Classifying Twitter data for the COVID-19 epidemic

Ananya H[*1], Muthualagesan Suryavarathan[*1], Ponni M[*1]

Abstract

In a time of crisis, social media tends to be people's go-to place for sharing information quickly and catering to a wider audience. This generates a significant amount of data, some of which are extremely valuable to be used for relief work. Here, we intend to study the nature of social-media content generated during the ongoing corona epidemic. We evaluate our techniques on a twitter dataset through a set of carefully designed experiments, to see which method fits best to extract useful information for this particular type of data.

Keywords: Health Crisis, COVID-19, Epidemic, Twitter, Classification

1. INTRODUCTION

Coronavirus disease 2019 (COVID-19) is an infectious disease caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). First identified in 2019 in Wuhan, it spread globally, resulting in the World Health Organization (WHO) declaring the outbreak as a Pandemic. As of 1 May 2020, more than 3,400,000 cases of COVID-19 have been reported in over 212 countries and territories, resulting in approximately 240,000 deaths. More than 1,080,000 people have since recovered.

During crises such as these, microblogging platforms like Twitter are widely used by affected people to post updates and awareness messages. These crisis-related messages disperse among multiple categories like available facilities, asking for help, warning about a region, information about affected, recovered, injured or dead people, etc. Extracting important situational updates from a plethora of such messages is a difficult yet important task. Given the importance of topic-specific tweets for a time-critical situational response, the crisis-affected communities and professional responders may benefit from using an automatic system that acts as the link between those who are ready to help and those who need help. We aim to excerpt relevant nuggets of information and assign them informational categories using appropriate machine learning techniques.

2. RELATED WORK

The Qatar Computing and Research Institute (QCRI)[1] has done a lot of research work for crisis computing using Machine Learning, for many calamities like Floods, Earthquakes, etc. Muhammad Imran has extensively contributed to using the power of AI for disaster response and has written surveys on processing emergency messages in social media. His research work involved "solving multiple challenges in the same, including parsing informal messages, handling information overload, and prioritizing different types of information found in messages"[2] and mapped these challenges to classical information processing operations such as filtering,

* Equal Contribution, [1]Department of Applied Mathematics and Computational Sciences, PSG College of Technology, India

classifying, ranking, aggregating, extracting, and summarizing.

H. Purohit et al, (2013)[3] in their paper on Emergency relief coordination on social media, presented machine learning methods to automatically identify and match needs and offers communicated via social media for items and services such as shelter, money, clothing, etc.

3. DATASET AND CLASSIFICATION

On Twitter, thousands of updates and opinions regarding this epidemic keep springing up every other day. Our aim is to classify the data into relevant labels so that the required information alone can be harnessed. We collected around 1000 tweets (as of 29th March 2020) directly from the twitter API. Though the number of tweets generated is a higher number, the number of irrelevant tweets were found to be pretty high from the reading, so we constrained the collection to only those tweets having at least 100 likes, 30 retweets and 30 replies. Basic pre-processing was done to it (such as removal of hyperlinks, mentions, hashtags, emoticons and out-of-vocabulary words used in social media).

3.1 Types of tweets posted

K Rudra et al, in their paper on "Classifying information from Microblogs during Epidemics"[4] classified tweets posted during epidemic into following classes — (i). symptom, (ii). prevention,
(iii) transmission, (iv) treatment, (v) death report, and (vi) nondisease.
In this work, the data was thoroughly read to understand the kind of labels that we can use for this model and decided on the below four classes:

1. Reports (Statistics and Government actions)
2. Awareness (Intended for prevention of the spread of epidemic)
3. Treatment (Facilities, Tests and Medicine)
4. Irrelevant (Everything else)

The data were manually annotated. Inter-annotator disagreements were solved through thorough discussions and research.

Table 1: Example of various types of tweets.

Type	Tweet text
Reports	#Coronavirus Number of deaths in Italy rises to 2978 an increase of 475 in 24 hours
Awareness	Washing your hands frequently with soap and water or a hand sanitiser gel as an alternative will remove viruses and bacteria from your hands #COVID19 advice.
Treatment	US researchers began the first human trial of a #Coronavirus vaccine More via business
Irrelevant	Dozens stock up on guns in the US amid #Coronavirus outbreak preparing for the worst?

4. DATA PRE-PROCESSING

Feature extraction and pre-processing are crucial steps for text classification applications. In this section, we introduce methods for cleaning text data sets, thus removing implicit noise and allowing for informative featurization.
Most text and document data sets contain many unnecessary words such as stopwords, misspelling, slang, etc. We have

removed all duplicate entries, numbers, special characters, punctuations, double-spaces and changed the data completely to lower case. Then we have removed all stop words and words with length one.

Inspecting at the distribution of data available in each class, we do upsampling wherever necessary.

Tokenization is a pre-processing method which breaks a stream of words into words, phrases or other meaningful elements called tokens. We have tokenized the dataset and found the frequency of occurrence for every token.

