Detecting Sarcasm in News Headlines

Onyinye Chudi-Iwueze[1] and Haithem Afli[2]

ADAPT Centre,
Cork Institute of Technology, Cork, Ireland
[1]o.chudi-iwueze@mycit.ie,[2]haithem.afli@cit.ie

Abstract. The use of sarcasm dates as far back as communication goes. Sarcasm is expressed in words, facial expression and can be noticed in the intonation of voice. In this digital age, sarcastic comments are passed everyday, via tweets, comment sections of different social media outlets, and even in news headlines. Some of these headlines can be misunderstood and taken to mean something different than the original intentions. This leads to a need to detect sarcasm especially in the news and on social media. Detection of sarcasm from text has its challenges because text lacks intonation and deflection of voice that occurs when a sarcastic statement is made vocally by a human. This paper concentrates on the effect of different feature encoding techniques applied to text for feature extraction for machine learning models. A deep learning model is also applied and the results are compared. Prior to feature extraction, data pre-processing techniques like tokenization, removal of stop-words and punctuation are applied by researchers in any work involving text analysis. These pre-processing techniques are widely used and accepted and are also applied in this project. Different feature extraction methods like Count Vectorizer, Term Frequency-Inverse Document Frequency and word embedding were implemented in this experiment. The Support Vector Machine, Naive Bayes and Logistic Regression were the traditional machine learning algorithms used in this research. Convolutional Neural Network was also used. Results of these algorithms are recorded in this paper and these include the F1-score, precision, recall and the accuracy scores. The results of these different algorithms for the methods of feature extraction examined in this paper show that a combination of more than one technique is better suited for the purpose of classifications. The results of this paper show

Keywords: Sarcasm Detection · Convolutional Neural Networks (CNN) · Feature Extraction · Machine Learning.

1 Introduction

In the last decade, there has been an increase in the use of online resources for dissemination of information. These texts have different characteristics that are explored by the use of Natural Language Processing techniques of machine learning for knowledge and insight. One of such important characteristics is the presence of sarcasm in text. Sarcasm is a complex act of communication that allows speakers the opportunity to express sentiment-rich opinions in an implicit way [1]. There is an abundance of sarcasm in texts distributed online and on social media. This includes news headlines and other media. Sarcasm is classically defined as the process of intentional misuse of words to convey some meaning (usually the opposite of what is said) [1]. For example, if a person says *'How lucky am I? I caught the corona virus!"* It is clear that although the words used are positive, the real meaning is negative, making the statement a sarcastic one. Sarcasm is characterised by the use of irony that reflects a negative meaning.

Although sarcasm is well known and widely used, it is challenging not only for computers but also for humans to detect promptly. Some humans find it difficult to identify and understand the use of sarcasm [2]. Due to this fact, the presence of sarcasm, if not detected and accounted for can affect other machine learning ventures like sentiment analysis and opinion mining. This makes the detection of sarcasm a crucial task. Automated sarcasm detection can be seen as a text classification problem. Text data is one of the simplest forms of data. Machine learning algorithms are unable to process non-numerical data and as such the need for feature extraction arises. It is important to extract useful information from any forms of data, especially unstructured forms like text data. Feature extraction and the proper representation of text for classification purposes is an important factor that affects the accuracy of classification. This paper explores the use of the Count Vectorizer, Word Level TF-IDF, Character Level TF-IDF and N-Grams Level TF-IDF on different supervised learning algorithms and the accuracy of these models is measured and compared. The Word2Vec is also an efficient feature extraction technique which is widely used. The Word2Vec is used for the Deep Learning Algorithm explored in this paper. Other methods like Doc2Vec and LDA are not discussed in this paper as they are outside the scope of this project.

The rest of this paper is arranged as follows: Section 2 goes over some related work both in sarcasm detection and feature extraction methods. Section 3 details the methodology and which models were applied for the analysis. Section 4 outlines the results of the application of the different feature extraction techniques to the chosen models and describes which performed best.

