Comparison of Supervised Machine Learning Models in COVID-19 Tweet Classification

Kopal Garg
Biomedical Engineering
University of Waterloo
Waterloo, Canada
kopal.garg@uwaterloo.ca

Abstract

Around March 2020, the outbreak of a global pandemic was declared by the World Health Organization. Naturally, the pandemic caused global outrage – people shared their opinions on public health responses along with their sentiments in form of fear, and panic fueled by incomplete and/or inaccurate information. The data analyzed in this paper was a collection of ~45,000 Tweets found in the Coronavirus Tweets NLP Dataset on Kaggle. A primary objective of this paper was to explore these behaviors at a time when the pandemic was starting to approach peak levels in some regions. This entails textual preprocessing, data visualization and analytics as well as a performance comparison of 4 supervised sentiment classification models: linear Support Vector Machines, Naïve Bayes, Decision Trees and Logistic Regression. The impact of feature extraction methods like Bag-of-Words, bigram Bag-of-Words and term frequency-inverse document frequency are also explored for each model. Out of all 12 experiments, the Logistic Regression model trained on the Bag-of-Words feature set had the best test accuracy of *81%*, closely followed by the linear Support Vector Machine classifier trained on the term frequency-inverse document frequency matrix with an accuracy of *80%*.

Keywords: COVID-19, Sentiment Analysis, Twitter, Textual Analytics, Linear Support Vector Machines, Naïve Bayes, Decision Trees, Logistic Regression, Bag-of-Words, TF-IDF, N-grams

Introduction

As the world faces the Coronavirus disease (COVID-19), Twitter has become a significant resource that helps relay important information to users, in real-time. The number of Tweets available encourage the study and development of sentiment analysis algorithms that aim to infer the attitudes of people towards a certain topic. As the global conversation continues around the spread of the pandemic, tens of millions of Tweets on this topic have been shared. Given the sensitive yet evolving nature of the pandemic, people continue to feel strongly about public health policies and announcements. They rightfully feel the need to debunk misleading or unverified claims and typically do so by sharing sentimental Tweets. Some of these Tweets are positive, while some can be negative.

In order to protect public conversation, Twitter is creating new rules surrounding behaviors that are considered acceptable on the platform [1]. It has a zero-tolerance policy for any attempts of abuse or malicious behaviors [1]. Given the volume of Tweets that are shared per second, it is largely impossible to manually identify those that are positive or negative and as expected during times of crises, negative emotions are dominant. Tweets that are excessively negative and could incite panic should ideally be flagged before they are widely circulated. In order to maintain the public's mental well-being, such Tweets should preferably be counterbalanced with strategic public health communication. The main motivation behind this paper is to identify Tweets that are excessively negative and may potentially lead to public outrage.

Existing solutions in the field of text sentiment analysis have not yet achieved a level of reliability adequate enough to be implemented. This paper performs a systematic comparison of multiple techniques in predicting whether a Tweet indicates a positive, negative or neutral sentiment. It focuses on exploratory data analysis, textual preprocessing, feature generation and sentiment classification methods. The sentiment classification methods train on feature vectors and identify a class associated with a data point. The dataset [2] used is a collection of COVID-19 related Tweets from March to April in 2020, reflecting varying sentiments. A systematic literature review of many research efforts in this area will also been presented. It addresses questions regarding the current state of research, most effective techniques and the most significant limitations of the current methods.

Background Review

Methods have been implemented in the past to explore the application of sentiment analysis in examining Twitter data on various topics. The reviewed studies were classified based on methods used for preprocessing & feature extraction and ML models for sentiment classification.

Go et al. [3] and Read et al. [4] used emoticons as "noisy labels" to annotate Tweets and examined Naïve Bayes (NB), and Support Vector Machines (SVM), using bigrams, unigrams and part-of-speech tags (POS) as features. NB with bigrams as features achieved an accuracy of *82.7%*. Pak and Paroubek et al. [5] used a similar emoticon-based labelling technique, but tackled the problem as a multi-class classification tasks by including neutral sentiments. They compared the performance of SVM and Multinomial NB (MNB) using unigrams, bigrams, POS tags and n-grams. A combination of MNB with n-grams and POS tags achieved the highest accuracy. Samuel et al. [6] compared the effectiveness of Naïve Bayes (NB) and Logistic Regression (LR) in classifying tweets of varying lengths. They formed training sets using Bag-of-Words (BoW), hashing and term frequency-inverse document frequency and proposed ways of concatenating the feature extraction techniques. They reported a *91%* classification accuracy of NB for short tweets, a *74%* classification accuracy of LR for shorter tweets and relatively weaker accuracies of both methods for longer tweets.