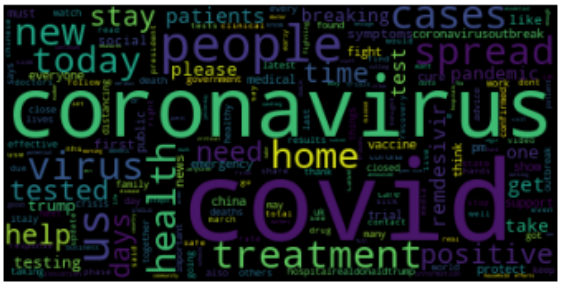

Fig 1: A visual representation of the vocabulary scaled to their frequencies.

5. FEATURE EXTRACTION

The mapping from textual data to real-valued vectors is called feature extraction. An n-gram is a contiguous sequence of n items from a given sample of text. We have used n-gram to find the common words that occur in all four classes and removed them from the dataset as they don't contribute to classification. We have used 1-gram and 2-gram.

CountVectorizer is used to transform corpora of text to a vector form of term/token count. We have fit the vectorizer on our corpus for each class separately with an n-gram range of 1 and 2. We have taken the top 10 unigrams (bigrams) from each class and found the intersection and removed them entirely from the corpus.

We have used the Term Frequency - Inverse term frequency technique on the corpus before feeding it into our models. Inverse Document Frequency(IDF) is a method used in conjunction with the term frequency(TF) in order to lessen the implicitly common words in the corpus. IDF assigns a higher weight to words with either high or low frequencies in the document. This combination of TF and IDF is well known as Term Frequency-Inverse document frequency (TF-IDF).

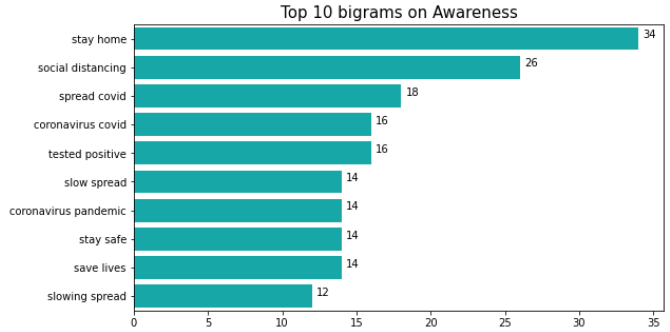

Fig 2: Bar plot of Top 10 bigrams of Awareness class.

6. MODEL SELECTION

For these experiments, we consider the state of the art models of supervised learning and compare the performances of each. Given that the data after the preprocessing steps are still in a string format, we need to get their word embeddings to carry out the experiments. To suit the needs of our data, we take the help of two mechanisms to get the task done easier and more efficiently: Pipelining and Grid Search.

6.1 Pipelining in Python

In any Machine learning problem statement, there is usually a fixed sequence of steps involved. Pipelining aims to chain the

multiple steps into one and thus, automating the process. Some codes are meant to transform features- say, normalising numericals or turn text into vectors, those kinds are the transformers. Other codes are meant to predict variables by fitting an algorithm such as a random forest or support vector machine, those are the estimators.

So, in a pipeline, it first sequentially applies a list of transformers (data modelling) and then a final estimator (ML model). It is a part of the scikit-learn library.[5]

6.2 Grid Search in Python

Grid search is used to perform hyperparameter tuning in order to determine the optimal values for a given model. The value to stipulate the hyperparameter that gives maximum accuracy is a very important factor determining the result, and here Grid Search CV of the scikit-learn library does the same efficiently.

6.3 The chosen models

6.3.1 Count Vectoriser and Logistic Regression

Count vectorising and then a logistic regression model was fed to the pipeline in that order, using the liblinear library for the latter, and an n-gram range of (1,1), (2,2) and (1,3) for the former.

6.3.2 Tfidf Vectoriser and Logistic Regression

The above models were pipelined and experimented with different ranges of parameters for tuning.

6.3.3 Tfidf vectoriser and multinomial Naive Bayes

Following the Tfidf embedding, a multinomial Naive Bayes model was deployed. The advantage of this model is that it specifically works well for text embeddings. Unlike the simple naive Bayes modelling a document based on the presence and absence of some words, multinomial naive Bayes explicitly models the word counts and adjusts the underlying calculations to deal with it.

6.3.4 Count vectoriser and multinomial Naive Bayes

Keeping in mind the advantages mentioned above for the multinomial NB model, we try leveraging it again, but with count vectoriser as a prerequisite. Parameters were adjusted accordingly

6.3.5 TfidVectoriser and Stochastic Gradient Classifier

We further experimented with using the SGD Classifier for the data taking a squared loss and experimented over a range of suitable alpha values.

6.3.6 The Sequential Deep Learning model

With the help of Keras in Tensorflow, we built a fully connected neural network with 512 layers and softmax activation function. The appropriately formatted text and label are fed into it with a categorical cross-entropy loss function.

