Sentiment Analysis of Covid-19 Vaccine Tweets

P7: Mohit Soni, Anagha Patil, Aditi Salunkhe, Sukhad Joshi
Department of Computer Science, North Carolina State University
Raleigh, NC 27695
{msoni, apatil28, acsalunk, sjoshi32}@ncsu.edu

1 Background and Introduction

The novel coronavirus disease (COVID-19) outbreak disrupted the daily lives of billions of people worldwide, causing unfathomable losses. It is critical to take precautions such as wearing masks, washing your hands frequently, and exercising caution when sharing personal information.
These, however, can only reduce coronavirus transmission, not eliminate it. In this case, immunization was the only option for combating the coronavirus most effectively and likely completely.
Vaccinations are the only long-term solution to this epidemic, assuming that the majority of the population receives immunizations. As a result, since the outbreak of the pandemic, organizations and nations all over the world have begun developing vaccines and conducting clinical trials. There were no fatalities in any of the trials, and the vaccine success rates were 94% on average.
Even with this advancement, people lacked knowledge about vaccines and were skeptical of their safety. People were concerned that the vaccination might cause long-term chronic illnesses because it had not been thoroughly studied. Another factor could be the spread of false information about COVID-19 on social media, which could incite those who are on the fence or have doubts about the vaccination to reject it. As a result of examining the conversations on social media platforms, early identification of people's views on vaccines and prompt reaction is advantageous to vaccine promotion.
We used a few research papers for our project. One of them was "Sentiment analysis and topic modeling for COVID-19 vaccine discussion," [10] in which they analyzed the topics and sentiments of COVID-19 vaccine-related Twitter discussions. They used two robust text mining techniques, Latent Dirichlet Allocation (LDA) and Valence Aware Dictionary and Sentiment Reasoner (VADER), to extract the hidden information buried in noisy social media discussions, and conducted a thorough analysis of COVID-19-related Twitter discussions. They discovered that discussions are mostly positive, and the dominant sentiment of trust indicates a high acceptance of the COVID-19 vaccine. The other paper that we referred to, "Deep Learning-Based Sentiment Analysis of COVID-19 Vaccination Responses from Twitter Data".[1] VADER calculates a probability value of 1 based on the valence values of each word in the lexicon. The compound score is the most commonly used measure for sentiment analysis; it is a float value in the range [-1,+1], whose index is determined by adding the values of each word in the lexicon, adapted according to rules, and then standardized to its range. We used a variety of methods and compared the results for our project.

1.0.1 Logistic Regression

Logistic Regression was one of the methods (LR). We discovered a paper that employed an RBF kernel SVM and Logistics Regression (LR) while utilizing unigram and bigram features (BoW). Using the BoW model, researchers reported cutting-edge performance for Twitter sentiment analysis. They also tried combining the unigram, bigram, word sentiment polarity, twitter-specific features, and word vector features with an RBF kernel SVM and Logistics Regression classifier.

1.0.2 BERT

One of the papers implemented the Bi-directional Encoding Representation for a Transformer (BERT) model [7] [2]for emotion classification. The meaning of a word in a sentence is determined by the words that surround it. To handle word dependencies, they implemented a BERT model that feeds all input at once. The researchers used a train-test split to divide the collected data into training and testing sets, and then converted the training set into respective torch tensors for the model. They then defined the batch size to generate tensors and iterators to fine-tune the BERT model. The BERT model was then trained using the model parameters and its accuracy was validated.

1.0.3 LinearSVC

SVMs are one of the most popular machine learning methods for classification of linear problems. They try to find a set of hyperplanes that divide the space into dimensions that represent classes. These hyperplanes are chosen in such a way that the distance from the nearest data point of each class is minimized. The Linear SVC [5] is the simplest and fastest SVM algorithm that assumes a linear separation of classes. The researchers used SVM classifiers for stock price forecasting with sentiment indicators and sentiment analysis of restaurant reviews, respectively.

2 Proposed method

2.1 Data pre-processing

To understand the data better and decide on the pre-processing strategies, we carried out a holistic statistical analysis of the data. This includes understanding the data types of the columns present in the data and calculating the mean, and standard deviation of the numeric columns to understand the variation in the data. We also checked the data for missing values and null values. In our experiment we were mostly concerned about using tweet and sentiment columns, we decided to drop the records which had missing any of those values. We checked the unique values to understand the duplicity in the data. We also checked the frequent items of each item. This is because twitter data might have tweets generated by automated bots [6] which can bias the sentiment analysis because of its large volume.

