A fake news detection model applied to COVID-19 related news

Isuru Abeysekara

Abstract: Mass hysteria about COVID-19 has caused the spread of misinformation about COVID-19 related news. A classification model using unigram analysis on news headings can help the general public understand what news to trust.

Key terms: Coronavirus, COVID-19, fake news detection, infodemic, misleading information, pandemic, SARS-CoV-2, social media, social networks, text classification, text mining, web mining, WHO.

At the end of December 2019, the World Health Organization (WHO) was informed of a cluster of pneumonia cases of unknown cause that were detected in the city of Wuhan, Hubei Province, China. Initially, these patients were diagnosed as having acute pneumonia. Most of them worked in a wet market in Wuhan and showed common symptoms of fever, dry cough, tiredness, and in more severe cases breathing difficulty. However, these symptoms were not of acute pneumonia as was first thought. With the increasing number of cases, China informed the WHO of the situation and its unknown cause in early January 2020.

The WHO named the virus ‘‘Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2)’’ and the disease as ‘‘Coronavirus Disease (COVID-19)’’. COVID-19 is a global health problem that requires extreme caution, strict maintenance of personal and general hygiene, and the cleanliness of all places. These practices help in avoiding the occurrence of mutations so that the virus can be controlled and contained. All reports issued by the WHO indicate that the epidemiological situation (since the beginning of January 2020) is very critical and scientists are frantically working to develop a vaccine to eradicate the virus.

Due to the novel nature of COVID-19, researchers have not had enough time to conduct in-depth research on its nature. Also, its widespread geographic impact over a very short period has made it a highly relevant issue and caused mass public hysteria. As a result, there exists a great deal of misinformation that's available to the public. Since the spread of COVID-19 is a global pandemic, the average individual must have access to the right information. Also given the spread of information possible over the internet, news articles become a quick and easy method for one to be informed about current affairs, including a global pandemic. The problem is that the Internet's speed of information dissemination also brings out the hysteria driven misinformation.

The misleading information may be intended to disrupt the economy of countries, reduce people’s confidence in their governments, or promote a specific product to achieve enormous profits. This has already happened with COVID-19. The shared misleading information about lockdowns, vaccinations, and death statistic, have fueled the panic of purchasing groceries, sanitizers, masks, and paper products. This led to shortages that disrupted the supply chain and exacerbated demand-supply gaps and food insecurity. Moreover, it has caused a sharp decline in the international economy, severe losses in the value of crude oil, and the collapse of the world’s stock markets. Additionally, some people have lost

faith in their governments as in Italy and Iran, due to the spread of COVID-19 and the shortage of medical protection products all over the world. All these are leading the world into an economic recession.

The average consumer of news articles does not have a guide to decide what news outlets to subscribe to. Is there a method to develop a means to classify true and fake news headings? This paper proposes a fake news headlines detection model using unigram analysis in the English language relying on sources such as the UN, UNICEF, WHO and other individual news organizations[1].

Project Workflow:

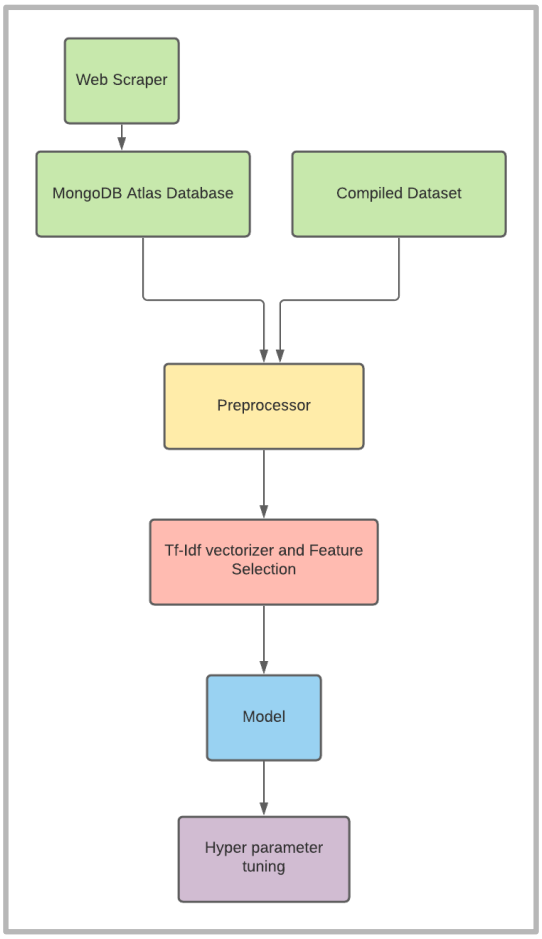

Figure 1: Project Workflow

[1] Detecting Misleading Information on COVID-19, Elhadad et al, 2020

The proposed model had the following stages:

1) Information collection: The data gathered for this project was two-fold. First, the model used data from the UN, UNICEF and WHO. Second, it also had to rely on third-party scraped datasets. The final dataset was a merged result of this two-pronged approach.
2) Preprocessing
3) Feature extraction
4) Model Building
5) Optimization

Information collection:

Extracting information on news headlines was initially challenging. The problem was determining whether a news article was fake or not because headlines do not typically come with a tag -True or False- to validate its claims. The only possible way forward was to outsource the annotation of headlines to a more informed third party. Susan Li's dataset on covid related headlines proved to be a well-crafted dataset, with a good mix of true and false labels.[2] The methodology she uses is similar to the data collection approach this project uses. She built a scraper to extract headlines from news articles from a variety of news organization. However, this dataset alone was not enough, and there was a need to safeguard that the model was trained on data that were as accurate as possible. A web scraper extracted new article headlines from the UN, UNICEF and WHO. The scraper extracted all the information related to the COVID-19 outbreak from these organizations' daily situation reports from the news published on their websites' newsroom. The news headings were scanned across a set of keywords to extract covid related news. This set was defined as follows:

– ‘‘Coronavirus’’
– ‘‘Corona_virus’’
– ‘‘Corona-virus’’
– ‘‘Novel_Coronavirus’’
– ‘‘2019-nCoV’’
– ‘‘Novel-Coronavirus’’
– ‘‘NovelCoronavirus’’
– ‘‘2019_nCoV’’
– ‘‘nCoV’’
– ‘‘COVID-19’’
– ‘‘SARS-CoV-2’’
-”covid19”

[2] https://towardsdatascience.com/automatically-detect-covid-19-misinformation-f7ceca1dc1c7

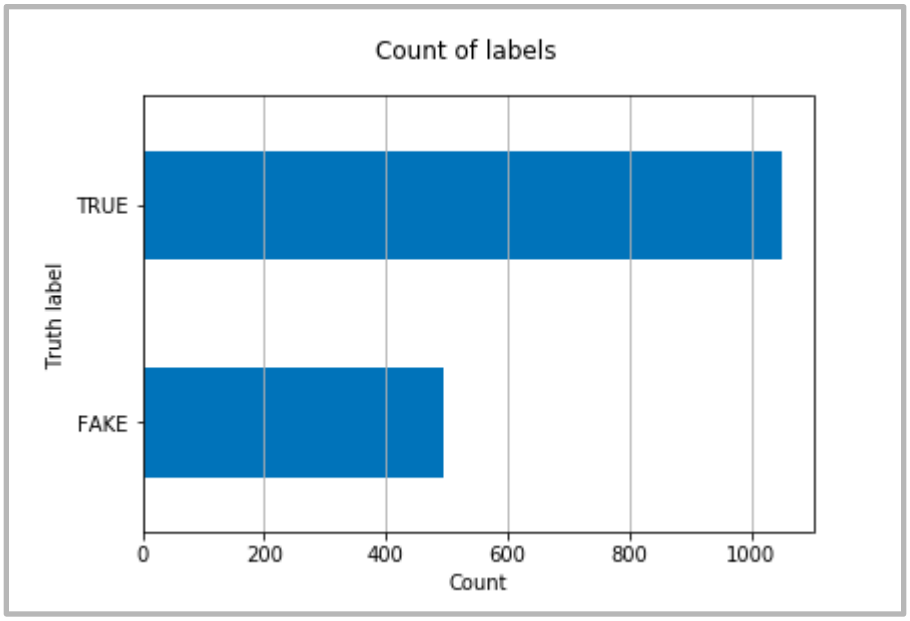

Figure 2: Count of labels

The final dataset had 1049 true headings and 495 fake headings.

The scraper stored the news headlines in a MongoDB Atlas database. It utilized python's Selenium and Beautiful Soup packages in the information collection database.

Preprocessing and feature extraction:

I created a preprocessor class that prepared the data for model training. The preprocessing relied on several conventional methods.

a) Removing stop words from the headlines. I found that nltk's stop word did not exhaustively remove all stop words from the headings. This improved stop word list helped the model perform better since it included abbreviations such as "don't" instead of "do not" for example.
b) Removing punctuation from headlines
c) Lemmatizing the headlines so that inflections of a given word are converted back to its root.
d) Tokenizing the headlines. Treating words in the scraped word set (such as "covid 19") as a single word was a challenge as the preprocessor split the phrase into covid and 19. As a result, I used nltk's multi-word expressions to add rules to the tokenization such that the phrases in the word set were treated as single words.

Feature extraction:

I employed Tf-idf as the main method of feature extraction. Most researchers depend on the use of TF-IDF as a feature extraction technique. Hence, we used TF-IDF to give numerical weights for the textual content to be used for mining purposes. TF-IDF computes the importance of a term t based on how

frequent it is within a document d, where it belongs, and its relative importance within the whole training dataset D.

