical data into numerical values, we are using five types of word embed- ding techniques like GloVe, ELMo, word2vec, LSTM, and CNN+LSTM. Using these techniques, we will extract the word vectors to train our machine-learning models. This process can be done in two steps. Firstly, we need to train our dataset with word embedding models and extract the word vectors. In addition to that, we need to perform dimensionality reduction techniques like PCA to reduce the dimensions of the data to fit into our model. However, in this proposed work we imported pre-trained models of GloVe and ELMo to extract the word vectors.

Word embedding techniques we used in this framework are: GloVe is an embedding strategy that conveys context-specific features from the text. GloVe is a word embedding strategy that is based on the weighted least-square model and trains not only on the local contextual information of the word but also on the global word-to-word co-occurrence count in a corpus. This is known as parallel execution, and it allows GloVe to model on a big dataset more efficiently. The preprocessed data will be used to extract word embeddings using GloVe in this pre-trained word embedding approach, and these features will be used as a feature for modeling. ELMo is another word embedding strategy which is used to extract context-based word vectors. Over a dual-layer, bidirectional language model, ELMo word vectors are computed (biLM). This biLM model is made up of two layers that are stacked on top of each other. Each layer has two passes, one forward and one backward. Word2vec method uses two methods like continuous bag of words and skip gram to extract word vectors. Python genism module contains word2vec model to train our data to extract feature vectors. In this proposed framework we are using skipgram model to extract numerical data. This word2vec method predicts the next incoming word based on the present word but it doesnot rely on the sequence order which is the drawback. To overcome this, in this model we are using ELMo model to extract word vectors. Recurrent neural networks (RNNs) with the ability to learn lengthy dependencies between sequential pattern time steps include LSTM networks. A word embeddinglayer learns the word embedding by mapping a pattern from word indexes to embedding vectors during training. Finally, after extracting the word vectors using these techniques, we need to reduce the dimensionality of the word vectors into twodimensions using principal component analysis. This dimen- sionality reduced features can be used to train our machine learning algorithms.

D. Classification:

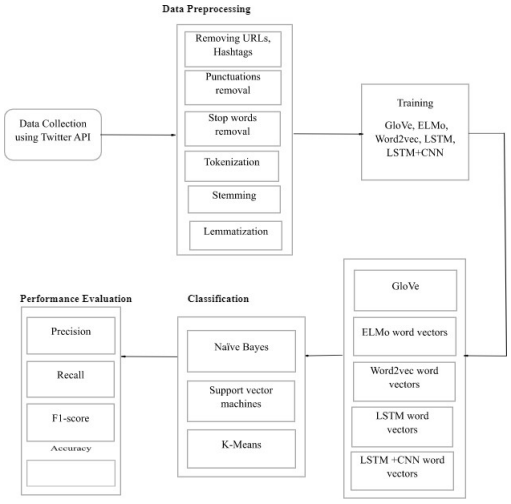
Both supervised machine learning Algorithms and unsupervised machine learning algorithms are trained using word vectors.

- Na¨ıve Bayes: The Nave Bayes classifiers include the Gaussian Nave Bayes, Multinomial Nave Bayes, and Bernoulli Nave Bayes. For this project, we settled on the Multinomial Naive Bayes classifier due to its strong performance with distinctive features. Multinomial naive Bayes is a probabilistic classification method that assumes there is no association between various variables, such as the likelihood that a particular record or piece of data belongs to a particular class, and utilizes theBayes probability theorem to forecast unknown classes. The most likely class is the one with the most promise.
- Support vector machines: The SVM technique seeks to locate an N-dimensional space hyperplane that distinctly classifies the data points. The two sets of data points can

be split using a variety of hyperplanes. We seek the plane with the largest margin or separation between the data points in the two classes.

