Text Classification of comments on COVID-19 vaccination using Supervised Learning Approaches

Kuvalaya Datta Jukanti, Shreyash Sanjay Kad
Data Science and AI
Chalmers University of Technology
kuvalaya@student.chalmers.se, shreyash@student.chalmers.se

Abstract

Text classification is one of the most prominent challenging tasks in the field of machine learning. It deals with the categorization of text documents into different classes. The goal of this technical paper is to describe a binary classification model for comments and opinions on COVID-19 vaccination. Various pre-processing techniques are used to extract the essential words and vectorize the text. Finally, classification is performed on the labelled comments being either pro-vaccination or anti-vaccination. The data is evaluated and compared on 4 different classifiers. The results shows that Bernoulli's Naive Bayes outperforms the other classifiers.

Keywords - Text classification; Binary classification; COVID-19 vaccination; Supervised Learning;

Introduction

With day-to-day advancements in the technology and its applications, textual data and the need to classify them has become one of the most important aspects in real world scenarios. Text classification is the process of categorizing the text documents into different classes. It helps in standardizing the platform and allows businesses to gain insights from the data and automate various processes.

This work is based on large collection of COVID-19 vaccination comments taken from various sources on social media platforms. The global pandemic hitting the entire world has made scientists work very hard and find a vaccine as a solution to eradicate the virus. However, as different vaccines are getting released on a timely basis, people have developed disparate thoughts about the vaccine. This paper presents the construction of a classification model for classifying the comments provided by people to determine whether their expressed opinion towards the vaccination is positive or negative. The classification here is a binary-labeled problem, where each comment can belong to the pro-vaccination(1) class or the anti-vaccination(0) class. The first phase of this work involves collection of data from different social media platforms and label them manually. The collected data is annotated multiple times by different people in order to get a firm decision on the label. The result of these annotations contributes to efficient text classification. The learning phase involves implementation of several pre-processing techniques including vectorization. The data is then utilized by numerous classifiers and based on the results, the best performing classifier algorithm is determined. The paper also focuses on the evaluation of the system by calculating metrics such as precision, recall, F1 and accuracy scores.

Methodology

The proposed system describes various classifiers used for the classification of user comments on COVID-19 vaccination. The implementation is structured into four modules namely, Collection of data, Preprocessing, Classification and Evaluation. The preprocessing stage involves techniques to clean and format the data while the classification phase involves the use of different classifiers to train and test the data. The evaluation module measures the accuracy and performance of the classification tasks.

1. Data

The text corpus carried out for this work is created manually. A group of 120 students have collected a total 100 comments each, 50 positive and 50 negative. The comments were labelled either as 0 or 1, where 0 represents anti-vaccination and 1 represents pro-vaccination. The labelled data undergone a second round of annotation by exchanging the documents amongst the students. The purpose for second round was to get more accurate and reliable labels for the comments. The comments which were not understood properly were labelled with the value -1. After two rounds of annotations, a set of around 8800 comments formed the training corpus while the testing set was around 400 comments. The model training and validation is done on the train set. Once the classifier is finalized, the final evaluation of the model is done on the test set.

2. Preprocessing

2.1 Formatting the data

The data set consists of two columns namely, comment and label. The 'label' column is annotated multiple times and hence the labels for each comment is more than one (Figure 1). In order to select a single value from the list of values, a function was implemented which performs a voting or a majority analysis. For each comment, based on the no. of values, the function picks the value which is repeated more

	label	comment
0	0/-1	It is easier to fool a million people than it...
1	0/0	NATURAL IMMUNITY protected us since evolutio...
2	0/-1	NATURAL IMMUNITY protected us since evolutio...
3	1/-1	The bigest sideffect of vaccines is fewer dea...
4	0/-1	90% of people that get vaccinated don't get th...

Figure 1: Sample training data

number of times in the list. If the values are tied, it picks the value that is newest i.e. the last value in the list. For example, if a comment has 0/0/1 as the label, the function chooses the majority value i.e. 0. Considering an another case where the label is 0/1 for a comment, the frequency is equal for both the values. In such a case, the function chooses the last i.e. the newest annotation made on the comment which in this case is 1.

It was observed that the data is very balanced and reliable after checking the value counts of each label. However there were very few (around 9%) discrepancies which has the label -1. These comments were more in a generic structure and not taking any side. These comments of label value -1 were dropped from the data set to avoid the noise in the training phase. After removing the comments with label -1, the number of pro-vaccine comments and anti-vaccine comments are almost same in other words the training set is not biased.

2.2 Normalizing case

Lowering the case of the text is the first pre-processing stride of any text classification method. The purpose of normalizing the case is because the model might treat a word which is at the beginning of the sentence starting with a capital letter different from the same word which might appear later in a sentence with no capital letters. This will lead to a decrease in the accuracy. Lowering the case helps in solving this problem. In our experiment, python's string lower() method is used to change the case of the text.