The TF-IDF measure, as shown in equation (1), has a weight calculated by multiplying two values: the normalized Term Frequency (TF) and the Inverse Document Frequency (IDF)[3].

$$\mathbf{TF - IDF\ (t, d) = TF(t, d) \times IDF(t, D)}$$

Equation 1

It became clear that the feature extraction could be optimized further as not all features extracted from the headings would be important. I set about to perform a series of tests to extract the most important features.

1) *Recursive feature elimination:* The goal of recursive feature elimination (RFE) is to select features by recursively considering smaller and smaller sets of features. First, the estimator is trained on the initial set of features and the importance of each feature is obtained either through any specific attribute or callable. Then, the least important features are pruned from the current set of features. That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached. I used sklearn's RFE package to perform this test[4].

2) *Ridge regression regularization:* In linear regularization, a penalty is applied across all coefficients that multiply each of the predictors. Ridge regression adds a squared magnitude of coefficients as the penalty term[5].

3) *Chi-squared variance reduction:* In the case of classification problems where input variables are also categorical, we can use statistical tests to determine whether the output variable is dependent or independent of the input variables. If independent, then the input variable is a candidate for a feature that may be irrelevant to the problem and removed from the dataset[6].

4) Tree-based feature importance: Tree-based estimators (see the **sklearn. tree** module and forest of trees in the **sklearn.ensemble** module) can be used to compute impurity-based feature importances, which in turn can be used to discard irrelevant features[7]

Performing feature reduction using these tests notably increased the accuracy of the models.

[3] Detecting Misleading Information on COVID-19, Elhadad et al, 2020
[4] https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html
[5] https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c
[6] https://machinelearningmastery.com/chi-squared-test-for-machine-learning/
[7] https://scikit-learn.org/stable/modules/feature_selection.html

Classifier	*Accuracy prior feature selection*	*Accuracy post feature selection*
Logistic Regression	0.82	0.85
Decision Tree	0.80	0.82
AdaBoost	0.80	0.83
Random Forest	0.83	0.85

Table 1: Comparison of results before and after feature selection

Model Building:

I utilized a stacking classifier as the classification algorithm. Stacked generalization consists of stacking the output of an individual estimator and use a classifier to compute the final prediction. Stacking allows using the strength of each estimator by using their output as the input of a final estimator[8]. The algorithms I stacked together include a Random Forest Classifier, a Complement Naive Bayes classifier and a Linear SVC. The stacking classifier also requires a final classify to predict the training data. To ensure that I picked an ideal algorithm, I used Random Forest, Gaussian Naive Bayes, Decision Tree and a K Neighbors classifier. My main metrics of performance were accuracy (across 5 cross-validations) and weighted f1 scores.

As per the table below, the Random Forest Classifier performed the best in terms of accuracy.

Classifier	F1 weighted	Accuracy
Logistic Regression	0.87	0.8**46**
Random Forest	0.85	0.85**39**
Decision Tree	0.82	0.82
AdaBoost	0.83	0.83

Table 2: Comparison of final f1 and accuracy scores of the classification algorithms used

Optimization:

Since I established that the Random Forest Classifier performed the best, I went about improving its performance. First, I performed a grid search using the following parameters.

[8] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingClassifier.html

```
{'n_estimators': range(50, 150, 10),
                'criterion': ["gini", "entropy"],
                'min_samples_leaf': range(1,3),
                }
```

Next, I altered the class weight hyperparameter to optimize results using a random search. I had to switch to random search because my range of class weights was large and it was computationally expensive to perform a grid search.

The accuracy scores of the random forest classifier improved greatly at the end of this stage as the table depicts.

Intuitive Results and implications

Figure 3: Words present in true (top) and fake (bottom) headings

The final model shows what words to look out for in true and fake news headings. Notice that words such as unicef, health and global appear in the true heading wordcloud. Bear in mind that the model has not achieved a hundred percent accuracy so there exist terms that appear in both word clouds.

Conclusion and possible improvements:

One possible avenue that can help the model perform better is scanning the content of the news articles, instead of scanning headlines alone. This would mean that the web scraper needs to change to a web

crawler to scan articles along with headings from the news sections of the UN, UNICEF and WHO web pages.

In terms of feature extraction, this project does not utilize sentiment analysis. Using nltk's Vader sentiment analyzer will help determine the overall sentiment of a headline. It could be the case that the sentiment of fake news headlines is different from true headlines.

In terms of optimization, there are many more hyperparameters that the Random Forest Classifier possesses. I did not consider all of them because doing so would be computationally expensive, but tuning these hyperparameters may open up possibilities for further optimization.

This model puts forth a framework for fake news classification of COVID related information. I believe that this model can come to use especially when covid related cases rise in possible future waves. Further, this model is not limited to health related issues. In fact, it is applicable to other fake news classification problems with small tweaks in its approach and design.