While cleaning the data we adopted the strategy of replacing emojis (emoticons) with their literal meanings. One of the most popular strategies to handle emoticons is to remove them completely from the text. However, with Twitter data where users use emoticons very heavily, we used Unicode Consortium's emoji code repository to replace emojis with appropriate meanings.

Our approach also suggests that Twitter handles, URLs, and hashtags should be removed from the text for sentiment analysis. In most cases, Twitter handles and URLs contribute nothing to the very minimum towards deciding the sentiment analysis unless a supplementary analysis is carried out on the mentioned users and web links. The hashtags might indicate the trend and sentiment of the tweet however due to the inconsistent format of handles we recommend dropping them.

Moreover, our approach also converts the text to lower-case to make data more useful for case-sensitive algorithms. We also remove stopwords from the text to remove words unnecessary for the sentiment analysis.

We propose using three separate methods for representing the tweets. We use count vectorizer to represent the frequency of each word in the document. We use TF-IDF to represent the importance of each word in the document. We have also applied our algorithms with words represented using a pre-trained word embeddings such as Word2Vec. Word embeddings are the representation of words trained on large corpus of the data and gives closer representation to that of real-world meaning.

2.2 Algorithms

In the project we have used six different algorithms to analyze the sentiments of the tweets. Namely, we have used multinomial logistic regression, linear SVC, Naive Bayes, Random Forest, BERT, and XLNet.

Multinomial logistic regression: Multinomial logistic regression is the algorithm that generalizes the classification problem where output can have more than one possible values. It is the easiest algorithm from the implementation point of view. Since it is competitive in terms of CPU and memory usage [4], it can provide baseline results.

Linear SVC : The support vector classification is one of the most popular algorithm for text classification problem. More specifically, we used SVC with linear kernel as due to numerous reasons such as seperable features, faster, and less parameters for training [3].

Random forest classifier: Random forest classifier works on decision trees specifically works well on unbalanced text classification [8]. The simplicity of the algorithm is one of the reason why we used the random forest classifier.

Naive Bayes classifier: Naive bayes classifier uses Baye's theorem for classifying the unknown tweets into positive, negative and neutral sentiments. Naive bayes considers all features to be independent of each other in order to predict the dependent variable.

BERT: BERT makes use of a Transformer that learns contextual relations between words in a sentence/text. The transformer includes 2 separate mechanisms: an encoder that reads the text input and a decoder that generates a prediction for any given task. BERT makes use of only the encoder as its goal is to generate a language model. In contrast to state-of-the-art models, the Transformer encoder reads the entire sentence at once as it is bidirectional and thus more accurate. The bidirectional characteristic allows the model to learn all surroundings (right and left of the word) of words to better understand the context. [2]

XLNet: XLNet is an enhanced version of BERT like pre-trained model. It outperforms BERT on some NLP tasks including Question Answering, Text Classification, and others. It uses Transformer XL as a feature extracting architecture, to give a deeper understand of the language context. [9]

3 Plan and Experiment

We have selected python language for data pre-processing and model training because of support of a wide range of libraries. We have used pandas and numpy for data manipulation. Numpy is used to perform a wide variety of mathematical operations on arrays. We have used matplotlib and seaborn for creating static, animated, and interactive visualizations in Python. The Re (regular expression) library helped in removing hashtags, emojis and other special characters. We have also used word cloud for visualizing unstructured text data and getting insights on patterns. We have used nltk toolkit which is a library for tokenizing, parse tree visualization, etc. NLTK also supports the VADER algorithm which we have used.