2.3 Expanding Contractions

Contractions are shortened versions of words or syllables. They are created by removing one or two letters from the words and formed by combining two words. The missing letters are indicated by using a apostrophe for representation. Nowadays, contractions are widely being used everywhere and might as well be used in the written comments on the social media platforms. This would affect the accuracy as the model may not understand the meaning of the words and would also create a problem while removing punctuation as it separates one word into two.

For example, "I don't want to take the vaccine" might not be understood and predicted accurately by the model. Hence the sentence is formatted to "I do not want to take the vaccine" by expanding the contractions. The conversion of contractions to its original form aids in text standardization.

In our experiment, a dictionary is created with the contractions as the keys and the expanded contractions as the values. This dictionary can be found on internet as it openly available for users to use. The textual data is compared with the dictionary and for all the available contractions, its corresponding expanded contractions are replaced in the text.

2.4 Converting emojis to words

People often use emojis or emoticons in a sentence to express their feelings instead of writing the same in words. Emojis and emoticons convey an expression in a message or a sentence. In the field of text classification or text analysis, these need to handle them. The simplest approach is to remove them from the text. But the possible affect of this would be to not understand a person's opinion on the vaccination as they might have expressed their thoughts in the form of emojis rather than in words. So, a better approach to deal with this is to convert emojis to words which will help to preserve information rather than losing information. In our implementation, the python emoji is library is used. The function emoji consists of an attribute demojize which is applied on the text to translate the emojis to words.

2.5 Removing punctuation

An important step in the text pre-processing phase is the removal of punctuation marks. Punctuation marks are used to divide text into sentences, paragraphs, frame the sentences etc. Non-removal of these will affect the results especially when finding the frequencies of words and phrases in a text processing approach. Since the punctuation marks are more frequently used in a text, these would affect the results of the model. Removing punctuation can be achieved by using the string.punctuation which is pre-initialized string constant in python. By applying the above method, all the text is preserved which is not in the the string constant.

2.6 Removing Numbers

Since the focus is on the text data, numbers might not add much information in the processing. Preserving such uninformative data in the text might add noise and thus can be removed. This can be achieved by the string.digit method similar to how the punctuation was removed. By applying this method, all the numbers from the text would be dropped.

3. Training

3.1 Feature Extraction

Feature extraction is the process of transforming the words of text to a feature set i.e. real valued vectors to make it usable by the classifiers. The feature extraction for our problem is performed by TF-IDF which stands for Term Frequency - Inverse Document Frequency. It is a numerical statistical measure that estimates how relevant a word is to a document in a collection of documents.

The term frequency (TF) is defined as the the frequency of words, say w, appearing in a document and document frequency (DF) is the number of documents that contains the word w. However, some of the frequent words are not pertinent for the task and therefore the weights of such words have to be diminished. The inverse document frequency (IDF) approach is used to distinguish between the relevant and non-relevant words and minimizes the weights of the non-relevant words while maximizing the weights of less frequent words. It is represented as:

$$\text{TF(word, text)} = \frac{\text{number of times the word occurs in the text}}{\text{number of words in the text}}$$

$$\text{IDF(word)} = \textbf{log}\left[\frac{\text{number of texts}}{\text{number of texts where the word occurs}}\right]$$

$$\text{TF-IDF(word, text)} = \text{TF(word, text)} \times \text{IDF(word)}$$

Figure 2: Computation of TF-IDF features

The process of TF-IDF is performed in a vector space and therefore, the TF-IDF feature extraction is performed on the training set by fitting and transforming the textual data into vectors. The data is thus represented in the form vector features.

The TF-IDF in scikit-learn library is very flexible to tune the hyper parameters depending on the use case. For our use case, some of the hyper parameters were changed. By default, the TF-IDF removes all the stop words in English language. This would affect the performance of our system as many of the stopwords in the English language such as 'no', 'dont', 'yes', 'not', 'shoulnt' are important for the model to clearly understand and predict accurately. Hence, the parameter stopwords is assigned to "None" denoting that no stop words will be removed while processing. Another hyper parameter change is in the 'ngram' range. By default, the ngram range is (1,1) considering only the unigrams. For our problem, considering the ngram range as (1,2) i.e. considering unigrams and bigrams was sensible.

3.2 Classification Algorithms

In order to train and test on the training data, the train set is divided into training and validation sets. There are several classification algorithms designed for various purposes. For this paper, four different classification algorithms are used in order predict and evaluate the performance. All these algorithms work well for binary labels and based on the results from these classifiers, the classifier providing the best accuracy is chosen to predict on the final testing set. All the classifiers were evaluated based on the precision, recall and accuracy scores.

Naive Bayes Classifier: A Naive Bayes classifier is a probabilistic machine learning model based on the Bayes theorem that is used for classification tasks. For our implementations, Bernoulli's Naive Bayes classifier is used since it works well on problems consisting of two classes.

Support Vector Machines: It is a supervised machine learning technique used for classification, regression and outlier detection. It is very effective working in high dimensional space. LinearSVC is the class which is capable of performing efficiently on binary-labelled and multi-labelled datasets.

Logistic Regression: It is statistical linear model that uses a logistic function to model the probability of certain class which is mostly binary.