Barbosa and Feng et al. [7] used POS tags and syntax features like retweets, hashtags, URLs, and emoticons to train their SVM mode. They also normalized the frequency of each feature by the number of tokes in the tweet. They achieved an *82%* accuracy with their SVM classifier. Pang et al. [8] used a unigram and bigram BoW framework to train SVM models from a collection of movie reviews annotated as positive or negative. The best accuracy achieved was *82.9%,* using an SVM trained on unigram BoW features. Through their study, Mohammad and Bravo-Marquez et al. [9] concluded that hashtags did not contain relevant information for sentiment classification. Neelakandan and Paulraj et al. [10] conducted extensive preprocessing before implementing a Gradient Boosted Decision Tree (DT) classifier. They removed stop-words, and hashtags, but used emoticons, and punctuation as features. Similarly, Murthy et al. [11], removed words that weren't present in the English dictionary while retaining specific Twitter-associated elements like usernames, mentions and emoticons. Alharbi et al. [13] added behavioral information to their analysis by including features like total number of tweets, number of followers, account verification status, etc. The results showed improvement in accuracy compared to more traditional methods described earlier.

Wehrmann et al [12] described the use of deep learning methods and word embeddings or mappings from words to a n-dimensional space. Words with similar meanings are close to one another in this n-dimensional space. They also addressed issues related to multilingual sentiment analysis, pointing to the fact that the n-dimensional space would become unwieldy if many languages are included. They proposed the use of character embeddings where each character is one-hot encoded. They trained a convolutional neural network with a ReLu activation layer that learns the relation between words and characters, and is able to identify misspelled words.

Application/Dataset

The data is obtained from the "Coronavirus Tweets NLP - Text Classification" dataset on Kaggle [2]. It contains 2 .csv files - *Corona_NLP_test.csv*, *Corona_NLP_train.csv*. Columns include geographic locations, dates on which the Tweets were published, the original/raw Tweets and sentiment labels for a combined ~45,000 Tweets from March to April in 2020. The usernames are coded to avoid privacy concerns. Table 1 contains a high-level summary of the training data.

Table 1. *Training Data Summary*

Attribute	*Summary*	*Attribute*	*Summary*
# unique tweets	41,156	time frame	03-16-20 to 04-14-20
# unique locations	12,221	# unique sentiments	5

Visualization and analytical techniques were implemented to conduct a preliminary exploration of the data. Figure 1.A. represents the frequency at which Tweets were published from March 16 to April 14, 2020. Figures 1.B. and E. show word clouds consisting of the top-50 frequent unigrams and bigrams found in positive and negative Tweets. In Figure 1.B., words, like, "thank", "good", "help", etc. are seen, while in Figure 1.E. words and phrases with negative connotations, like, "panic", "crisis", and "oil price", etc. are frequently observed. Figure 1.C. represents a circular bar plot showing the top-5 locations from which Tweets were published. Figure 1.D. represents the sentiment distribution in Tweets. It is observed that the number of Tweets with a positive polarity is fairly larger in the training dataset.

