Covid-19 Open Research Dataset Challenge (CORD-19) - Project Report

Using NLP to Answer High-Priority Scientific Questions Related to COVID-19

Joseph Cheng

June 23rd, 2020

Table of Content

Using NLP to Answer High-Priority Scientific Questions Related to COVID-19 1

Table of Content 1

1. Definition 2

1.1. Project Overview 2

1.2. Problem Statement 3

1.3. Metrics 4

2. Analysis 5

2.1. Data Exploratory with Exploratory Visualization 5

2.1.1. Metadata Analysis 5

2.1.2. Medical Articles (PDF_JSONS) Analysis 6

2.1.3. Priority Questions (2_relevant_factors) Analysis 6

2.1.4. Other key abnormalities identified 7

2.1.5. Key Outcome from Data Exploratory 8

2.2. Algorithms and Techniques 8

2.2.1. Word Vectorizers - aka one hot encoding 9

2.2.1.1. Countvectorizer 9

2.2.1.2. TF-IDF Vectorizer 9

2.2.2. Search Engine Match Algorithm 10

2.2.3. Supervised Machine Learning model 10

3. Benchmark Model - TF-IDF Search Engine 10

3.1. Creating the TF-IDF Vector Space Model 10

3.2. Running the Search Engine 11
3.3. Evaluation metrics for Search Engine 11

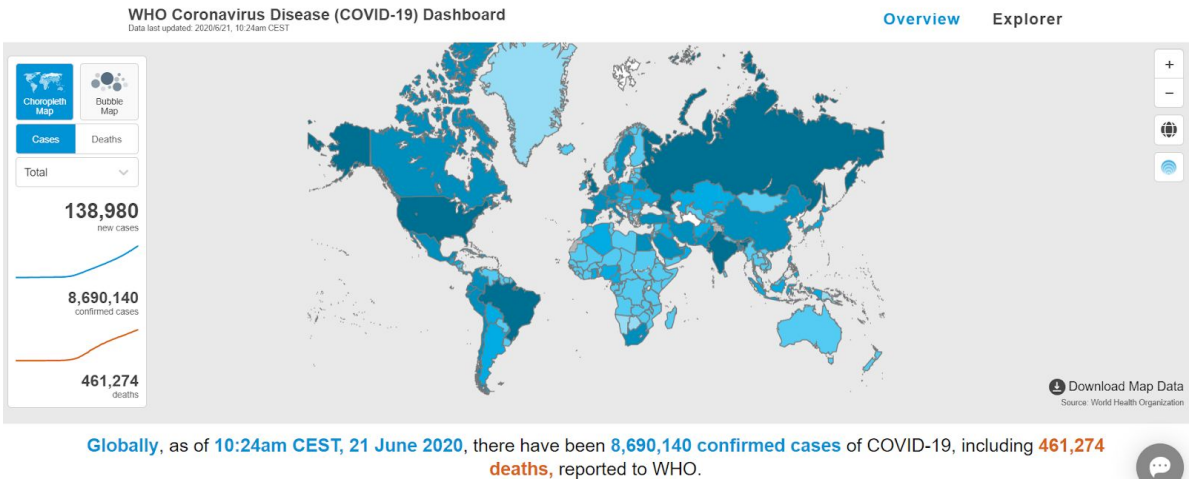
4. Supervised Learning Methodology 11
4.1. Data Pre-processing 11
4.2. Training the Supervised Learning Model 12
4.2.1. Splitting Test and Train data 12
4.2.2. Creating a TF IDF Vector Space 13
4.2.3. Selecting and Training Machine Learning Model - Linear Support Vector Classifier 13
4.2.3.1. Model Selection based on cross validation 13
4.2.3.2. Training the Linear Support Vector Classifier Model 14
4.3. Running the Classification Model 14
4.4. Refinement 14

5. Results 15
5.1. Model Evaluation and Validation 15
5.2. TF IDF Search Engine Evaluation and Validation 16
5.3. Justification 16

5. References 18

1. Definition

1.1. Project Overview



[1]

[1] "(COVID-19) Dashboard." https://covid19.who.int/. Accessed 21 Jun. 2020.

While there has been significant progress in studies of COVID-19, medical research teams have yet to solve some of the key scientific questions addressing the relevant factors related to the pandemic.

The goal of this project is to support the *Call to Action by the White House*[2], in particular to come up with new text and data mining techniques that can easily extract insights to help the research team answer high-priority scientific questions related to COVID-19. This project is derived from the *Kaggle challenge*[3] initiated by the White House, AI2, CZI, MSR, Georgetown and NIH.

The research community has prepared the **COVID-19 Open Research Dataset (CORD-19)** with over 138,000 scholarly articles about COVID-19, SARS-CoV-2, and related coronaviruses.

Based on the database, this project aims to design and implement machine learning models using Natural Language Processing techniques to help organize and generate insights from the overwhelming literature created by scholars each day.

1.2. Problem Statement

With the rapid increase in Coronavirus literature, it has become increasingly difficult for the research community to keep up with the latest information. There are many high-priority questions about COVID-19 that are awaiting to be answered. The task within the scope of the project is to create summary tables that address relevant factors relating to COVID-19.