- K-Means: K-means clustering is an unsupervised learning technique for clustering data points. The method iteratively separates the datasets into K clusters by reducing the variance within each cluster. Each data point is initially assigned to one of the K clusters at random.Following that, we calculate the center of each cluster andreassign each data point to the cluster with the nearest centroid. This procedure is repeated until the cluster assignments for any given data point no longer change. K-means clustering requires us to choose K, the number of clusters into which we want to group the data
- Conclusion: In this proposed framework, we are training word vectors generated from word embedding methods to determine which word embedding technique performs better with machine learning algorithms. These results are compared based on the recall, precision, f1-score, and accuracy values.

  TN: True Negative TP: True Positive
  FN: False Negative FP: False Positive
  Accuracy: TN+TP / P +N
  Recall: TP / FN + TP
  Precision: TP / FP + TP
  F1-score: 2 * Precision * Recall / Precision + Recall



VI. Experimentation:

The first step in experimentation is data collection. In this process we collected raw data containing 16,486 tweets using twitter API. In this study, after collecting data we applied several data preprocessing steps to remove any unwanted or noisy data present in the data. These data are being trained on pre-trained word embedding methods like GloVe, ELMo are adopted. We also trained our data on the word2vec, LSTM, and LSTM+CNN to extract the word vectors fromthe categorical data. During the generation of word vectors,we split our data into 70 percent for training the data andthe remaining 30 percent for testing the data. Here are the few hyperparameter settings used during our experimentation:

Hyper-parameter	Value
Dimension of GloVe vector	100
Hidden units for LSTM	100
Minimum batch size	64
Regularization	Dropout operation
Drop-out rate	0.2
Learning rate	0.2
Word embeddings	GloVe
Activation function	Sigmoid
Epoch	3
Optimization algorithm	Adam Algorithm

Using principal component analysis, we reduced the dimensionality of the word vectors into two dimensions by setting n-dimensions to be equal to two. These results are used to give input to our machine learning models by splitting our data into 70 percent for training and 30 percent for testing part to predict the best word embedding method to get contextualized features for sarcasm identification.

VII. Data Description:

In this proposed framework we extract the raw data which contains sarcastic and non-sarcastic tweets by using the Twitter API. After extracting the dataset, we labeled our data based on the two categories, the tweets with sarcastic hashtags are labeled with 0 and the tweets with the non-sarcastic hashtag are labeled with 1. The dataset we are using contains 16,486 tweets which include 8,954 sarcastic tweets and 7,532 non- sarcastic tweets. We gathered twitter data from the time between 2015 and present. Using the hashtag, we gathered dataof sarcastic and non-sarcastic tweets. For example, to gather sarcastic tweets we used sarcastic or saracasm keywords and similarly for non-sarcasm. Later we gathered the data togetherinto one file by using concat method of pandas and shuffled the data to mix both the category tweets. After labelling them based on their category we removed data column which represents the time stamp when the user has been tweeted because it doesn't effect the performance of our models.

VIII. Results/ Experimentation and Comparison/Analysis

This section displays the results we obtained after the experimentation on the dataset extracted from twitter API. Machine learning algorithms like Na¨ıve Bayes, Support vector machine, and K-Means are trained on the word vectors generated from the word embedding techniques. Among four different matrices like accuracy, precision, recall, and f1-score, we used precision as a major evaluation. Here, we are reporting the all the values of matrices. Using these values, we are predicting the best word embedding technique for sarcasm identification

A. Results obtained for Word2vec word vectors:

Word2vec word vectors are trained on three machine learning algorithms and those results as follows:

B. Na¨ıve Bayes:

```
              precision    recall  f1-score   support

           0       0.95      0.75      0.84      2705
           1       0.76      0.95      0.85      2241

    accuracy                           0.84      4946
   macro avg       0.86      0.85      0.84      4946
weighted avg       0.87      0.84      0.84      4946

[[2039  666]
 [ 102 2139]]
accuracy is 0.8447230084917104
```

