DM882 – Text Mining

Document Classification and Subsequent Analysis of COVID-19 in Scientific Publications, Spring 2021

Md Shihab Ullah
mdull20@student.sdu.dk

Department of Mathematics and Computer Science
Faculty of Science, University of Southern Denmark, Campus Odense

Contents

1 Introduction

In this project, I have tried to find out whether the articles available in one of the recommended dataset comprising scientific publications are COVID-19 related or not using Naive Bayes classifier.

Various utility steps, function and core methods undertaken to achieve the tasks are mentioned and coded both with comments and required documentation in the submitted Jupyter notebook file.

In this report, I will be mentioning the problems faced while working in the projects along with the reasoning behind the solution of those problems respectively.

2 Document Classification Process

Each cell of jupyter notebook if run sequentially will give us both document classification to find out COVID-19 related articles and how accurate the classification were along with NER visualization and analysis respectively.
As each text normalization and classification steps are discussed in the notebook file, here I present a generic flowchart of the overal document classification steps which is shown in the image as follows:

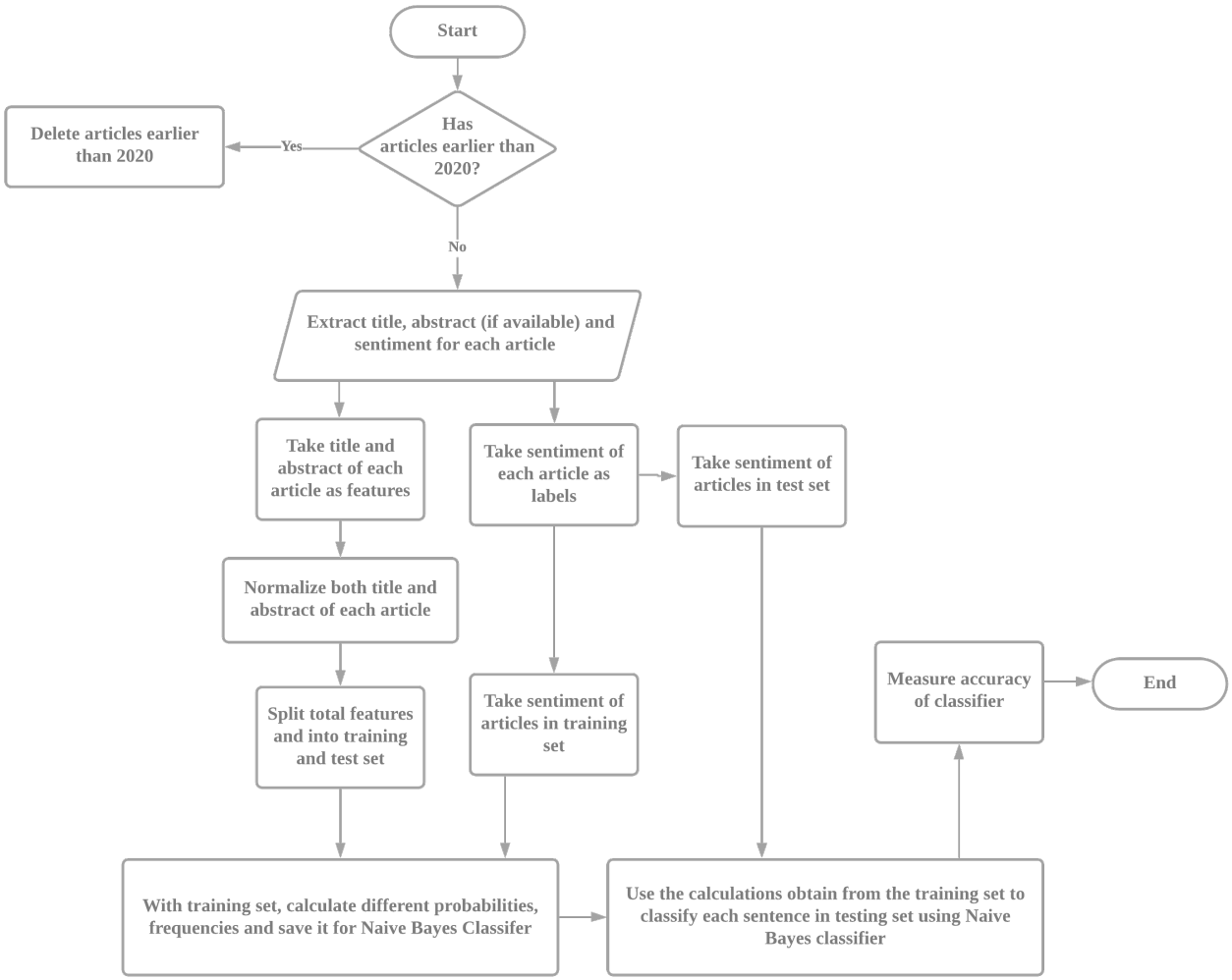

Figure 1: Flowchart of Document Classification using Naive Bayes Text Classifier

3 Experimental Outcome

Using the classifier, I have tried to estimate the proportion of papers related to COVID-19 from a subset of 17.376 articles present in the dataset named "non_comm_use.I-N.xml" :
- As proportion of all papers published in and after 1st January,2020 in the subset you downloaded.
- As proportion of articles in two journals named "Lancet" and "Ind_Health" Nature) published in and after 1st January,2020.

The reason behind not stricting articles till the end of 2020 is that there might be many COVID-19 related article after that timeline which can let the classifier learn more.

Dataset	Journal	Article	Accuracy
non_comm_use.I-N.xml	All	All	93.9%