As part of the Kaggle Challenge, the goal is to uncover information within the **CORD-19** dataset that can support the following research topics:

1. Effectiveness of case isolation/isolation of exposed individuals (i.e. quarantine)
2. Effectiveness of community contact reduction
3. Effectiveness of inter/inner travel restriction
4. Effectiveness of school distancing
5. Effectiveness of workplace distancing
6. Effectiveness of a multifactorial strategy prevent secondary transmission
7. Seasonality of transmission
8. How does temperature and humidity affect the transmission of 2019-nCoV?
9. Significant changes in transmissibility in changing seasons?
10. Effectiveness of personal protective equipment (PPE)

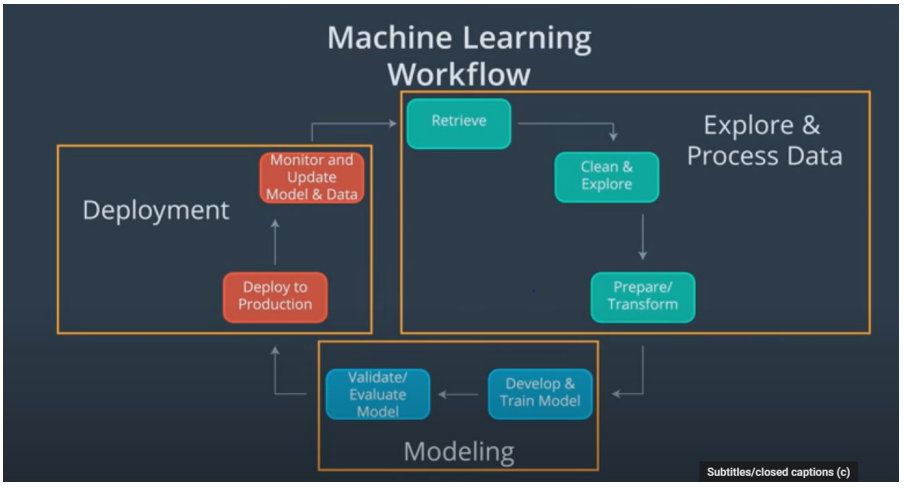
This project provides two software solutions to achieve this goal:

1. **Search Engine** - A benchmark model used to search relevant text within the CORD-19 dataset based on user's query

[2] "Call to Action to the Tech Community on New Machine ...." 16 Mar. 2020, https://www.whitehouse.gov/briefings-statements/call-action-tech-community-new-machine-readable-covid-19-dataset/. Accessed 15 Jun. 2020.
[3] "COVID-19 Open Research Dataset ...." 10 Jun. 2020, https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge. Accessed 15 Jun. 2020.

a. The Search Engine will provide the top N (i.e. 10) most relevant paragraphs that are most relevant to the user's query.
b. While search engines are a great start to dive into the CORD-19 dataset, without any context, there are limitations of the benchmark model to correlate direct questions to relevant information

2. **NLP Classification Model** - A machine learning model that requires training and testing labelled datasets which can be used to categorise new scientific articles
    a. The classification model will determine unlabelled text from the CORD-19 dataset into a division of key research topics
    b. The benefit of the classification model is that it will continuously learn and improve as more data are trained
    c. Having a good machine learning workflow (*Explore & Process data, Modelling, Deployment*[4]) is critical for the model to be effective and up-to-date



Note: CORD-19 has provided labelled data within the buckets of key research topics required for us to create a classification model

The search engine and the NLP classification model have a clear difference when targeting the goal. The search engine identifies paragraphs of relevant information based on the user's **search query** where else the NLP classification model requires **paragraphs within the medical articles** to be fed into the model.

In the next section, we will identify common metrics used as a measurement of performance for each of the solutions mentioned above.

1.3. Metrics

The main criteria for the success of the model is whether or not the model:

- Identifies **most** relevant paragraphs that are labelled from the CORD-19 dataset
- Exclude non-relevant paragraphs

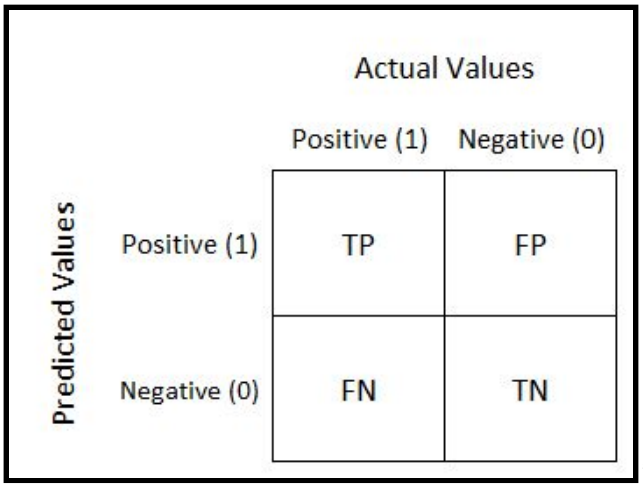
[4] "Become a Machine Learning Engineer - Udacity." https://www.udacity.com/course/machine-learning-engineer-nanodegree--nd009t. Accessed 21 Jun. 2020.

- Ensure overfitting does not occur - model is able to categorise unlabeled data accurately

*Note: There is no **perfect** solution in extracting **all** the relevant paragraphs for each key scientific topic because this is a subjective problem. Hence, the objective of this project is to design an optimised solution / approach rather than a perfect solution.*

The **search engine** and the **NLP classification model** will be evaluated based on the same metrics. The following metrics will be used:

- **Precision -** Can the model correctly classify each paragraph into the correct questions without misjudging irrelevant literatures into the same question
  - $Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$
- **Recall -** Can the model identify all paragraphs that are relevant to the questions without missing any relevant paragraphs within the research literature
  - $Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$



[5]