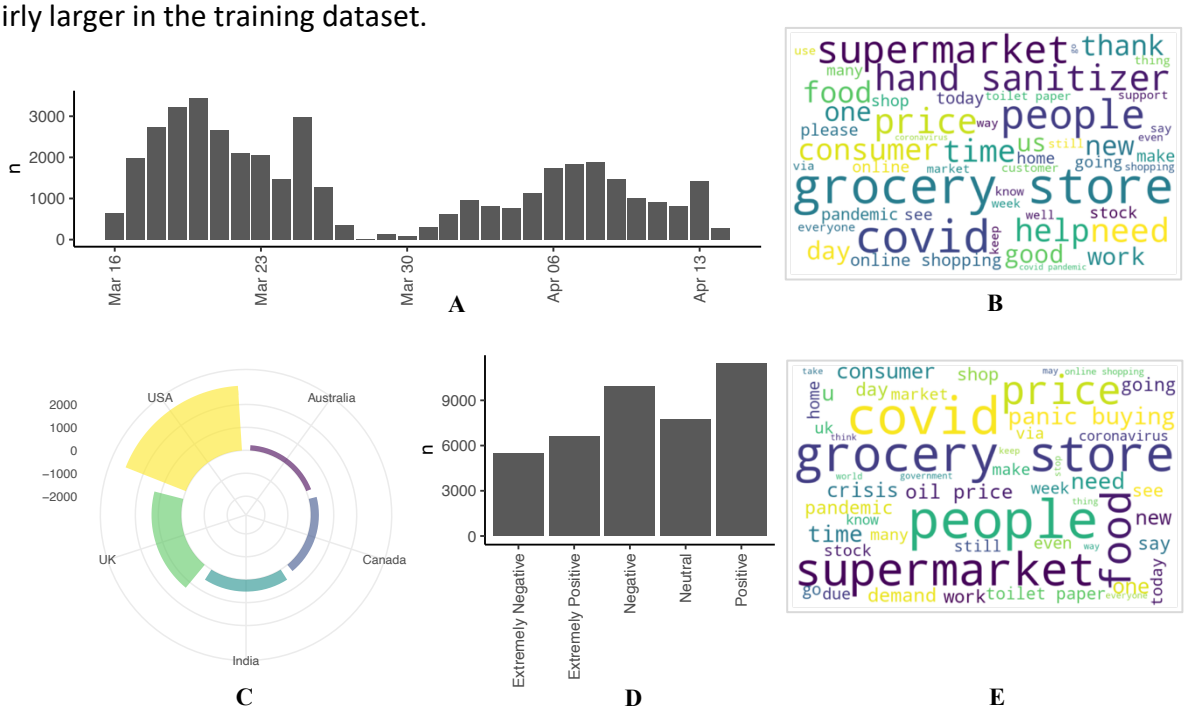

Figure 1. ***A.*** *Tweet Frequency.* ***B.*** *Word Cloud – Top-50 Positive Words.* ***C.*** *Top-5 Locations.* ***D.*** *Sentiment Distribution.* ***E.*** *Word Cloud – Top-50 Negative Words.*

Proposed Scheme/Algorithm

Figure 2 depicts the primary components of the proposed sentiment prediction method. A preprocessing pipeline was written to create a data set ready for further analysis. This involved eliminating some textual and non-textual variables in the data that don't necessarily contribute to sentiment analysis. Twitter handles, URLs, non-ASCII or special characters, punctuation and numeric values were filtered out and the Tweets were lower cased. Using a corpus of stop-words, including articles, prepositions, and conjunctions, from the *Natural Language Toolkit* (*NLTK*) suite, words with little lexical content (e.g. *the*, *a*, *also*, *from,* etc.) were filtered out. Using the *emot* package, all emojis and emoticons were replaced with their Unicode CLDR short-name (e.g. ☺ replaced by '*happy face smiley')*. The *pyspellchecker* package was used to return the most probable results for misspelled words. Segments of text were further processed with tokenization and lemmatization, both of which were performed using packages in the *NLTK* suite. Tokenization converted text to analysis relevant word tokens while lemmatization transformed words to a simpler form, returning the word's lemma – a canonical form of all its inflectional forms (e.g. *go* represents its inflected forms of *goes*, *going*, *went*, *gone*).

Input	'Coronavirus Australia: Woolworths to give elderly disabled dedicated shopping hours amid outbreak https://t.co/bInCA9Vp8P'
Output	['coronavirus', 'australia', 'woolworths', 'give', 'elder', 'disable', 'dedicate', 'shop', 'hour', 'amid', 'outbreak']

The target variable was encoded with values between 0 and 2 (negative = 0, neutral = 1, positive = 2). Three techniques for encoding text data into numerical vectors were implemented to extract suitable features from processed texts. Bag-of-Words (BoW) is a vectorization method to covert variable-length texts into fixed-length vectors, without considering the semantic relation between words. Consider the following two sentences, with differing sentiments:

Input 1	'The dish was affordable yet tasty'
Output 2	['dish, 'affordable', 'yet', 'tasty']
Input 2	'The dish was affordable, but not tasty'
Output 2	['dish, 'affordable', 'but', 'tasty']

Table 2. *Example Bag-of-Words Representation Matrix*

Unigrams	affordable	but	dish	tasty	yet	…
Output 1	1	0	1	1	1	…
Output 2	1	1	1	1	0	…

After converting processed outputs into their respective vector representations (Table 2), the Euclidean distance is computed to check for similarity. Using just unigrams or words as features, a pairwise comparison of both sentences will result in a small Euclidean distance, although they have differing sentiments. Hence unigram features may be insufficient in discriminating the two sentiment classes. Typically, BoW may result is a very sparse representation. For a large dataset with the vocabulary of a few thousand words, preprocessing text before employing this

technique can therefore be useful. Similar to BoW, Bigram BoW (B-BoW) represents a text document as a weakly ordered collection of contiguous sequences, but of two items. It allows for the preservation of more word locality information. Using bigrams, the two sentences will generate different sets of features (Table 3), and varying subjectivity scores may be assigned.

Table 3. *Example Bigram Bag-of-Words Representation Matrix*

Bigrams	dish affordable	yet tasty	but tasty	…
Output 1	1	1	0	…
Output 2	1	0	1	…

For sentences of different polarity in this dataset, it is therefore hypothesized that bigrams can substantially raise the quality of the feature set. Both methods were implemented using *scikit-learn*'s CountVectorizer where 'ngram_range' was set to (1,1) for BoW and (2,2) for B-BoW. Term frequency-inverse document frequency (TF-IDF) is frequency-based embedding method that measures the importance of a word in a given document. $TF(t,d)$ is the frequency of term t in document d. The total number of words in Input 1 is 6, so the $TF(t,d)$ for 'affordable' is 1/6. $IDF(t) = \log\frac{|D|}{|d:t\in d|}$, where $|D|$ is the number of documents in the corpus, and $|d: t \in d|$ is the number of documents where t appears ($TF(t,d)\neq 0$). The $IDF(t)$ for 'yet' is $\log(2)$. Similarly, the $TF-IDF = TF(t,d) * IDF(t)$ scores for some unigrams in both sentences are below.

Table 4. *Example TF-IDF Scores Matrix*

Unigrams	affordable	but	yet	…
Input 1	0.00	0.00	0.05	…
Input 2	0.00	0.04	0.00	…

A high weight in TF-IDF matrix is reached by a high term frequency in the given document and a low document frequency of the term in the corpus. This was implemented using *scikit-learn's* TfidfVectorizer function where argument 'min_df' (cut-off threshold for low frequency for n-grams) was set to 0.2, 'max_df' (cut-off threshold for high frequency n-grams) to 0.8, 'use_idf' to True and 'ngram_range' to (1,1) to include unigrams. Based on findings from the Background Review, a comparison of the four most frequently used ML models was conducted to find one with the highest performance in predicting the sentiment polarity (positive, neutral and negative) of COVID-19 Tweets. These models include support vector machine (SVM), decision tree (DT), multinomial Naïve Bayes (MNB) and Logistic Regression (LR). For many experiments, 10-fold cross-validation (CV) was employed for hyperparameter tuning. Since the feature sets are large, a leave-one-out CV was deemed infeasible. The unseen portion of the training set and the test set were used to compute the validation and testing error. Standard evaluation parameters were used to find the models' accuracy, precision, recall and F_1 score.

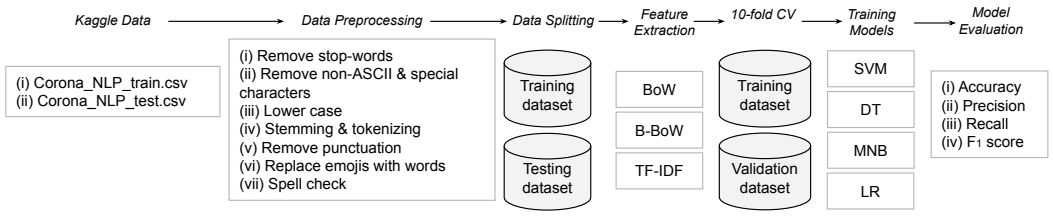

Figure 2. *Proposed Sentiment Prediction Method Overview*

Experiments and Results

In this section, the experimental evaluation of the proposed methodology is presented. Figure 3 shows the number of words that were eliminated with each filtering step in the preprocessing pipeline, with an end result of 13,297 unigrams and 251,384 bigrams in the training set.