2 Related Work

2.1 Feature Extraction

Text extraction describes the process of taking words from text data and transforming them into a numerical feature set that is useful for a machine learning classifier [13]. Extracting features is useful for all analyses involving text data in all domains. In the work detailed in [8] the impact of pre-processing is discussed and reviewed. The researchers determine the importance of 'slang' and correct spelling. SVM classifier is applied for their experiment. Another researcher uses vector representations to provide the solution to the problem of sentiment analysis [9], obtaining an accuracy of (86%) which is relatively high. Another study considers the use of four data sets and the the use of Bag-Of-Words Features,lexicon and part of speech based features [10]. An ensemble model of SVM, Logistic Regression and Naive Bayes was implemented for the analysis. The authors in [11] applied three levels of feature extraction techniques. They also applied three classifiers in their analysis. Another group of researchers in the work detailed in [12] applied ten different feature extraction techniques to a sentiment analysis problem. They concluded that feature extraction techniques applied for a problem have the potential to improve the performance of a classifier in any given problem. Paper [7] compares the results of 6 algorithms when features are extracted using Bag-Of-Words and TF-IDF. They conclude that the TF-IDF gives better results of up to 4% difference. Much of the research experiments are done with the SVM, Logistic Regression and Naive Bayes algorithms for the traditional supervised machine applications. This informed the use of these algorithms for this analysis. A CNN model is also run at the end of the analysis using the Count Vectorizer vector as the features and also using features generated with word2vec.

2.2 Sarcasm Detection

The detection of sarcasm is very crucial in the domain of sentiment analysis and opinion mining. Different machine learning algorithms have been applied to the problem. Some researchers used the Naive Bayes Classifier and Support Vector Machines for analyses of social media data in Indonesia [5]. The classifiers used in the work detailed in [5] performed well for the task and that informed the decision to use Naive Bayes and the SVM for this analysis. Another body of work that applied traditional supervised techniques to the problem is described in [7]. They used support vector machines with Tf-IDF and Bag-Of-Words. The work by Davidov et.al [6] details a technique of sarcasm detection using semi-supervised methods on two different data set, one containing the reviews on Amazons products and one with tweets collected from Twitter. These researchers concentrated on features like punctuation, sentence syntax, hashtags used etc. The proposed system had an accuracy of over 75% [6]. The work by [14] shows the use of Support Vector Machines also for the detection of sarcasm from news headlines. The optimal method of feature extraction used in this work results in an accuracy score of about 80%. Some other researchers applied deep learning techniques to the problem [2]. Gathering data for the purpose of sarcasm detection is challenging and even more so for supervised learning. This is especially the case for social media data. This research uses some of the methods described in the previous works with better results for accuracy and other measures as well.

3 Methodology

3.1 Data set

The data used for this analysis was taken from Kaggle. The news headlines were curated into two different files for competitions on Kaggle. Each file was a JSON file, containing news headlines and the label named 'is_sarcastic' that indicates the presence or absence of sarcasm in that headline. The two files were joined together to create a bigger data set for the analysis. The complete data used in the analysis contains over 50000 headlines. The data set is not significantly imbalanced but there were a few more non-sarcastic news headlines than the sarcastic headlines. Table 1 below shows a few rows in the data and the labels for those data points.

	Headline	**Is_sarcastic**
1	former versace store clerk sues over secret 'black code' for minority shoppers	0
2	why writers must plan to be surprised	0
3	boehner just wants wife to listen, not come up with alternative debt-reduction ideas	1
4	remembrance is the beginning of the task	0
5	4 lessons prison taught me about power and control	0
6	top snake handler leaves sinking huckabee campaign	1
7	courtroom sketch artist has clear manga influences	1
8	stock analysts confused, frightened by boar market	1
9	gillian jacobs on what it's like to kiss adam brody	0
10	diy: sports equipment closet	0

Table 1. Cross section of some headlines in the data set

Table 2 below shows the distribution of headlines and tokens among the train and test sets.

Sarcasm Data Set	
Data	Number of Tokens
Train	Sarcastic: 19010 Non-Sarcastic: 22486
Test	Sarcastic: 6348 Non-Sarcastic: 7484

Table 2. Table showing distribution of headlines between test and train

Figure 1 below shows the distribution of headlines in the data set. There are more non-sarcastic emails but the classes are not grossly imbalanced.