2. Analysis

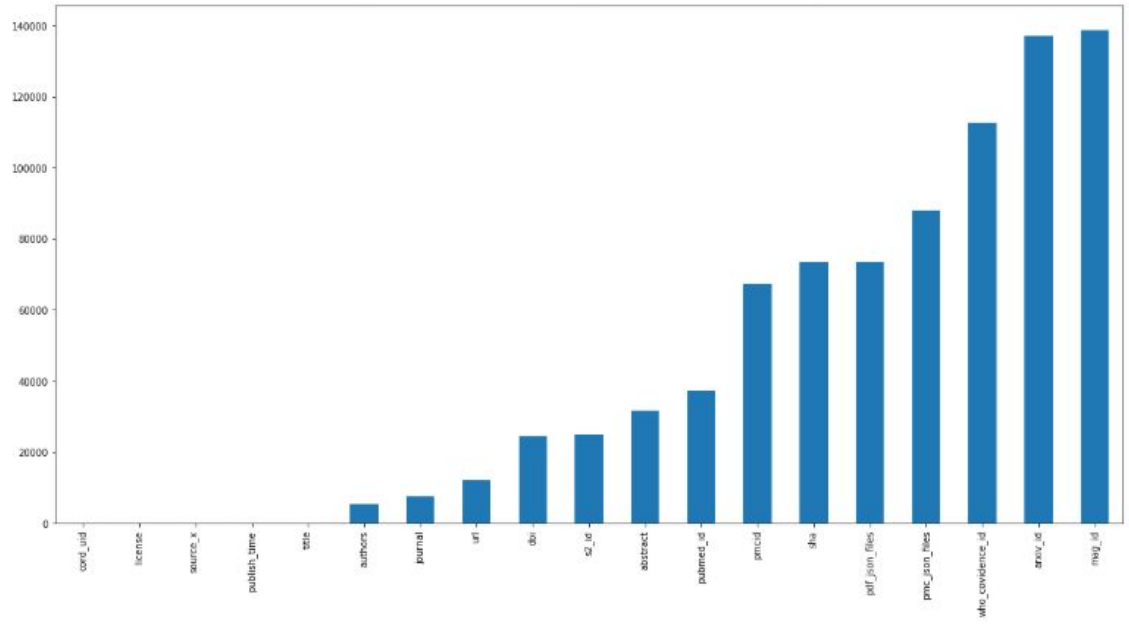
2.1. Data Exploratory with Exploratory Visualization

The first step of the project is to understand the **CORD-19 dataset** and what data is relevant.

2.1.1. Metadata Analysis

The metadata contains information about all medical literatures that are stored within the CORD-19 datasets, including **'title'**, **'abstract'**, **'publish_time'**, **'authors'**, and **'pdf_json_files'**.
*Note: **pdf_json_files** provides the path of the medical literature in json format.*

[5] (n.d.). Understanding Confusion Matrix - Towards Data Science. Retrieved June 15, 2020, from https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62



The figure above shows the number of N/A's of each column.
The **key observation** made from this graph is that not all the medical literature can be searched within the **CORD-19** database.

2.1.2. Medical Articles (PDF_JSONS) Analysis

The medical articles of CORD-19 have been processed into machine readable format (JSON) with the schema included in the **json_schema.txt** file. Based on the requirements of this project, there is a need to break down these medical articles into paragraph level data. These medical articles are extracted:

- Abstract with text
- Body text

2.1.3. Priority Questions (2_relevant_factors) Analysis

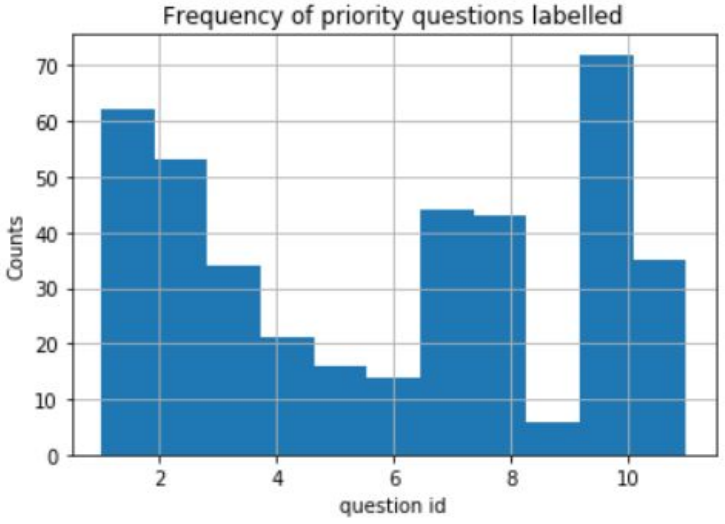
The relevant_factors folder contains a number of csv files with the list of key research topics mentioned in section (1.2. Problem Statement). This is the **labelled table** where we are provided with **'Except'**, which is a segment of the text within the medical journal addressing these key research topics. This dataset is the basis of our training / test data for our NLP classification model.

Some of the key columns include **'Excerpt'**, **'Study'**, **'Study Link'**, '**Factors**', **'Influential'**, '**Measure of Evidence**', and **'Date'**.

- Combining all the key research topics csv files within the folder as one dataframe, while labelling each row of data with their topic
- Data cleansing to address **inconsistent column names** between each key research topics table