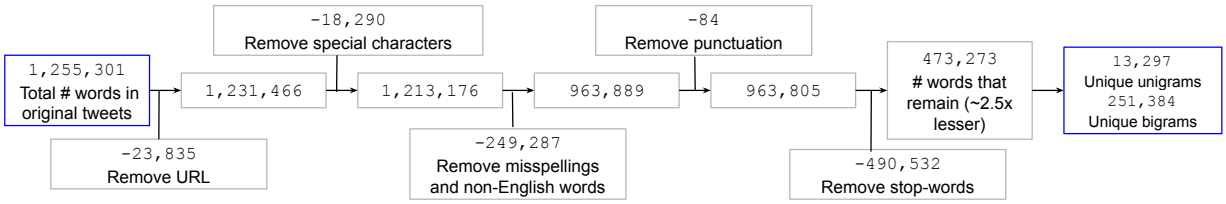

Figure 3. *Change in Total # of Words in the Training Set Over the Preprocessing Pipeline*

A MNB classifier was trained using the *sklearn.naive_bayes.MultinomialNB* function. User defined parameters to fit this estimator included a training set in the sparse matrix format, a target variable and an alpha value. The *sklearn.model_selection.cross_val_score* function was used to conduct a 10-fold CV and simultaneously evaluate the performance of different settings for alpha. Figure 4. A-C, show the alpha values that obtained the highest CV score for all three models (BoW: 0.68, B-BoW: 0.55, TF-IDF: 0.65).

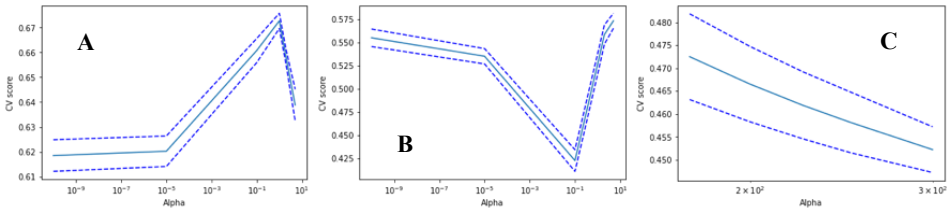

Figure 4. *Multinomial NB – CV Scores for Alpha Values.* ***A.*** *BoW.* ***B.*** *B-BoW.* ***C.*** *TF-IDF.*

A DT classifier was trained using the *sklearn.tree.DecisionTreeClassifier* function. User defined parameters to fit this estimator included the maximum depth of the tree, which was left at its default value of 'None', so that nodes will continue to expand until all leaves are pure or until 2 samples are reached. An exhaustive search was performed over the criterion parameter ("Entropy", "Gini") using *GridSearchCV*, and higher accuracy for all three models was obtained using Entropy. A one-vs-all SVM with a linear kernel was trained using the *sklearn.svm.SVC* function. Only the linear kernel was experimented with as linear SVMs are known to be useful for text classification. They acknowledge particular properties of the text classification problem like a high-dimensional feature space, sparse instance vectors and dense concept vectors [14]. The C regularization parameter was tuned using *GridSearchCV* and the best value was used to fit each model. A LR classifier was trained using the *sklearn.linear_model.LogisticRegression* function. By default, the one-vs-all scheme was implemented. Figure 5. A-C, show the results of a grid search that was conducted to tune C, which is a value representing the inverse of the regularization strength. By default, the LM-BFGS algorithm was used to solve the optimization problem.