C. Support Vector Machine (SVM):

```
              precision    recall  f1-score   support

           0       0.94      0.96      0.95      2705
           1       0.95      0.93      0.94      2241

    accuracy                           0.94      4946
   macro avg       0.94      0.94      0.94      4946
weighted avg       0.94      0.94      0.94      4946

[[2593  112]
 [ 165 2076]]
accuracy is 0.9439951475940154
```

D. K-Means:

```
nclusters = 3  # this is the k in kmeans
km = KMeans(n_clusters=nclusters)
km.fit(X)

y_cluster_kmeans = km.predict(X)
from sklearn import metrics
score = metrics.silhouette_score(X, y_cluster_kmeans)
print(score)
```

```
0.4742779908167591
```

When we trained our word2vec word vectors over Na¨ıve Bayes, Support vector machine, and k-means, support vector machine(svm) outperformed the remaining two algorithms with accuracy of 94.3per where for na¨ıve bayes 84.4per and k-means with 47.2per silhouette score. Overall, when we compared all the values of precision, recall, and f1-score support vector machines got better results.

Results obtained for ELMo word vectors:

ELMo word vectors are trained on three machine learning algorithms and those results as follows:

E. Na¨ıve Bayes:

```
              precision    recall  f1-score   support

           0       0.52      0.38      0.43      2280
           1       0.56      0.69      0.62      2629

    accuracy                           0.55      4909
   macro avg       0.54      0.53      0.53      4909
weighted avg       0.54      0.55      0.53      4909

[[ 855 1425]
 [ 805 1824]]
accuracy is 0.5457323283764515
```

F. Support Vector Machine (SVM):

```
              precision    recall  f1-score   support

           0       0.53      0.38      0.44      2280
           1       0.57      0.71      0.63      2629

    accuracy                           0.55      4909
   macro avg       0.55      0.54      0.54      4909
weighted avg       0.55      0.55      0.54      4909

[[ 867 1413]
 [ 775 1854]]
accuracy is 0.554288042371155
```

G. K-Means:

```
nclusters = 3  # this is the k in kmeans
km = KMeans(n_clusters=nclusters)
km.fit(X)

y_cluster_kmeans = km.predict(X)
from sklearn import metrics
score = metrics.silhouette_score(X, y_cluster_kmeans)
print(score)
```

```
0.42760342
```

When we trained our ELMo word vectors over Na¨ıve Bayes, Support vector machine, and k-means, support vector machine(svm) outperformed the remaining two algorithms with accuracy of 55.4per where for na¨ıve bayes 54.5per and k-means with 42.7per silhouette score. Overall, when we compared all the values of precision, recall, and f1-score support vector machines got better results.

Results obtained for GloVe word vectors:

GloVe word vectors are trained on three machine learning algorithms and those results as follows:

H. Na¨ıve Bayes:

```
              precision    recall  f1-score   support

           0       0.55      0.95      0.70      2705
           1       0.50      0.06      0.10      2241

    accuracy                           0.55      4946
   macro avg       0.53      0.51      0.40      4946
weighted avg       0.53      0.55      0.43      4946

[[2581  124]
 [2115  126]]
accuracy is 0.547310958350182
```

I. Support Vector Machine (SVM):

```
              precision    recall  f1-score   support

           0       0.55      0.99      0.71      2705
           1       0.56      0.01      0.02      2241

    accuracy                           0.55      4946
   macro avg       0.55      0.50      0.36      4946
weighted avg       0.55      0.55      0.39      4946

[[2690   15]
 [2222   19]]
accuracy is 0.5477153255155681
```

J. K-Means:

```
nclusters = 3  # this is the k in kmeans
km = KMeans(n_clusters=nclusters)
km.fit(X)

y_cluster_kmeans = km.predict(X)
from sklearn import metrics
score = metrics.silhouette_score(X, y_cluster_kmeans)
print(score)
```

```
0.64388
```

When we trained our GloVe word vectors over Na¨ıve Bayes, Support vector machine, and k-means. Support vector machine and na¨ıve bayes got the same accuracy and k-means with 64.3per silhouette scores. Precision and recall values of both the algorithms got same values but f1-score na¨ıve bayes got 43per while SVM with 39per.