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 547 entries, 0 to 58
Data columns (total 3 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Influential        454 non-null    object
 1   Infuential         75 non-null     object
 2   Influential (Y/N)  18 non-null     object
dtypes: object(3)
memory usage: 17.1+ KB
<class 'pandas.core.frame.DataFrame'>
Int64Index: 547 entries, 0 to 58
Data columns (total 2 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Factors            488 non-null    object
 1   Factors Described  59 non-null     object
dtypes: object(2)
memory usage: 12.8+ KB
<class 'pandas.core.frame.DataFrame'>
Int64Index: 547 entries, 0 to 58
Data columns (total 2 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Date            470 non-null    object
 1   Date Published  77 non-null     object
dtypes: object(2)
memory usage: 12.8+ KB
```

- Merging the tables with Metadata in order to identify the pdf location within the CORD-19 dataset
- Removing labelled data without the Metadata path



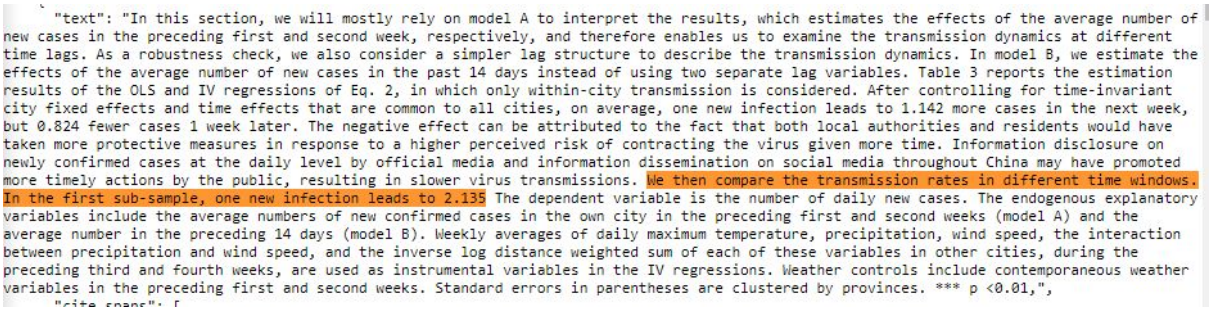
From the figure above, there is an **imbalance of data** labelled for each topic. There are data with as low as only 8 entries with some topics as high as 72 entries.

2.1.4. Other key abnormalities identified

- The Excerpt within the key research topics files does not match with the actual json pdf paragraphs. The example is shown below:

```
In [32]: scoped_categorised_literature.iloc[2]['Excerpt']

Out[32]: 'We then compare the transmission rates in different time windows. In the fir
         st sub-sample, one new infection leads to 2.135 more cases within a week, imp
         lying a fast growth in the number of cases. However, in the second sub-sampl
         e, the effect decreases to 1.077, suggesting that public health measures impo
         sed in late January were effective in limiting a further spread of the viru
         s.'
```

Above figure shows the Excerpt text from the key questions table versus below - actual paragraph within the research paper.