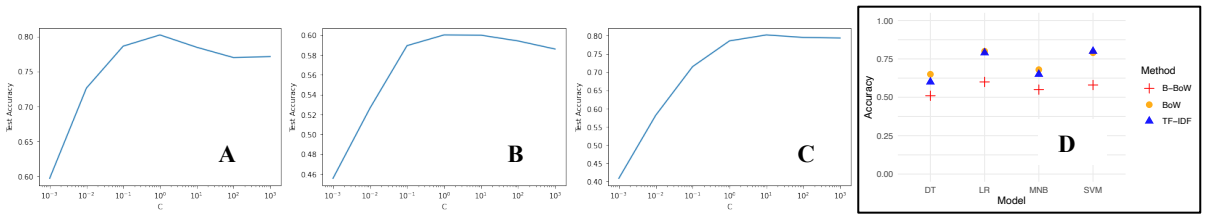

Figure 5. *Linear Regression – Accuracy for C Values.* ***A.*** *BoW.* ***B.*** *B-BoW.* ***C.*** *TF-IDF.* ***D.*** *Model Accuracy Performance for Three Feature Extraction Methods.*

Table 5. *Standard Evaluation Parameters on Testing Data for all Experiments*

	Accuracy				Class	Precision				Recall				F_1 Score			
	MNB	**DT**	**SVM**	**LR**		**MNB**	**DT**	**SVM**	**LR**	**MNB**	**DT**	**SVM**	**LR**	**MNB**	**DT**	**SVM**	**LR**
BoW	0.68	0.65	0.79	0.80	0	0.75	0.69	0.81	0.83	0.73	0.61	0.80	0.80	0.74	0.64	0.80	0.82
					1	0.61	0.55	0.68	0.68	0.03	0.61	0.72	0.71	0.06	0.57	0.70	0.69
					2	0.61	0.66	0.82	0.82	0.85	0.72	0.82	0.85	0.72	0.69	0.82	0.83
B–BoW	0.55	0.51	0.58	0.60	0	0.65	0.60	0.66	0.66	0.52	0.49	0.56	0.59	0.57	0.54	0.61	0.63
					1	0.71	0.29	0.36	0.44	0.01	0.33	0.52	0.35	0.02	0.31	0.43	0.39
					2	0.50	0.53	0.64	0.60	0.80	0.60	0.63	0.71	0.62	0.56	0.63	0.64
TF–IDF	0.65	0.60	0.81	0.79	0	0.74	0.64	0.82	0.81	0.69	0.58	0.81	0.79	0.71	0.61	0.81	0.80
					1	0.73	0.44	0.72	0.72	0.05	0.54	0.67	0.60	0.09	0.49	0.70	0.66
					2	0.59	0.63	0.81	0.78	0.85	0.64	0.84	0.85	0.70	0.63	0.82	0.82

After training all learning models, their performance was evaluated on accuracy, precision, recall and F_1 scores that are presented in Table 5. The features, such as count vector of unigrams, and bigrams, were experimented separately. Here, 0, 1 and 2 represent positive, neutral and negative sentiments, respectively. Figure 5. D., shows a comparison of the classification accuracies for all three feature extraction methods, and all four classifiers. In general, the mean classification accuracy of *BoW > TF-IDF > B-BoW*, and *LR > SVM > MNB > DT*. The best performance was achieved by linear models like the SVM (with a linear kernel) trained on TF-IDF vector representations (accuracy of 81%, mean precision of *78%*, mean recall of *77%*, and mean F_1 score of *78%*). This was closely followed by the LR classifier trained on the BoW feature matrix (accuracy of *80%,* mean precision of *78%*, mean recall of *79%*, and mean F_1 score of *78%*). Linear models are generally suitable for large datasets and are more computationally efficient. However, it was observed that SVM's performance decreased when the feature size increased. Consequently, SVMs were superior for relatively small datasets like BoW and TF-IDF. The worst performance was achieved by the DT classifier trained on the B-BoW feature matrix (accuracy of *51%,* mean precision of *47%,* mean recall of 47%, and mean F_1 score of *47%*), and the MNB classifier also trained on the B-BoW feature matrix (accuracy of *55%*, mean precision of *62%*, mean recall of 44%, and mean F_1 score of *40%*). It is observed that most classifiers perform slightly better for BoW than TF-IDF and B-BoW on the testing set. It is likely that since B-BoW has a larger and sparser feature matrix, the models overfit to the training data and therefore underperform on the testing data. In general, more misclassifications are made for the 'neutral' class. A possible reason behind this observation is that the 'neutral' class has a lower frequency of occurrence in the training set as compared to other classes. Deep learning models like CNN and LSTM seem like a promising alternative to traditional ML models described in this report. They perform adequately on large datasets with a large number of features, and complex classification tasks like the one described in this paper. Since the models were implemented on non-standardized datasets, their accuracy may prove to be unreliable on other datasets. In order to make more generalizable results, standardized datasets will be used in the future and benchmarks in different domains will be compared against to get a more reliable measure of accuracy.