non_comm_use.I-N.xml	Lancet, Ind_Health	All	95.9%
non_comm_use.I-N.xml	Lung	All	66.7%

Table 1: Accuracy table of the Document Classification using Naive Bayes Classifier

4 Most frequent Named Entity Visualization

One of the major task was to find and analysis the most commonly mentioned Named Entities with respect to COVID-19 related articles. For this reason, I tried to find all the Named Entities and its type using one of the ScispaCy NER models which is trained on the BC5CDR corpus. It is used to find two entity types: a) Disease and b) Chemical.

The following table shows top 10 most common Named Entities along with its type for journal named "Lung", set of journals ["Lancet" and "Ind_Health"] and last but not the least, the complete subset of "non_comm_use.I-N.xml" dataset:

Lung	Lancet, Ind_Health	non_comm_use.I-N.xml
('COVID-19', 'CHEMICAL')	('COVID-19', 'CHEMICAL')	('COVID-19', 'CHEMICAL')
('cough', 'DISEASE')	('infection', 'DISEASE')	('infection', 'DISEASE')
('CHF', 'DISEASE')	('cancer', 'DISEASE')	('pandemic', 'CHEMICAL')
('exertional dyspnea', 'DISEASE')	('SARS', 'DISEASE')	('infections', 'DISEASE')
('Cheyne-Stokes', 'DISEASE')	('insomnia', 'DISEASE')	('SARS', 'DISEASE')
('±', 'CHEMICAL')	('pandemic', 'DISEASE')	('fever', 'DISEASE')
('PAH', 'CHEMICAL')	('pain', 'DISEASE')	('coronavirus disease 2019', 'DISEASE')
('acute infection', 'DISEASE')	('infectious diseases', 'DISEASE')	('pneumonia', 'DISEASE')
('AHRF', 'DISEASE')	('exacerbations', 'DISEASE')	('SARS-CoV-2', 'CHEMICAL')
('HFOV', 'CHEMICAL')	('pneumonia', 'DISEASE')	('death', 'DISEASE')

Table 2: Top Ten Most Common used Entity Name and Type

There are two types of visualization used for demonstration purpose which are as follows:

Bar Chart: It shows the frequency of each entity.
Pie Chart: It shows the percentage of its frequency against the total frequency of top ten most common entities.

Here, I demonstrate the 10 most frequent Named Entity sequentially for journal named "Lung", set of journals ["Lancet" and "Ind_Health"] and last but not the least, the complete subset of "non_comm_use.I-N.xml" dataset respectively.