```
"text": "In this section, we will mostly rely on model A to interpret the results, which estimates the effects of the average number of
new cases in the preceding first and second week, respectively, and therefore enables us to examine the transmission dynamics at different
time lags. As a robustness check, we also consider a simpler lag structure to describe the transmission dynamics. In model B, we estimate the
effects of the average number of new cases in the past 14 days instead of using two separate lag variables. Table 3 reports the estimation
results of the OLS and IV regressions of Eq. 2, in which only within-city transmission is considered. After controlling for time-invariant
city fixed effects and time effects that are common to all cities, on average, one new infection leads to 1.142 more cases in the next week,
but 0.824 fewer cases 1 week later. The negative effect can be attributed to the fact that both local authorities and residents would have
taken more protective measures in response to a higher perceived risk of contracting the virus given more time. Information disclosure on
newly confirmed cases at the daily level by official media and information dissemination on social media throughout China may have promoted
more timely actions by the public, resulting in slower virus transmissions. We then compare the transmission rates in different time windows.
In the first sub-sample, one new infection leads to 2.135 The dependent variable is the number of daily new cases. The endogenous explanatory
variables include the average numbers of new confirmed cases in the own city in the preceding first and second weeks (model A) and the
average number in the preceding 14 days (model B). Weekly averages of daily maximum temperature, precipitation, wind speed, the interaction
between precipitation and wind speed, and the inverse log distance weighted sum of each of these variables in other cities, during the
preceding third and fourth weeks, are used as instrumental variables in the IV regressions. Weather controls include contemporaneous weather
variables in the preceding first and second weeks. Standard errors in parentheses are clustered by provinces. *** p <0.01,",
```

- Key observation that can be made from here is that the JSON files within the **CORD-19 data** are out of date or are **not extracted properly**.

2.1.5. Key Outcome from Data Exploratory

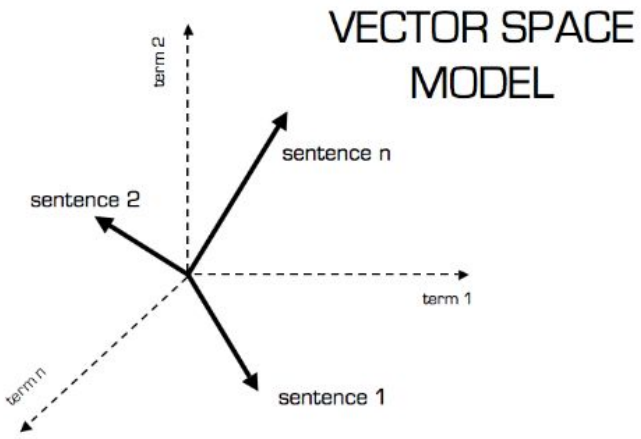
From the Data Exploratory step, the following two pickle output files have been processed for the machine learning model.

- **1_scoped_cat_lit.pkl** - After data cleansing, there are **400 labelled data** categorised into **11 key topics**. There is a data imbalance between each topic referred to here.
    - **Excerpt** - exact phrase extracted to be relevant to the key question topics
    - **research_topic** - actual key question identified from 1.2. Problem Statement
    - **topic_idx** - key question index used to differentiate different topics
    - **pdf_json_file** - location of where the text is extract from
    - **Factors** - elements that contributes to the reasoning behind this text
    - **Date** - Date of publish
    - Other related columns that are not used in this project
- **2_extracted_literature_data** - paragraph level data extracted from all the medical literatures within the **1_scoped_cat_lit.pkl** file. There are **145 documents** that are referenced with **6496 paragraphs** extracted from the medical literature.
    - **Extract_id** - unique id to each paragraph
    - **Json_path** - literature document path where the paragraph was collected
    - **Section** - section of the document the paragraph was extracted
    - **Text** - the actual paragraph

2.2. Algorithms and Techniques

2.2.1. Word Vectorizers - aka one hot encoding

- Machines do not understand text data directly. Therefore our text data requires some sort of encoding and conversion into number vectors which the machine understands.



Vector Space Model 6

2.2.1.1. Countvectorizer

- A **Countvectorizer** creates a vector space of all the words that has been provided and prioritise them based on the number of frequency of the word that it has appeared throughout the whole data
- Since there are millions of possibilities of words that a text could contain, we limit the size of the vector into a limited range (i.e. 5000) that is within a machine's capability to process. This will take in the most frequent 5000 words as features
- Additional features such as 'no word' and 'in-frequent word', which takes in account all the words that are left out from the 5000 features that were filtered out.
- The **limitation** of a count vectorizer is that it could have biasing factors where high frequency of the word appearing could be contained in a single document - causing the word to be scaled up when it only appeared in that one single document.

2.2.1.2. TF-IDF Vectorizer

- **TF-IDF** stands for **Term Frequency - Inverse Document Frequency**. It is an encoding technique used to perform **weights** on words based on the elements below.
- **Term Frequency** - Frequency of a word in a single document
    - Since frequency of the word is highly dependant on the length of the document, we would **normalize** the frequency of a word per document dividing by the total length of the document
    - $tf(word,\ document)\ =\ \#\ of\ word\ in\ document\ /\ total\ words\ in\ document$

[6] "Machine Learning :: Cosine Similarity for Vector Space ...." 12 Sep. 2013, http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/. Accessed 23 Jun. 2020.

- **Document Frequency** - Number of documents in which the word is present
    - Measures the importance of document in the whole set of corpus
    - $df(word,\ all\ documents)\ =\ \#\ of\ documents\ that\ contains\ the\ word$
- **Inverse Document Frequency** - inverse of the Document Frequency
    - Measure the **informativeness** of the word
    - More occurring words will be less weighting, while words are weighed up if they are more distinct and unique to the topic
    - $idf(word,\ all\ documents)\ =\ \#\ of\ all\ documents\ /\ df(word,\ all\ documents)$
- **TF-IDF** - Multiplying the Term Frequency and Inverse Document Frequency
- $tfidf(word,\ document,\ all\ documents)\ =\ tf(word,\ document)\ \times\ log(idf(word,\ all\ documents))$

2.2.2. Search Engine Match Algorithm

- **Cosine similarity** allows the comparison of two vectors disregarding the magnitude (size). This is useful when comparing two texts that have very different lengths
- This will be used within the search engine to find the top most relevant documents relating to the question that the user queries.

$$cos\ \theta\ =\ \frac{\bar{a}\cdot\bar{b}}{||a||\cdot||b||}$$

2.2.3. Supervised Machine Learning model

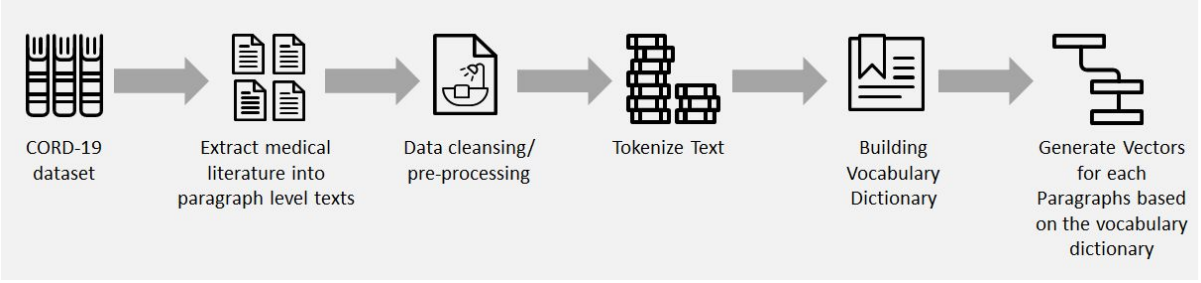
The model algorithms that will be used in this projects are:

- Linear SVM
- Random Forest Classifier
- Naive Bayes Classifier
- Logistic Regression

*For the scope of the project, we will perform **cross validation** on the 4 models and only perform evaluation on the model with the highest accuracy.*

3. Benchmark Model - TF-IDF Search Engine

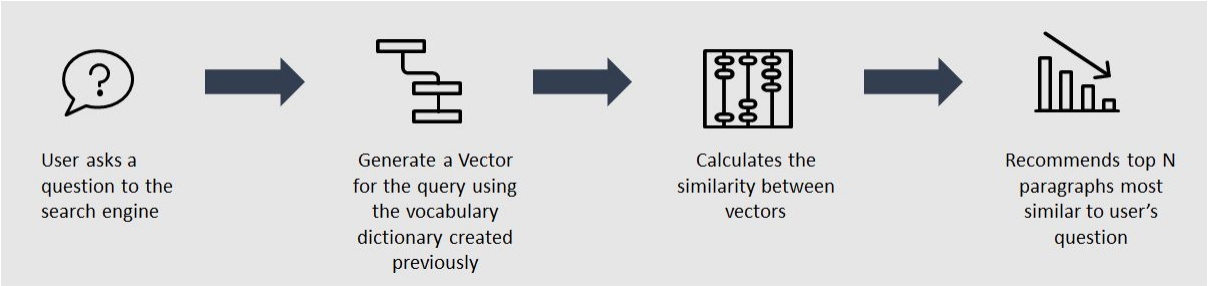
3.1. Creating the TF-IDF Vector Space Model



- The first step of creating the search engine is extracting medical literatures into paragraph level texts

- Then using the Data-preprocessing technique we are able to cleanse and tokenize the text into single words that will be used to create a Vocabulary Dictionary
- This Vocabulary Dictionary uses TF-IDF Vectorizer
- From the TFIDF Vectorizer, we will generate unique vectors for each paragraph which will be used to compare our user's input.

3.2. Running the Search Engine



- When a user queries a question to the Search Engine, the code first performs data pre-processing technique to cleanse and tokenize the input text.
- It is then transformed using the TFIDF Vectorizer created above into an unique vector
- Cosine Similarity algorithm is used to compare a user's query with all unique paragraphs that has been generated previously.
- The code will generate the top N results of paragraphs that are most similar to the question asked by the user.

3.3. Evaluation metrics for Search Engine

- Since the mechanism of the search engine is a little different to a classification model, the following criteria has been used to match the metrics between the benchmark model and the classification model

4. Supervised Learning Methodology

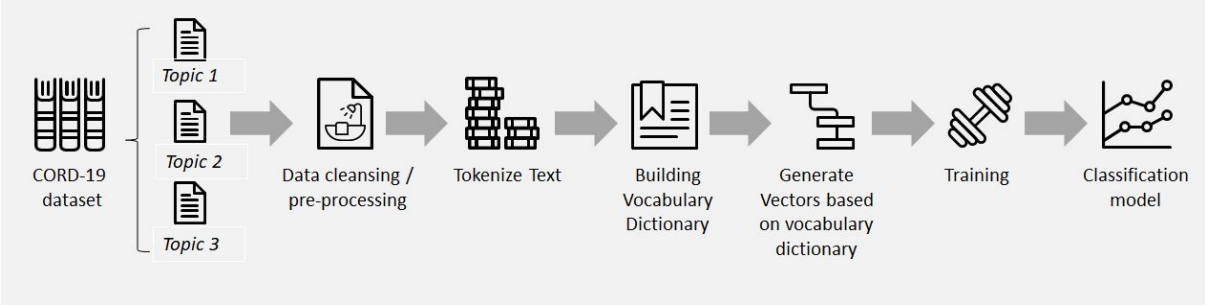
4.1. Data Pre-processing