Results obtained for LSTM word vectors:

LSTM word vectors are trained on three machine learning algorithms and those results as follows:

K. Na¨ıve Bayes:

```
              precision    recall  f1-score   support

           0       0.55      1.00      0.71      2705
           1       0.80      0.00      0.00      2241

    accuracy                           0.55      4946
   macro avg       0.67      0.50      0.36      4946
weighted avg       0.66      0.55      0.39      4946

[[2704    1]
 [2237    4]]
accuracy is 0.547513141932875
```

L. Support Vector Machine (SVM):

```
              precision    recall  f1-score   support

           0       0.55      1.00      0.71      2705
           1       0.50      0.00      0.01      2241

    accuracy                           0.55      4946
   macro avg       0.52      0.50      0.36      4946
weighted avg       0.53      0.55      0.39      4946

[[2698    7]
 [2234    7]]
accuracy is 0.5469065911847958
```

M. K-Means:

```
nclusters = 3  # this is the k in kmeans
km = KMeans(n_clusters=nclusters)
km.fit(X)

y_cluster_kmeans = km.predict(X)
from sklearn import metrics
score = metrics.silhouette_score(X, y_cluster_kmeans)
print(score)
```

```
0.43765166
```

When we trained our LSTM+CNN word vectors over Na¨ıve Bayes, Support vector machine, and k-means, Na¨ıve Bayes gets the better accuracy of 54.7per while the remaining two algorithms 54.6per for support vector machine and k-means with 43.7per silhouette score. Overall, when we comparedall the values of precision, recall, and f1-score, naive bayesgot better results for precision and f1-score while both the algorithms got same results for recall.

N. Conclusion

: Overall, when we compared all the results, word vectors generated from word2vec got better results with an accuracy of 94per when we trained with support vector machine. GloVe word vectors got better results when we trained with k-means. ELMo, GloVe, LSTM, and LSTM+CNN got almost the same values when we trained out machine learning models with our dataset. In future, we need to apply more preprocessing methods to our dataset to perform better on our machine learning algorithms and need to apply more techniques for extracting contextualized feature sets. In order to get better results for sarcasm identification, we need to get more feature sets related to context of the text because we trained ourmodels only with different word embedding features.

REFERENCES

[1] C. I. Eke, A. A. Norman, L. Shuib, and H. F. Nweke, “Sarcasm identification in textual data: Systematic review, research challenges, and open directions,” Artif. Intell. Rev., vol. 53, pp. 4215–4258, Nov. 2019.

[2] A. Onan and M. A. Tocoglu, “Satire identification in Turkish news articles based on ensemble of classifiers,” Turkish J. Electr. Eng. Comput. Sci., vol. 28, no. 2, pp. 1086–1106, Mar. 2020.

[3] T. P. Kumar and B. V. Vardhan, ”Multimodal Sentiment Analysis using Prediction-based Word Embeddings,” 2022 International Conference on Edge Computing and Applications (ICECAA), 2022, pp. 258-262, doi: 10.1109/ICECAA55415.2022.9936350.

[4] Malak Abdullah, Jumana Khrais, Safa Swedat, ”Transformer-Based Deep Learning for Sarcasm Detection with Imbalanced Dataset: Resampling Techniques with Downsampling and Augmentation”, 2022 13th International Conference on Information and Communication Systems (ICICS), pp.294-300, 2022.

[5] S. M. Adeel Ibrahim, S. Manoharan and X. Ye, ”A Study of Using Language Models to Detect Sarcasm,” 2020 IEEE Conference on e-Learning, e-Management and e-Services (IC3e), 2020, pp. 38-42, doi: 10.1109/IC3e50159.2020.9288427.

[6] E. Riloff, A. Qadir, P. Surve, L. D. Silva, N. Gilbert, and R. Huang, “Sarcasm as contrast between a positive sentiment and negative situation,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2013, pp. 704–714.