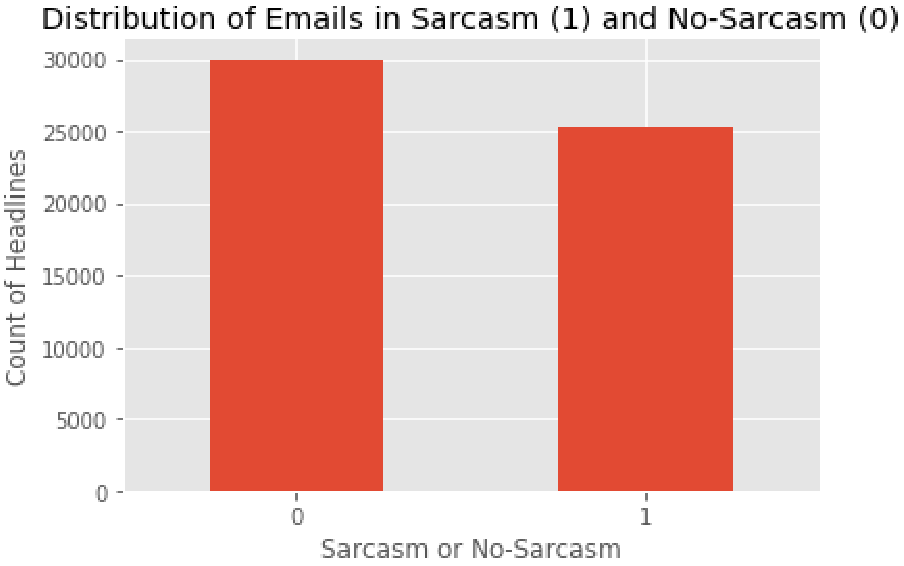

Fig. 1. Distribution of Headlines in the Data set.

3.2 Data Pre-processing

The first step in the pre-processing pipeline was the conversion of all the text to lower case. This is done to achieve uniformity of words. Unwanted symbols like digits and newlines were removed also. All punctuation were also removed from the data set. After these, the texts were tokenized. Tokenization is the breaking down of sentences into words [7]. Stopwords are words that are commonly used in any language. They do not contribute any contextual meaning to the data and as such should be eliminated. Removal of stopwords was the last step in the pre-processing pipeline and the data was thus ready for further preparation prior to model building.

3.3 Feature Extraction

Count Vectorizer The Count Vectorizer is a simple means used to tokenize and build vocabulary of known words. It is also used to encode new documents using the vocabulary that has been created. The product of the count vectorizer is an encoded vector which has the length of the entire vocabulary of the document and a count for the number of times each word appeared in the document. Count Vectorizer was implemented using scikit-learn for this project.

Term Frequency – Inverse Document Frequency (TF-IDF) TF-IDF is a popular and well recognized method for the evaluation of the importance of a word in a document. Term Frequency measures the frequency of a term in a document. For a variety of documents with different lengths, the probability of a word occurring more times in a longer document is present. Due to this fact, normalization is required.

$$TF(t) = \frac{N}{T}$$

where N = Number of times term t appears in a document
and TN = Total number of terms in the document

The Inverse Document Frequency is the measure of importance of a term. This is done to reduce the influence of words e.g 'of', 'the' that could be used multiple times in a document but have no real meaning and importance in context.

$$IDF(t) = \log_e \frac{TD}{ND}$$

where TD = Total number of documents
and ND = Number of documents with term t in it

For most traditional applications the bag-of-words model was used. However, words that occurred frequently in the document were given significant weights in the Bag-OF-Words model and this affected the accuracy of these results. The use of TF-IDF was introduced to solve these problems of the typical Bag-Of-Words Model. The IDF of an infrequent term due to the log applied is high and the IDF for common words is low [13]. Three levels of TF-IDF were used for this analysis. They are the Word-level, N-Gram and Character Level TF-IDF. All these are implemented using the scikit-learn package of python.