```
def preprocess(inputText):
    #define stopwords
    stop_words = set(stopwords.words("english"))
    #lower case the text
    outputText = inputText.lower()
    #Convrt percentages into the string percent
    outputText = re.sub('(\\d+%)', 'percent', outputText)
    # Remove special characters and digits
    outputText=re.sub("(\\d|\\W)+"," ",outputText)
    # Tokenisation
    outputText = outputText.split()
    # Remove Stop Words
    outputText = [word for word in outputText if not word in stop_words]
    # Stemming
    ps=PorterStemmer()
    outputText = [ps.stem(word) for word in outputText]
    # Lemmatisation
    lem = WordNetLemmatizer()
    outputText = [lem.lemmatize(word) for word in outputText]
    outputText = " ".join(outputText)

    return outputText
```

All text data will need to be **cleansed** and **normalized** before vectorizing our text. This is to normalize different ways of saying the same word into one. This will ensure that the features that are extracted are more focused and more relevant to the classification and the search engine. The following steps are performed on the text data:

1. Lower case the text
2. Convert percentages to the word ‘percent’
3. Removing any special characters other than digits and word
4. Removal of Stop words

4.2. Training the Supervised Learning Model



The Supervised Machine Learning Model uses the labelled dataset in **1_scoped_cat_lit.pkl** from the data exploratory section with **400 trained texts** in **11 key topics**. Since the data is very limited in size, this will affect the performance of the model’s output.

4.2.1. Splitting Test and Train data

We have split the data into 80% train and 20% testing. Due to the imbalance of data mentioned in the previous section, we will see that some categories will have very low performance compared to other topics.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
```

4.2.2. Creating a TF IDF Vector Space

Similar to the Search Engine, we will be building a TF IDF Vector Space to vectorize our train data. However, for this vector space, we will only build it through our **training data**.

```
In [9]: vectorizer = TfidfVectorizer()
        features = vectorizer.fit_transform(X_train).toarray()
        labels = y_train
        features.shape

        # Each of 320 of train paragraphs is represented by 909 features, representing the tf-idf score
Out[9]: (320, 909)
```

4.2.3. Selecting and Training Machine Learning Model - Linear Support Vector Classifier

Once the Train data has been vectorized, we can simply select a model and train our model.

4.2.3.1. Model Selection based on cross validation

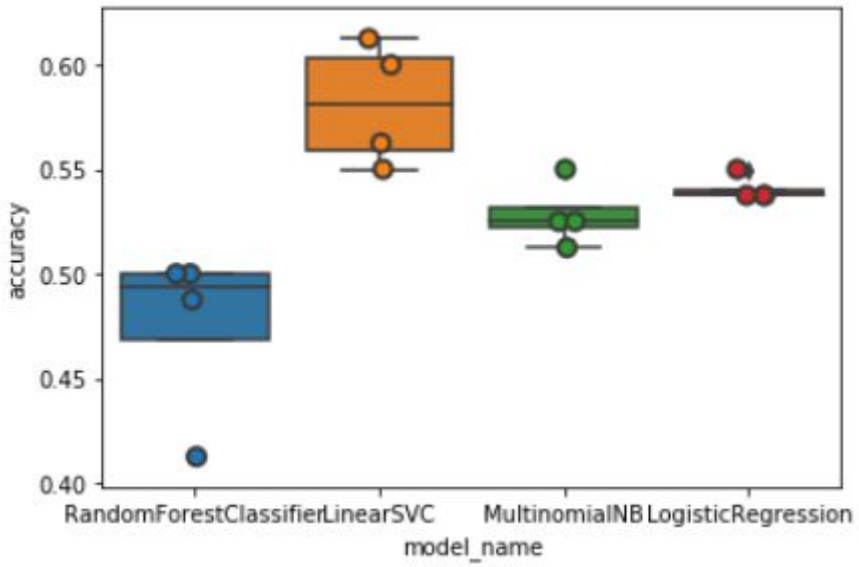
Firstly, I have initialized and chosen 4 popular classifiers from sklearn:

- Naive Bayes Classifier
- Logistic Regression Model
- Random Forest Classifier
- Support Vector Classifier.