7. RESULTS

7.1 Performance metrics

This section presents the experimental results and performance metrics for different models. Accuracy is one of the common

performance metrics. It is the measure of all the correctly identified cases. It is mostly used when all the classes are equally important. Accuracy is the proportion of correctly classified examples to the total number of examples, while the error rate is incorrectly classified instead of correctly.

$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP is True-positive, TN is True-negative, FP is False-positive and FN is False-negative. The six models discussed in Section 6.3 have yielded different accuracy scores, which are displayed in the below bar graph.

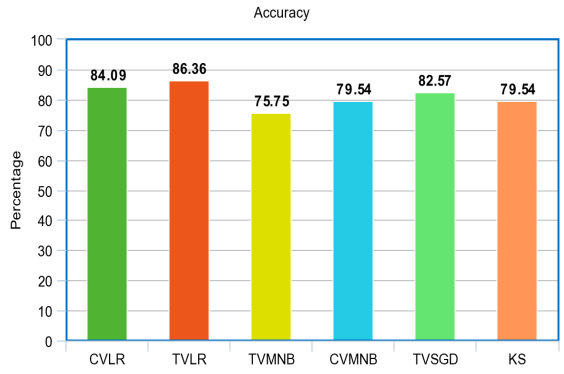

Fig 3: Bar graph of the accuracy measures of different models.

Out of the discussed models, it is evident that *TfidfVectorize & Logistic Regression* yields the highest accuracy score of 86.36% followed by *CountVectorizer & Logistic Regression*. Other performance metrics such as Precision, Recall and F1-score have also been calculated for all the models.

$$Precision = \frac{tp}{tp + fp}$$

$$Recall = \frac{tp}{tp + fn}$$

$$F = 2\frac{Precision * Recall}{Precision + Recall}$$

We display these metrics for the most accurate model.

	precision	recall	f1-score
Awareness	0.92	0.92	0.92
Irrelevant	0.78	0.93	0.85
Report	0.81	0.61	0.69
Treatment	1.00	0.96	0.98
accuracy			0.86
macro avg	0.88	0.85	0.86
weighted avg	0.87	0.86	0.86

Fig 4: Performance metrics of TV & LR.

7.2 Misclassification of tweets

A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. We also display the confusion matrix for the most accurate model with the help of a heatmap.

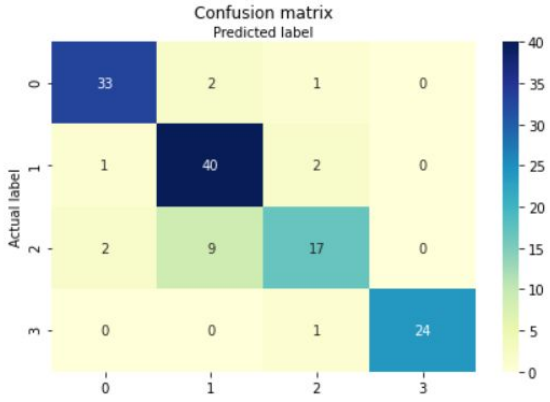

Fig 5: Representation of the best model using the Confusion matrix.

From the above confusion matrix, it is evident that more around 13% of the tweets have been misclassified. It is observed that in these misclassified tweets, the tweets contain information about more than one class like both Reports and Awareness or Reports and Treatment.

8. CONCLUSION AND FUTURE WORK

Natural Disasters and Epidemics create various kinds of problems for people: physically, mentally and emotionally. Though this does not have a very direct impact, harnessing the power of Machine Learning is in some way associated with making the situation better, giving how it has become like the go-to place for people to rant about their condition, seek help, announce their plan to provide help or spread awareness. Given the vastness and distribution of this data, classifying it also has the capability to go a long way in providing even health organisations and NGOs seeking to provide help with relevant information.

With regards to future work, we seek to expand our dataset, both in terms of numbers and diversity, as a first step to achieve better results. More relevant labels can be added according to the status quo, and various deep learning models can be experimented on.

REFERENCES

[1] https://crisisnlp.qcri.org/

[2] Muhammad Imran, Carlos Castillo, Fernando Diaz, and Sarah Vieweg. 2018. Processing Social Media Messages in Mass Emergency: Survey Summary. In *WWW '18 Companion: The 2018 Web Conference Companion, April 23–27, 2018, Lyon, France. ACM, New York, NY, USA* 5 Pages.

[3] Hemant Purohit, Carlos Castillo, Fernando Diaz, Amit Sheth, and Patrick Meier. 2013. Emergency-relief coordination on social media: Automatically matching resource requests and offers. First Monday.

[4] Koustav Rudra, Ashish Sharma, Niloy Ganguly, and Muhammad Imran. 2017. Classifying Information from Microblogs during Epidemics. In *Proceedings of the 2017 International Conference on Digital Health* (*DH '17*). Association for Computing Machinery, New York, NY, USA, 104–108.

[5] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.