[7] A. Ghosh and D. T. Veale, “Fracking sarcasm using neural network,” in Proc. 7th Workshop Comput. Approaches Subjectivity, Sentiment Social Media Anal., 2016, pp. 161–169.

[8] S. Oraby, V. Harrison, L. Reed, E. Hernandez, E. Riloff, and M. Walker, “Creating and characterizing a diverse corpus of sarcasm in dialogue,” Tech. Rep., 2017.

[9] Chandra, Y., Jana, A. (2020). ”Sentiment analysis using machine learning and Deep Learning. 2020 7th International Conference on Computing for Sustainable Global Development (INDIACom)”, 1–4.

[10] L. Gui, J. Leng, J. Zhou, R. Xu and Y. He, ”Multi Task Mutual Learning for Joint Sentiment Classification and Topic Detection,” in IEEE Transactions on Knowledge and Data Engineering, vol. 34, no.4, pp. 1915-1927, 1 April 2022, doi: 10.1109/TKDE.2020.2999489.

[11] J. Watkins, M. Fabielli and M. Mahmud, ”SENSE: a Student Performance Quantifier using Sentiment Analysis,” 2020 International Joint Conference on Neural Networks (IJCNN), 2020, pp. 1-6, doi: 10.1109/IJCNN48605.2020.9207721.

[12] Pandya, V., Somthankar, A., Shrivastava, S. S., and Patil, M. (2021). Twitter sentiment analysis using machine learning and Deep Learning Techniques. 2021 2nd International Conference on Communication, Computing and Industry 4.0 (C2I4), 1–5.

[13] Poornima, A., and Priya, K. S. (2020). A comparative sentiment analysis of sentence embedding using machine learning techniques. 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), 493–496.

[14] Rathor, S., and Prakash, Y. (2022). Application of machine learning for sentiment analysis of movies using IMDB rating. 2022 IEEE 11th International Conference on Communication Systems and Network Technologies (CSNT), 196–199.

[15] A. M. Putri, D. Ananda Putra Basya, M. T. Ardiyanto and I. Sarathan, ”Sentiment Analysis of YouTube Video Comments with the Topic of Starlink Mission Using Long Short Term Memory,” 2021 International Conference on Artificial Intelligence and Big Data Analytics, 2021, pp. 28-32, doi: 10.1109/ICAIBDA53487.2021.9689718.

[16] D. K. Sharma, B. Singh and A. Garg, ”An Ensemble Model for detecting Sarcasm on Social Media,” 2022 9th International Conference on Computing for Sustainable Global Development (INDIACom), 2022,pp. 743-748, doi: 10.23919/INDIACom54597.2022.9763115.

[17] F. Hasnat et al., ”Understanding Sarcasm from Reddit texts using Supervised Algorithms,” 2022 IEEE 10th Region 10 Humanitarian Technology Conference (R10-HTC), 2022, pp. 1-6, doi: 10.1109/R10-HTC54060.2022.9929882.

[18] A. M. Putri, D. Ananda Putra Basya, M. T. Ardiyanto and I. Sarathan, ”Sentiment Analysis of YouTube Video Comments with the Topic of Starlink Mission Using Long Short Term Memory,” 2021 International Conference on Artificial Intelligence and Big Data Analytics, 2021, pp. 28-32, doi: 10.1109/ICAIBDA53487.2021.9689718.

[19] R. Kumar and S. Sinha, ”Comprehensive Sarcasm Detection using Classification Models and Neural Networks,” 2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS), 2022, pp. 1309-1314, doi: 10.1109/ICACCS54159.2022.9785305.

[20] A. Onan and M. A. Toçoğlu, ”A Term Weighted Neural Language Model and Stacked Bidirectional LSTM Based Framework for Sarcasm Identification,” in IEEE Access, vol. 9, pp. 7701-7722, 2021, doi: 10.1109/ACCESS.2021.3049734.