Using the **cross_val_score** functionality in Sklearn, I will be performing an evaluation of the Linear SVC model and perform a final evaluation with the TF IDF Search Engine.

```
# Naive Bayes Classifier
from sklearn.naive_bayes import MultinomialNB

# Logistic Regression Model
from sklearn.linear_model import LogisticRegression

# Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier

# support Vector Classifier
from sklearn.svm import LinearSVC

# Cross validation library
from sklearn.model_selection import cross_val_score
```

```
model_name
LinearSVC                 0.575000
LogisticRegression        0.546875
MultinomialNB             0.546875
RandomForestClassifier    0.521875
Name: accuracy, dtype: float64
```

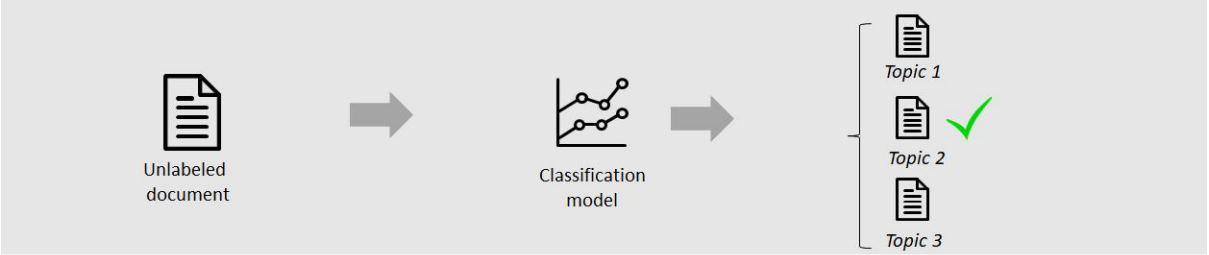


4.2.3.2. Training the Linear Support Vector Classifier Model

Training the model is as simple as executing the 'fit' function after creating the model object.

```
model = LinearSVC()

X_train, X_test, y_train, y_test, indices_train, indices_test = train_test_split(features, labels, X_train.index, test_size=0.3,
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

4.3. Running the Classification Model



To run the classification model, we can use the 'predict' function from the model object. This allows the user to feed in unlabelled documents and categorise them under each topic for COVID-19.

4.4. Refinement

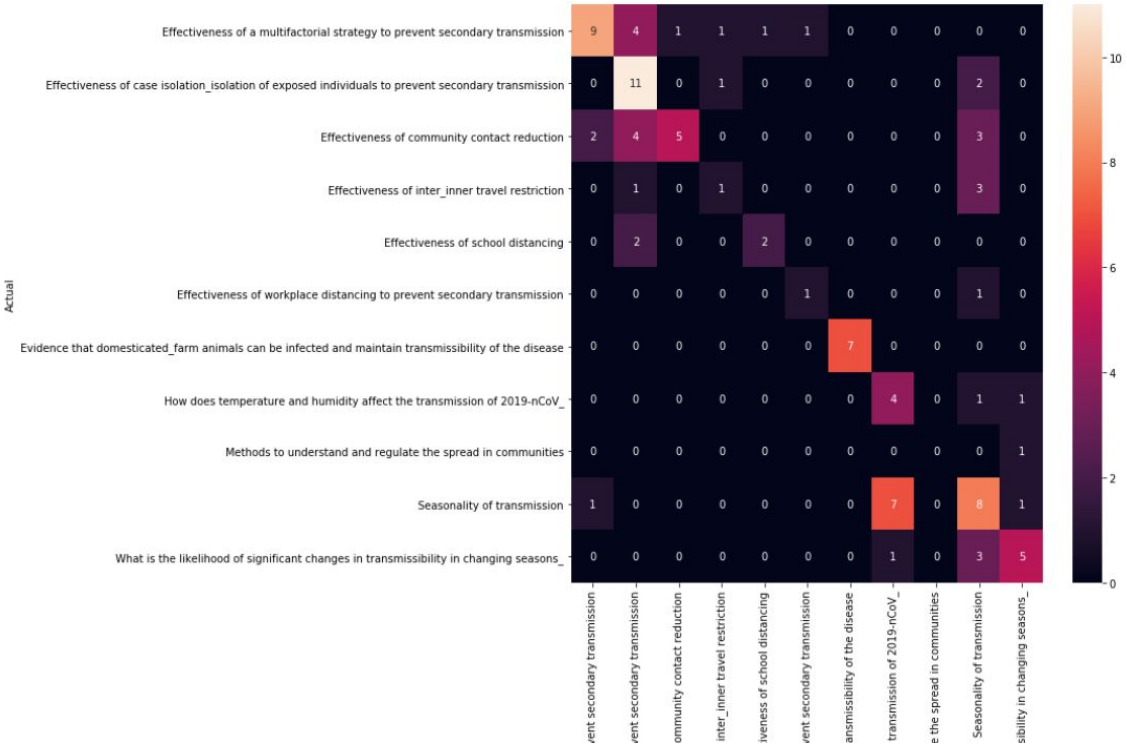
In this project we have compared 4 different kinds of models using cross validation technique to select the optimal model. However, to refine the solution even more, there are some problems that would be good to solve:

- **Class imbalancing** - there are classes with a lot less data than some topics with a lot of labelled data. We could use re-sampling technique to up-sample data for topics with less data
- **Obtain more labelled data from the institute** - more data would definitely help with the progress and improvement of the model

- **Feature Engineering** - Experiment with more features and NLP techniques.

5. Results

5.1. Model Evaluation and Validation



Topic ID	Topic name	precision	recall	f1-score	support
1	Effectiveness of a multifactorial strategy to prevent secondary transmission	0.75	0.53	0.62	17
2	Effectiveness of case isolation_isolation of exposed individuals to prevent secondary transmission	0.50	0.79	0.61	14
3	Effectiveness of community contact reduction	0.83	0.36	0.50	14
4	Effectiveness of inter_inner travel restriction	0.33	0.20	0.25	5
5	Effectiveness of school distancing	0.67	0.50	0.57	4
6	Effectiveness of workplace distancing to prevent secondary	0.50	0.50	0.50	2

	transmission				
7	Evidence that domesticated_farm animals can be infected and maintain transmissibility of the disease	1.00	1.00	1.00	7
8	How does temperature and humidity affect the transmission of 2019-nCoV_	0.33	0.67	0.44	6
9	Methods to understand and regulate the spread in communities	0.00	0.00	0.00	1
10	Seasonality of transmission	0.38	0.47	0.42	17
11	What is the likelihood of significant changes in transmissibility in changing seasons_	0.62	0.56	0.59	9

Accuracy: 0.575
Weighted average Precision: 0.54
Weighted average Recall: 0.58
Weighted average F1-score: 0.55
Test data provided: 96 entries

5.2. TF IDF Search Engine Evaluation and Validation

Due to the difference in nature between a search engine and classification model, the following expectation is set so that accuracy can be measured from the search engine:

- User asks the search engine the 11 key scientific questions that this kaggle challenge is focusing on
- The search engine will provide the top 5 most relevant paragraphs based on TF IDF algorithm (with an internal search engine index)
- In parallel, run the search engine with all the labelled data, and record their search engine indexes
- Now, compare the top 5 indexes of the key scientific topic to the indexes from the labelled data
- If there is a match in any of the top 5 indexes, then the question is considered as a match
- Finally, to obtain the accuracy, we calculate the number of matches for each scientific topic to the total amount of topics asked to the search engine.