3.1 Dataset

The dataset "COVID-19 All Vaccines Tweets" was chosen in this research where the data of almost all the renowned vaccines, such as Pfizer, Oxford, Moderna, Covaxin, etc. are available. All Vaccines Tweets data set available on the Kaggle. This dataset consists of a total 228207 records with 16 columns. Each record in this dataset is associated with a tweet related to Covid-19 vaccines. It has information about the user who has tweeted, the date and time of the tweet, and the actual contents of the tweet. It also has the data about the number of retweets and endorsements received by the tweet. The detailed description about each of the column is as follows -

Column Name	Description
user_id	ID assigned to each user
user_name	Twitter handle of the user
user_location	Location of the user as recorded by the platform.
user_description	Description added by the user on his Twitter profile.
user_followers	Number of other twitter users that follow the user
user_created	Timestamp of the user creation
user_friends	Number of other users that the user follows
user_favorites	Number of tweets liked by the user
user_verified	A boolean value to show if a user is verified or not
date	Timestamp of the tweet
text	The raw content of the tweet
hashtags	List of hashtags extracted from the tweet
sources	Specification of the device used to tweet
retweets	Number of retweets received by the tweet
favorites	Number of likes received by the tweet
is_retweet	The boolean to show if the tweet is a retweet

The data is stored as a csv file and has a size of total 90 MB.

3.2 Computation

For running the code we have used Google Collab. Google Collab allowed all of our group members to collaborate on the code and allows us to run code on GPUs.

3.3 Exploration of dataset

Steps involved in exploration are -

[1] Checking the dataset shape: Dataframe shape is calculated using .shape() function of pandas.

[2] Checking the dataset info: Information about the data frame is calculated using the .info() function of pandas.

[3] Describing the data: Basic details about the dataset like mean, median, $25\%, 50\%, 75\%$, etc are calculated using .describe() function of pandas.

[4] Checking for Missing values: We have used pandas' built-in .isnull() function and our own missing values function to calculate the percentage of missing values in each column.

[5] Checking for unique values in each column: We are implementing a function to check the unique values in each column.

[6] Checking for most frequent values in each column: We created a function to check the most frequent item, their frequency and their percentage from total.

[7] Count plot: We created a count plot function to plot each column with its count values with percentage.

[8] Checking prevalent text using word-Cloud: Created a function to check the word-cloud of whole text in the dataset.

3.4 Data Cleaning

Most of the data cleaning process is done using regular expression (re) library of python.

[1] Mapping of emoji with their meanings: Created a function to replace the emoji with their meaning in the dataset.

[2] Removing Twitter handles: A Twitter handle is the username that appears at the end of your unique Twitter URL. Removing those handles is a must process in data-processsing.

[3] Removing URLs: Removing the URL from the text is another step in data cleaning process.

[4] Removing hashtags: Hastags are removed by applying lambda function using regular expression.

[5] Normalizing case folding: Converting whole text into lowercase for better analysis.

[6] Punctuation - replacing them with spaces: A simple regular expression is written to remove the punctuations from the text.

[7] Removing numbers and special characters: A lambda function is written using regular expression to remove the numbers and special characters.

[8] Removing stopwords: The words which are generally filtered out before processing a natural language are called stop words. These are actually the most common words in any language (like articles, prepositions, pronouns, conjunctions, etc) and does not add much information to the text. Examples of a few stop words in English are "the", "a", "an", "so", "what". Stop words are available in abundance in any human language. By removing these words, we remove the low-level information from our text in order to give more focus to the important information.

[9] Lemmatization: Lemmatization is the grouping together of different forms of the same word. So, we try to remove those words with the same meanings.

[10] Remove single characters and double space removal: After each cleaning process, we would be left with spaces and single characters, so removing those as well.

3.5 Visualization

WordCloud: It allows to creation cloud shaped visuals according to the frequency of the words in the relevant text. For this, we need to specify all the texts as a single text. So, we have to make a single line to all elements of text column.

3.6 Hypothesis

As we worked on a specific twitter dataset which had COVID-19 vaccine reviews captured only between a specific timeframe(when the vaccines were just introduced and COVID-19 outrage was still on its peak), we are interested to see how effectively our implemented system predicts the tweets that were collected towards the end of the pandemic. This is because, when the people were in the middle of COVID-19 outrage, when the vaccines were just introduced, they were anxious, nervous, and skeptical about the vaccines as there wasn't enough data to prove the authenticity of the vaccines. We believe that because of this, people gave very sharp reviews about the vaccines (i.e. by directly using words such as 'hazardous', 'death', 'clueless', 'good', 'savior' etc.). So, the tweet text was entirely biased towards these words. The other data in the tweet text was almost unimportant. Over time, as governments from all across the world made it mandatory for everyone to get vaccinated and people could actually see the effects of vaccines, people's opinion about the vaccines got a little mild. So, instead of expressing their opinions in sharp words (i.e. 'hazardous', 'death') in the tweet text, they actually would write a tweet where the entire tweet data would give us a context rather than just a few words deciding the sentiment of the tweet. Based on this context, we can then decide if the tweet is positive, negative or neutral.