Word2Vec Word embeddings are vector representations of words and word2Vec is one of the most popular methods of word embedding. The word2Vec method was first proposed in the work of Mikolov et al [15] as a method of feature extraction for Natural Language Processing (NLP). It is an improvement on the traditional methods of feature extraction. The construction of word embedding using neural networks and can be doe using either the Common Bag of Words or Skip Gram. According to [15], the Skip Gram method performs better for smaller data set. The word2Vec vectors applied in this experiment are trained on the words available in the data set.

3.4 Classification algorithms

A couple of classification algorithms were chosen for this experiment. These algorithms are briefly described below:

Support Vector Machines The Support Vector Machine is efficient for both classification and regression. It is known to give good results for classification. The classes are separated by a hyper plane found by the algorithm. The LinearSVC from scikit-learn was implemented for this analysis.

Naive Bayes The Naive Bayes is a powerful algorithm used for classifying data based on probabilities [7]. This algorithm is based on the Bayes theorem in statistics. The classification of the data is done using various probabilities. It is fast and scalable but is not without disadvantages as it assumes independence among predictors. It works well for small data sets and has been known to give great results. The Naive Bayes Classifier employed in all stages of the analysis are built using the scikit-learn package available in python.

Logistic Regression Logistic Regression is a popular classification algorithm. It belongs to the class of Generalized Linear Models. Its loss function is the sigmoid function which minimizes the results to be a value between 0 and 1. The Logistic Regression Model was implemented for this analysis using the function available in the scikit-learn package.

Convolutional Neural Networks (CNN) CNNs are very commonly used in text classification problems due to their success and great results. This has contributed to the decision to use CNN for this aspect of the experiment. In the first experiment with CNN, one CNN model is used on the features extracted using the Count Vectorizer. Another CNN model is also trained on the features extracted using Word2Vec. The results of these two models are plotted and compared.

4 Experiments and Results

4.1 Experiments

Figure 2 below is a pictorial representation of the processes involved in this experiment. The first step taken was the amalgamation of the two data sets obtained from Kaggle. Text pre-processing techniques were applied to the data as described above. The main body was an iterative process involving the application of one type of feature extraction technique accompanied with the building of the model and recording the results. Cross Fold validation was applied using the GridSearch function of python's sklearn for the final results for the supervised learning models. The results of these experiments are recorded and reviewed in the section that follows.

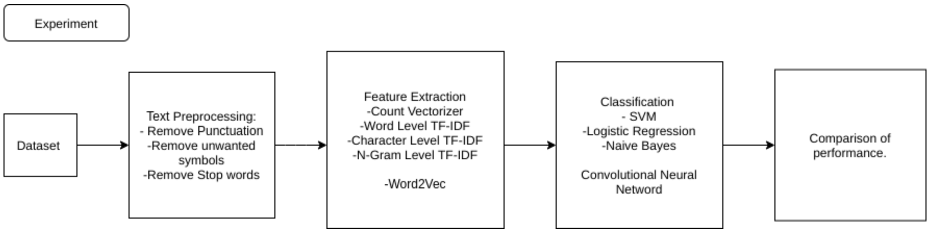

Fig. 2. Experiment Structure

4.2 Results

Table 3 contains the results of the experiments. The metrics of measurement for each algorithm and the accompanying scores are detailed in that table.

	F1 Score	Accuracy	ROC AUC Score
Count Vectorizer			
Linear SVM	0.94 for 0 0.92 for 1	0.93	0.927127083461732
Logistic Regression	0.90 for 0 0.88 for 1	0.89	0.891574615638756
Naive Bayes	0.87 for 0 0.85 for 1	0.86	0.862782631933632
Word Level TF-IDF			
Linear SVM	0.84 for 0 0.81 for 1	0.83	0.826282163974597
Logistic Regression	0.84 for 0 0.80 for 1	0.82	0.802490934661872
Naive Bayes	0.83 for 0 0.78 for 1	0.81	0.81771997863453
Character Level TF-IDF			
Linear SVM	0.82 for 0 0.78 for 1	0.8	0.800651976895386
Logistic Regression	0.81 for 0 0.76 for 1	0.79	0.7817870730821
Naive Bayes	0.77 for 0 0.70 for 1	0.74	0.736702613127708
N-Gram Level TF-IDF			
Linear SVM	0.75 for 0 0.49 for 1	0.66	0.638295324080576
Logistic Regression	0.75 for 0 0.49 for 1	0.66	0.63666163513879
Naive Bayes	0.75 for 0 0.49 for 1	0.66	0.637254498317267