```
The accuracy of the Search Engine based on the above criteria is:  2 / 11 = 0.182
```

5.3. Justification

In this project, we have successfully implemented two different NLP solutions towards answering high-priority scientific questions related to COVID-19. Based on the results, both solutions have accuracies that are not yet practical for the researchers to use. With the accuracy of the classification model being 58% and the accuracy of the search engine of 18%.

From the search engine's perspective, it is very limited in the contextual information as the search engine developed is based solely on word weighting and matches. Since the queries and the key questions are very short in length, the vectors generated will be quite vague and hence the paragraphs that are matching may not be relevant to the topic.

From the classification's model perspective, there is definitely a good opportunity to refine the model. There was a limitation on the labelled data with only 400 text extractions within 11 key topics to classify. There are also class imbalances which I've identified.
To continue to refine and make the model perform better, more feature engineering techniques can be developed such as:

- Keywords , Entities, Synonyms
- Combining contextual phrases such as the Title of the Document, freshness (Date), location, and key findings
- Obtain more labelled data

5. References

Allen Institute For AI, 2020, COVID-19 Open Research Dataset Challenge (CORD-19), Available at: https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge [Accessed 15/06/2020]

Office of Science and Technology Policy, 2020, Available at: https://www.whitehouse.gov/briefings-statements/call-action-tech-community-new-machine-readable-covid-19-dataset/ [Accessed 15/06/2020]

Google, 2019, How Google Search Works, Available at: https://www.youtube.com/watch?v=0eKVizvYSUQ [Accessed 15/06/2020]

Sarang Narkhede, 2018, Understanding Confusion Matrix, Available at: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62 [Accessed 15/06/2020]

Google News, Coronavirus (COVID-19), Available at: https://news.google.com/covid19/map?hl=en-AU&mid=/m/0chghy&gl=AU&ceid=AU:en [Accessed 15/06/2020]

Silvio Mori Neto, 2020, Capstone Project Proposal, Available at: https://github.com/silviomori/udacity-machine-learning-capstone-starbucks/blob/master/proposal.pdf [Accessed 15/06/2020]

Moghazy, 2020, COVID-19 Literature Ranking + Web Scraping, Available at: https://www.kaggle.com/moghazy/covid-19-literature-ranking-web-scraping [Accessed 15/06/2020]

World Health Organisation, 2020, WHO Coronavirus Disease (COVID-19) Dashboard, Available at: https://covid19.who.int/ [Accessed 21/06/2020]

Semantics Scholar, 2020,CORD-19 COVID-19 Open Research Dataset, Available at: https://www.semanticscholar.org/cord19 [Accessed 20/06/2020]

Kartheek Akella, 2020, Implementing the TF-IDF Search Engine, Available at: https://medium.com/analytics-vidhya/implementing-the-tf-idf-search-engine-5e9a42b1d30b [Accessed 22/06/2020]

Zayed Rais, 2020, Build your semantic document search engine with TF-IDF and Google-USE, Available at: https://medium.com/@zayedrais/build-your-semantic-document-search-engine-with-tf-idf-and-google-use-c836bf5f27fb [Accessed 23/06/2020]

Washington University, 2020, Project 3, part 2: Searching using TF-IDF, Available at: https://courses.cs.washington.edu/courses/cse373/17au/project3/project3-2.html [Accessed 23/06/2020]

Praveen Dubey, 2018, An introduction to Bag of Words and how to code it in Python for NLP, Available at: https://www.freecodecamp.org/news/an-introduction-to-bag-of-words-and-how-to-code-it-in-python-for-nlp-282e87a9da04/#:~:text=Bag%20of%20Words%20(BOW)%20is,documents%20in%20the%20training%20set. [Accessed 23/06/2020]

Susan Li, 2018, Multi-Class Text Classification with Scikit-Learn, Available at: https://towardsdatascience.com/multi-class-text-classification-with-scikit-learn-12f1e60e0a9f http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/ [Accessed 23/06/2020]