This would be an interesting study to know how a classification system built on one type of data (sharp words biased) works for the another type of data (context oriented). Among the pretrained text classification models, BERT and XLNet are the models used by Google in its data classification, data translation and question answering tasks . As XLNet outperforms BERT in several of the above mentioned tasks, we had an intuition that we would get best results with XLNet.

However, in our case, we obtained the best results using BERT. BERT typically masks a few words in a sentence and tries to predict these words by referring to the context in both directions of the words. One of the shortcomings of BERT is that it doesn't consider the relationship between the masked words. XLNet overcomes this problem of BERT.

The dataset that we'd worked on was more biased due to the sharp words used in it. So, the remaining text data has negligible contribution towards building a context for sentiment prediction. So, masking a few words in the tweet text does not affect the polarity of the tweet and hence BERT worked really well for us.

It would be really interesting to know which one of the above two pretrained models works well for the context oriented data (i.e data with tweets without any sharp words). It is our intuition that XLNet

would work better than BERT for context oriented tweets as the polarity of the tweets depend not only on a few words of the tweet text, but the entire context built by the tweet text data as a whole.

3.7 Experimental Design

To test the above stated hypothesis, we had to preprocess the data available at hand and then feed it to each of the pretrained models (i.e. BERT and XLNet). Only when we train our classification model using both pretrained models (i.e. BERT and XLNet), we could compare the performance of the two models on different datasets (i.e. our dataset - sharp words oriented and the other dataset - context oriented). So, we had to train our classification model using both pretrained models on the given data (i.e. sharp words oriented). For that, our data had to be clean enough to be fed to the pretrained models. So, we started with the basic preprocessing of the data.

As a part of basic preprocessing of the tweet text, we removed the twitter handle present at the end of every tweet and the URLs associated with that tweet. Additionally, we removed hashtags and stopwords from the tweet texts and applied lemmatization. After this basic preprocessing was complete, instead of feeding the data directly to the pretrained models, we decided upon trying some simple models like Logistic Regression, Naive Bayes, Random Forest Classifier and Linear SVC to get some idea about the further needed preprocessing of the data. With each basic algorithm that we use to train our model, we understood that our data needs more cleaning and precise representation.

Initially, we faced trouble in identifying why our trained model (trained using Logistic Regression) would not give good results in spite of the data being preprocessed using basic preprocessing steps. But, soon we realized that our dataset has tweets with a lot of emojis in it. We had to interpret those emojis to corresponding words so that they could help in determining the polarity of the tweet text too. So, we made use of the 'emoji_mapping' file to translate those emojis. Furthermore, while training these models using basic algorithms, we learned that the numbers and special characters need to be removed or translated using the regular expression function. In this way, we preprocessed the data until it was ready to be fed to the pretrained models.

To train our models using BERT and XLNet, we did the following steps (we used bert-base-uncased here): Transformer library installation Tensorflow library installation Import of classes like BertTokenixzer, TFBertForSequenceClassification, TFXLNetModel, XLNetTokenizer

4 Results

Table I summarizes the experiment results of the presented models evaluated in the testing set. Specifically, we reported the precision, recall, f1-score, accuracy, macro avg, and weighted avg of 3 different variant if each model. It can be observed that the individual BERT model classifier has comparable scores using all evaluation metrics. We can also combine those classifiers in ensemble framework to improve the performance by a significant margin, especially for the accuracy and the precision, with an advantage of the staking approach over the trivial majority voting.

Further analysis of the results per class shows that the performance measured by f1-score is the highest for the neutral sentiment class as shown in the diagram in appendix. These finding align with our expectations that the performance score in this class will be highest as it has the largest number of tweets. The negative sentiment tweets represent only 16.5% of the dataset size, whereas the neutral and positive comprises 44.4% and 38.9% of the dataset. By looking at the precision and recall, we noticed that recall is highest for the neutral class. The fact that this class is dominant in the dataset explains the high recall of scores. We can also look into confusion matrices of all models given in appendix. We can see the apparent confusion between neutral and negative class in all models. i.e many examples from positive class were mislabed as neutral and vice versa. Fig 1 and 2 in appendix illustrate word cloud in positive and negative class.