References

[1]"Coronavirus: Staying safe and informed on Twitter", *Blog.twitter.com*, 2021. [Online]. Available: https://blog.twitter.com/en_us/topics/company/2020/covid-19.html. [Accessed: 27- Apr- 2021].

[2]"Coronavirus tweets NLP - Text Classification", *Kaggle.com*, 2021. [Online]. Available: https://www.kaggle.com/datatattle/covid-19-nlp-text-classification. [Accessed: 27- Apr- 2021].

[3]A. Go, R. Bhayani and L. Huang, "Twitter Sentiment Classification using Distant Supervision", *CS224N project report, Stanford*, vol. 1, no. 12, pp. 1-6, 2009. Available: https://d1wqtxts1xzle7.cloudfront.net/34632156/Twitter_Sentiment_Classification_using_Distant_Supervision.pdf.[Accessed 27 April 2021].

[4]J. Read, "Using Emoticons to Reduce Dependency in Machine Learning Techniques for Sentiment Classification", *Proceedings of the ACL Student Research Workshop*, pp. 43–48, 2005. Available: https://www.aclweb.org/anthology/P05-2008/. [Accessed 27 April 2021].

[5]A. Pak and P. Paroubek, "Twitter as a Corpus for Sentiment Analysis and Opinion Mining", *European Language Resources Association (ELRA)*, 2010. Available: http://www.lrec-conf.org/proceedings/lrec2010/pdf/385_Paper.pdf. [Accessed 27 April 2021].

[6]J. Samuel, G. Ali, M. Rahman, E. Esawi and Y. Samuel, "COVID-19 Public Sentiment Insights and Machine Learning for Tweets Classification", *SSRN Electronic Journal*, 2020. Available: 10.2139/ssrn.3584990.

[7]L. Barbosa and J. Feng, "Robust Sentiment Detection on Twitter from Biased and Noisy Data", 2010. Available: https://www.aclweb.org/anthology/C10-2005.pdf. [Accessed 27 April 2021].

[8]B. Pang and L. Lee, "A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts", vol. 42-04, pp. 271–278, 2014. Available: https://www.aclweb.org/anthology/P04-1035/. [Accessed 27 April 2021].

[9]S. Mohammad and F. Bravo-Marquez, "Emotion Intensities in Tweets", 2017. Available: https://arxiv.org/pdf/1708.03696.pdf. [Accessed 27 April 2021].

[10]S. Neelakandan and D. Paulraj, "A gradient boosted decision tree-based sentiment classification of twitter data", *International Journal of Wavelets, Multiresolution and Information Processing*, vol. 18, no. 04, p. 2050027, 2020. Available: 10.1142/s0219691320500277.

[11]J. Murthy, G. Siddesh and K. Srinivasa, "TwitSenti: A Real-Time Twitter Sentiment Analysis and Visualization Framework", *Journal of Information & Knowledge Management*, vol. 18, no. 02, p. 1950013, 2019. Available: 10.1142/s0219649219500138.

[12]J. Wehrmann and W. Becker, "A character-based convolutional neural network for language-agnostic Twitter sentiment analysis", *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017. Available: 10.1109/IJCNN.2017.7966145 [Accessed 27 April 2021].

[13]A. Alharbi and E. de Doncker, "Twitter sentiment analysis with a deep neural network: An enhanced approach using user behavioral information", *Cognitive Systems Research*, vol. 54, pp. 50-61, 2019. Available: 10.1016/j.cogsys.2018.10.001.

[14]T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features", *Universit□at Dortmund*, 2005. Available: https://link.springer.com/chapter/10.1007/BFb0026683. [Accessed 27 April 2021].