Stochastic Gradient Descent classifier: Stochastic Gradient Descent (SGD) is an efficient approach to fit linear classifiers and regressors under convex loss functions such as Support Vector Machines and Logistic Regression.

4. Testing, Results and Discussion

4.1 Evaluation metrics

The metrics used for determining the quality and evaluation of the system are precision, recall, and accuracy measures. They are calculated by the following formulas:

Accuracy = (TP + TN) / (TP + TN + FP + FN)
Recall = TP / (TP + FN)
Precision = TP / (TP + FP)

TP and TN correspond to the number of true positives and true negatives i.e. the number of comments that are correctly classified as pro-vaccination and the number of comments that were correctly classified as anti-vaccination respectively. On the other hand, FP and FN correspond to the number of false positives and false negatives i.e. the number of comments that were incorrectly classified as pro-vaccination and the number of comments that were incorrectly classified as anti-vaccination. They can be represented in the form of a confusion matrix.

4.2 Results and Discussion

The results of all the classifier algorithms performed on the training set is given as follows:

	Classes	Precision	Recall	F1 - score	Accuracy
Bernoulli Naive Bayes	0	83%	82%	82%	83%
	1	83%	83%	83%	
Linear SVC	0	80%	85%	82%	83%
	1	86%	81%	84%	
Logistic Regression	0	78%	83%	80%	81%
	1	84%	79%	82%	
Stochastic Gradient Descent	0	80%	85%	82%	83%
	1	86%	81%	83%	

Figure 3: Evaluation scores of classifiers

To summarize the results, Bernoulli's Naive Bayes has proved its efficiency over the other classifiers where it has gained the best result in all of the evaluation measures taken into account. Although Naive Bayes, Linear SVC and SGD algorithms achieved the same accuracy, looking at the columns 0 (anti-vaccination) and 1 (pro-vaccination) seems to be very balanced in the case of Bernoulli's Naive Bayes compared to the other two classifiers where the precision, recall and accuracy scores for both classes differ by a large amount.

Now, using the Bernoulli's Naive Bayes classifier, the prediction is performed on the final testing set and determine the accuracy using the same evaluation measures. By predicting on the test set, the model gives 90% accuracy. The classification report looks like the follows:

	precision	recall	f1-score	support
0	0.87	0.92	0.89	189
1	0.92	0.88	0.90	203
accuracy			0.90	392
macro avg	0.90	0.90	0.90	392
weighted avg	0.90	0.90	0.90	392

Figure 4: Classification report

It is also observed that the number of false positives and false negatives are less (Figure 5). These false positives and false negatives occur due to multiple factors. From the observations made on our work, some reasons have made us believe about why the system performed those errors and what might have gone wrong. Based on the confusion matrix, we extracted the statements which were predicted falsely. Our observations and findings found the following factors to be the reason for the errors.

1. If the text does not belong to any class. For example, text such as "Nooooo thank you"

2. If an anti vaccine comment contains words that usually are present in pro vaccine comments. For example we found text like, "No for me even if every one gets vaccineted. I will wait for real vaccine".

3. Due to incorrect labels. Our findings include that few pro-vaccine comments are labeled as anti vaccine.

4. If the comment is not in the form of statement rather in the form of question. Eg, "Why should I get vaccinated? Whats the proof it is effective"

5. Another example such as "got my second vaccine 3 days ago no fatigue no fever no side effects". This comment was classified under anti-vaccination category but the statement should be under pro-vaccine. The reason might be because of the word "NO" in the statement.

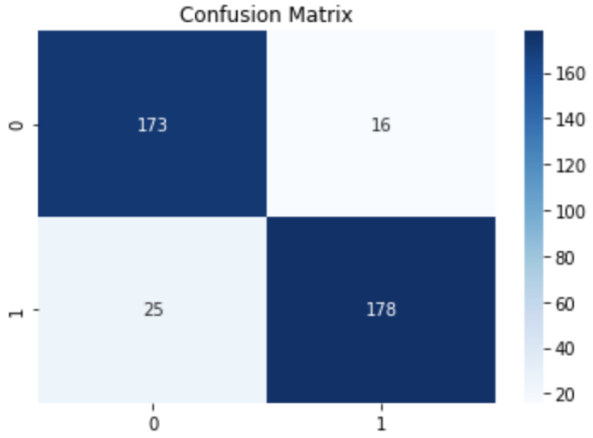

Figure 5: Confusion matrix

Furthermore, to understand which features the model considers important, the model attributes "get_feature_names()" and "coef_" can be used to get the values and sort them in descending order (Figure 6). This will tell what features are most important in the process of prediction.

feature_names	importances
hope	1.876005
yes	1.698532
poison	-2.026200
no	-2.264533

Figure 6: Feature Importances

Conclusion

Analyzing the text data and classifying into different classes is a critical task. This work is focused on applying common pre-preprocessing techniques and apply TF-IDF to vectorize the text and evaluate its effectiveness by running the data through several classifiers. The results showed that Bernouillis Naive Bayes classifier is the better model for our data and study.