Algorithm	Accuracy	Precision	Recall	F-1 Score
Logistic regression	0.84	0.86	0.78	0.80
Random forest	0.87	0.88	0.87	0.87
MultinomialNB	0.81	0.81	0.81	0.80
LinearSVC	**0.89**	0.89	0.89	0.89
BERT	**0.94**	-	-	-
XLNet	0.62	-	-	-

5 Conclusion

The experiment concludes that BERT stands out among all the algorithms used. This is because BERT is convoluted and pre-trained on large corpus of text. All other algorithms outputs moderate to high accuracy.

To our surprise, XLNet performed worse than BERT even though [9] shows that XLNet outperforms BERT in 20 NLP tasks. We argue that this could be very specific data and pre-processing methods applied.

As future enhancement, features that are dropped in current model such as user_description, user_location, favorites, retweets, etc. can also be used along with the tweet text to predict the polarity of the tweet.

Moreover, the models that we have trained separately on our data can be combined to form an ensemble modeling system. The classifiers that we have used right now (i.e. Logistic Regression, Multinomial Naive Bayes, Linear SVC, Random Forest Classifier, BERT, XLNet) will act as base classifiers in ensemble model and we will predict the polarity of the upcoming new tweet based on the these base learners.

There are two methods that we can deploy to predict the polarity of the new data date of tweet:

[1] Majority Voting: In this method, we test the new data on each classifier and the polarity determined by the majority of the classifiers is assigned to the polarity of the tweet.

[2] Stacking: In this method, we train a meta learner over the output of our base learners. This meta learner can be any model which is easy to implement like, SVM, Naive Bayes etc. The idea is that the data obtained from base learners is now used to train the meta learner and the data tested on this meta learner is not used anywhere in the base learners before.

References

[1] Kazi Nabiul Alam et al. *Deep learning-based sentiment analysis of COVID-19 vaccination responses from Twitter data*. Dec. 2021. URL: https://www.hindawi.com/journals/cmmm/2021/4321131/.

[2] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2018. DOI: 10.48550/ARXIV.1810.04805. URL: https://arxiv.org/abs/1810.04805.

[3] Alexandre KOWALCZYK. *Linear Kernel: Why is it recommended for text classification ?* July 2020. URL: https://www.svm-tutorial.com/2014/10/svm-linear-kernel-good-text-classification/.

[4] *Machine learning blog & software development news*. URL: https://blog.datumbox.com/machine-learning-tutorial-the-multinomial-logistic-regression-softmax-regression/.

[5] Author links open overlay panelSymeonSymeonidisPersonEnvelopeDimitriosEffrosynidis-EnvelopeAviArampatzisEnvelope et al. *A comparative evaluation of pre-processing techniques and their interactions for twitter sentiment analysis*. June 2018. URL: https://www.sciencedirect.com/science/article/pii/S0957417418303683.

[6] A Ramalingaiah, S Hussaini, and S Chaudhari. "Twitter bot detection using supervised machine learning". In: *Journal of Physics: Conference Series*. Vol. 1950. 1. IOP Publishing. 2021, p. 012006.

[7] Mrityunjay Singh, Amit Kumar Jakhar, and Shivam Pandey. *Sentiment analysis on the impact of coronavirus in social life using the bert model - social network analysis and Mining*. Mar. 2021. URL: https://link.springer.com/article/10.1007/s13278-021-00737-z.

[8] Qingyao Wu et al. "ForesTexter: an efficient random forest algorithm for imbalanced text categorization". In: *Knowledge-Based Systems* 67 (2014), pp. 105–116.

[9] Zhilin Yang et al. "Xlnet: Generalized autoregressive pretraining for language understanding". In: *Advances in neural information processing systems* 32 (2019).

[10] Hui Yin et al. *Sentiment analysis and topic modeling for COVID-19 vaccine discussions - World wide web*. Feb. 2022. URL: https://link.springer.com/article/10.1007/s11280-022-01029-y.

GitHub Repository Link : https://github.ncsu.edu/msoni/engr-ALDA-Fall2022-P7