Table 3. Table showing results of supervised learning

For the supervised learning techniques, the results to be examined are the results of the SVM, Naive Bayes and Logistic Regression when text features are extracted using the Count Vectorizer technique and different levels of the TF-IDF technique. These results are detailed in Table 3 above and can be seen graphically in 3 below.

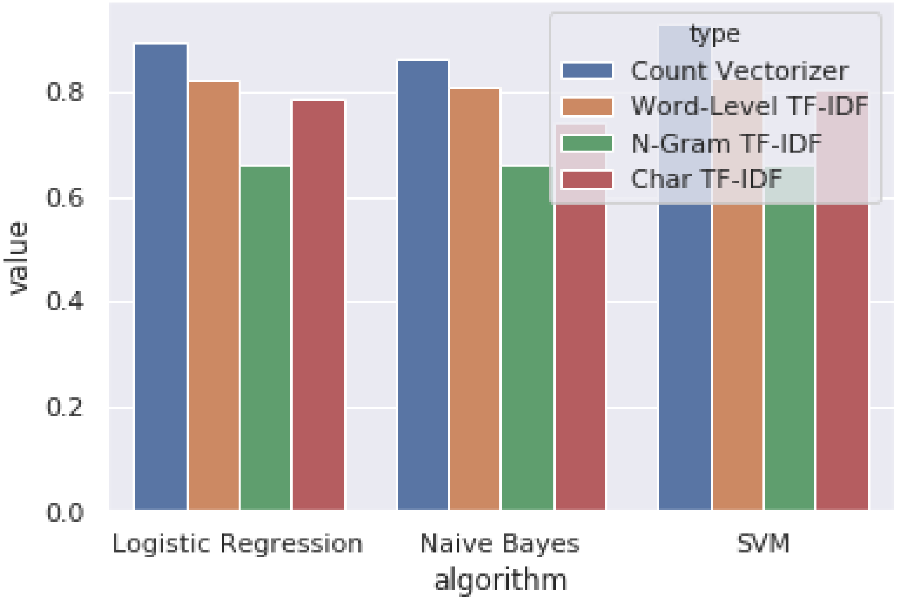

Fig. 3. Supervised Learning Results Diagram.

The bar plot that follows in Figure 3 is pictorial representation of the results of the analysis performed using the supervised learning methodologies. It is clear that the SVM outperforms both the Logistic Regression and the Naive Bayes in all the cases, except the N-Gram level TF-IDF where they all perform similarly poor. The figures that follow concentrate mainly on the results of the SVM algorithm due to this fact.

The results of the work done in our research shows that the use of Count Vectorizer and TF-IDF gives an accuracy of 93% when this feature set is fed to SVM. This is an improvement on the work of [14] as the methods used here provide a higher accuracy all round.

The results as seen above corroborate the known fact that Support Vector Machine usually outperforms other algorithms for classification purposes, depending on the measure of data available. This is clear in the measure of the accuracy as seen in the bar plots in Figure 3. The accuracy of the SVM with the Count Vectorizer is about 90%. The results for the n-gram level TF-IDF are the worst of all. This is similar to the Bag-Of-Words model and as such the bad results are expected.

Figure 4 below shows the training and validation accuracy scores for the model trained over 5 epochs. Unsurprisingly, the accuracy of training for the CNN is very close to 100%. The validation accuracy however is about 92.5%. The loss also reduces for both the training and validation data.

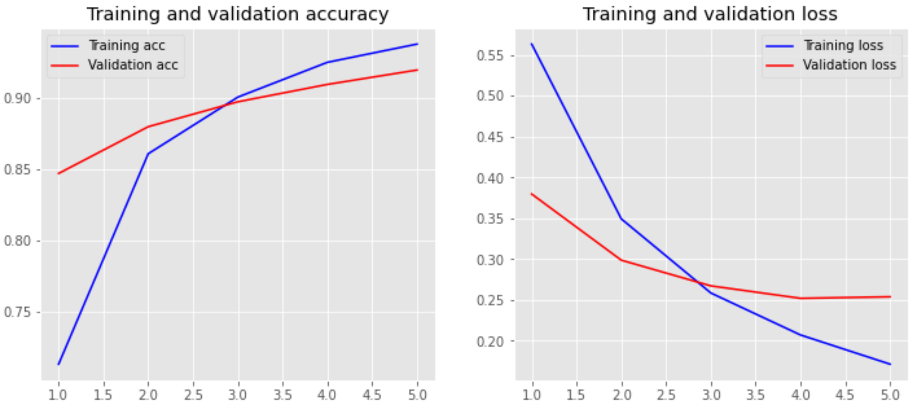

Fig. 4. CNN Results with features extracted using the Count Vectorizer

The training and validation accuracy score is higher when word embedding is applied to the CNN model. However, the Figure in 5 shows an interesting pattern for the validation loss when word2vec is applied with the CNN. The loss reduces and then begins to increase again for the data used in validation. This suggests that the model does not actually perform as expected when word embedding is applied. This is due to the fact that word2vec is a neural network based solution and these work optimally when the volume of data is substantial.

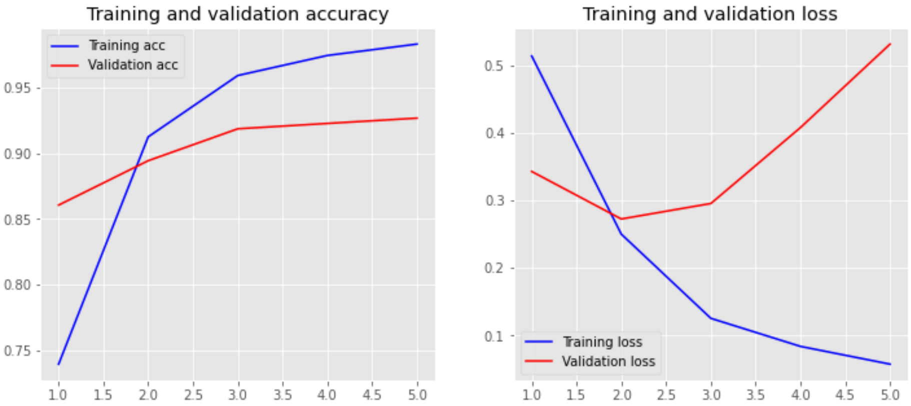

Fig. 5. CNN Results with features extracted using the Word Embeddings

5 Conclusion

Sarcasm can be found in a wide variety of areas. To be able to accurately detect sarcasm from different aspects and topics, a large enough data set will be needed. The accuracy of prediction as seen in this paper greatly depends on the mode of feature extraction. The use of N-grams and characters produced less desirable results than the use of words. The combination of the Count Vectorizer and the Term Frequency-Inverse Document Frequency for the supervised learning techniques gave the most satisfactory performance metrics.

This results of this research favor the supervised machine methods over the deep learning methods. Although this could be due to the limited amount of data available. This goes to show that the use of simple methods can produce great results, especially in the situation that the data available is limited and not suitable for more complicated deep learning techniques. The CNN performed satisfactorily also. The use of word embedding for this task showed great performance in terms of accuracy, both in the training and validation sets so does the use of the features extracted using the count vectorizer. The overall performance of both the CNN and supervised learning methods suggests that the use of Count Vectorizer is most appropriate for this task. As stated above, the volume of data available limits the strengths of the Word2Vec method, this the Count Vectorizer outperforms in this situation. This is due to the fact that simpler methods are always a better choice than more complicated methods provided that the results are similar or better with the simpler methods.

Sarcasm can also be multi-modal, contained in images and GIFs. This is especially available on social media like Twitter where tweets can be replied with GIFs and videos. Future research would involve an extension of this system to include multi-modal methods for sarcasm detection. Social media data for sarcasm detection will also contain emoticons (emojis) and analysis of those emojis could add more meaning to the text. A bigger data set will allow for the use of feature extraction techniques that require large data sets. This paper has not explored the use of word embedding for both supervised and deep learning techniques. More exploration into these methods of feature extraction available is warranted and would be a focus of future research in this area.

References

1. Liu, P., Chen, W., Ou, G., Wang, T.,Yang, D., Lei, K.: Sarcasm Detection in Social Media Based on Imbalanced Classification In: Web-Age Information Management, pp. 459-471. Cham, (2014). https://doi.org/10.1007/978-3-319-08010-9_49
2. Mehndiratta, P., Sachdeva, S., Soni, D. :Detection of Sarcasm in Text Data using Deep Convolutional Neural Networks In: Scientific International Journal for Parallel and Distributed Computing, vol. 18, September (2017) https://doi.org/10.12694/scpe.v18i3.1302
3. Zhibo, W. , Ma, L., and Zhang, Y. : A Hybrid Document Feature Extraction Method Using Latent Dirichlet Allocation and Word2Vec. In : IEEE First International Conference on Data Science in Cyberspace, June (2016). https://doi.org/10.1109/DSC.2016.110

4. Zhang, W., Xu, W., Chen G., Guo, J. : A Feature Extraction Method Based on Word Embedding for Word Similarity Computing. In: Communications in Computer and Information Science, vol. 496,pp. 160-167 January, (2014) https://doi.org/10.1007/978-3-662-45924-9_15
5. Lunando, E., Purwarianti, A. : Indonesian Social Media Sentiment Analysis With Sarcasm Detection. In: Int. Conf. Adv. Comput. Sci. Inf. Syst. ICACSIS, pp. 195-198. September (2013). https://doi.org/10.1109/ICACSIS.2013.6761575
6. 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, July (2010) https://doi.org/10.5555/1870568.1870582
7. Ahuja, R., Chug, A., Kohli, S., Gupta, S., Ahuja, P. : The Impact of Features Extraction on the Sentiment Analysis In: Procedia Computer Science, vol. 152, pp. 341-348. Elsevier, January (2019). https://doi.org/10.1016/j.procs.2019.05.008
8. Agarwal, A., Xie, B., Vovsha, I., Rambow, O., and Passonneau R.:Sentiment Analysis of Twitter Data In: Proceedings of the Workshop on Language in Social Media, pp. 30-38, June (2011). https://doi.org/10.5555/2021109.2021114
9. Fan, X., Li, X.,F. Du, F., Li, X., Wei, M.:Apply word vectors for sentiment analysis of APP reviews. In: 3rd International Conference on Systems and Informatics(ICSAI), pp. 1062-1066, (2016)
10. Fouad, M.M., Gharib, T.F., Mashat, A.S: Efficient Twitter Sentiment Analysis System with Feature Selection and Classifier Ensemble. In: The International Conference on Advanced Machine Learning Technologies and Applications (AMLTA2018), vol. 723, pp. 516-527, Springer, Cham January (2018). https://doi.org/10.1007/978-3-319-74690-6_51
11. Angulakshmi, G., Manicka Chezian, R.: Three level feature extraction for sentiment classification. In: International Journal of Innovative Research in Computer and Communication Engineering 2, vol. 8,pp. 5501-5507 (2014).
12. Prusa, Joseph, D., Taghi, M.K, and David J.D. :Impact of Feature Selection Techniques for Tweet Sentiment Classification. In: FLAIRS Conference 2015, pp. 299-304, April (2015)
13. Waykole, R.N, Thakare, A.D: A Review Of Feature Extraction Methods For Text Classification. In: International Journal of Advance Engineering and Research Development,vol. 5, April (2018).
14. Vaishvi, P. J.: Optimal Feature Extraction based Machine Learning Approach for Sarcasm Type Detection in News Headlines. In: International Journal of Computer Applications,vol. 177, March (2020).
15. Mikolov, T., Chen, K., Corrado, G., Dean, J.:Efficient Estimation of Word Representations in Vector Space.