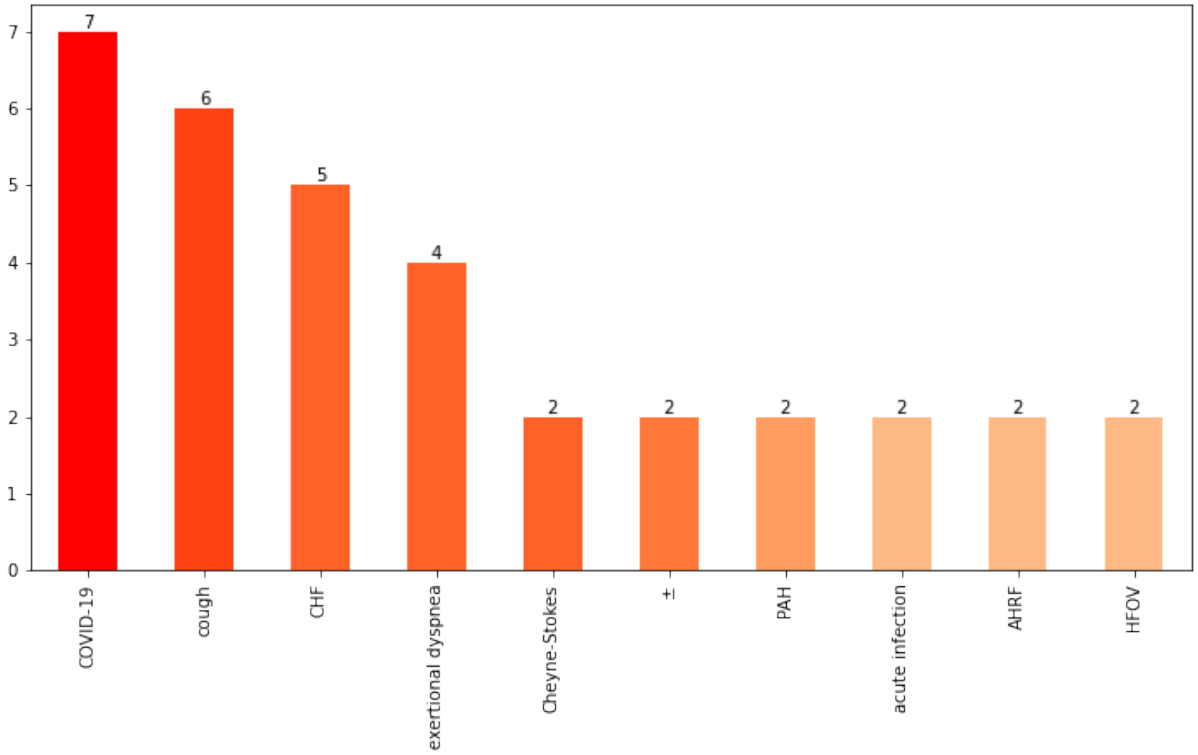

Figure 2: Top ten most frequent Named Entities in "Lung" Journal

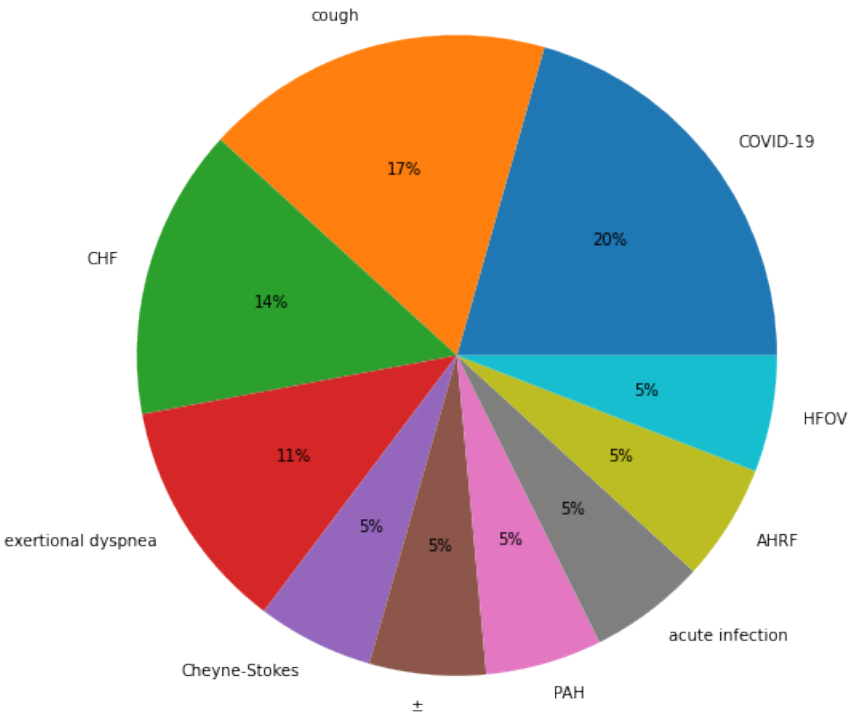

Figure 3: Percentage of each most frequent Named Entities in "Lung" Journal

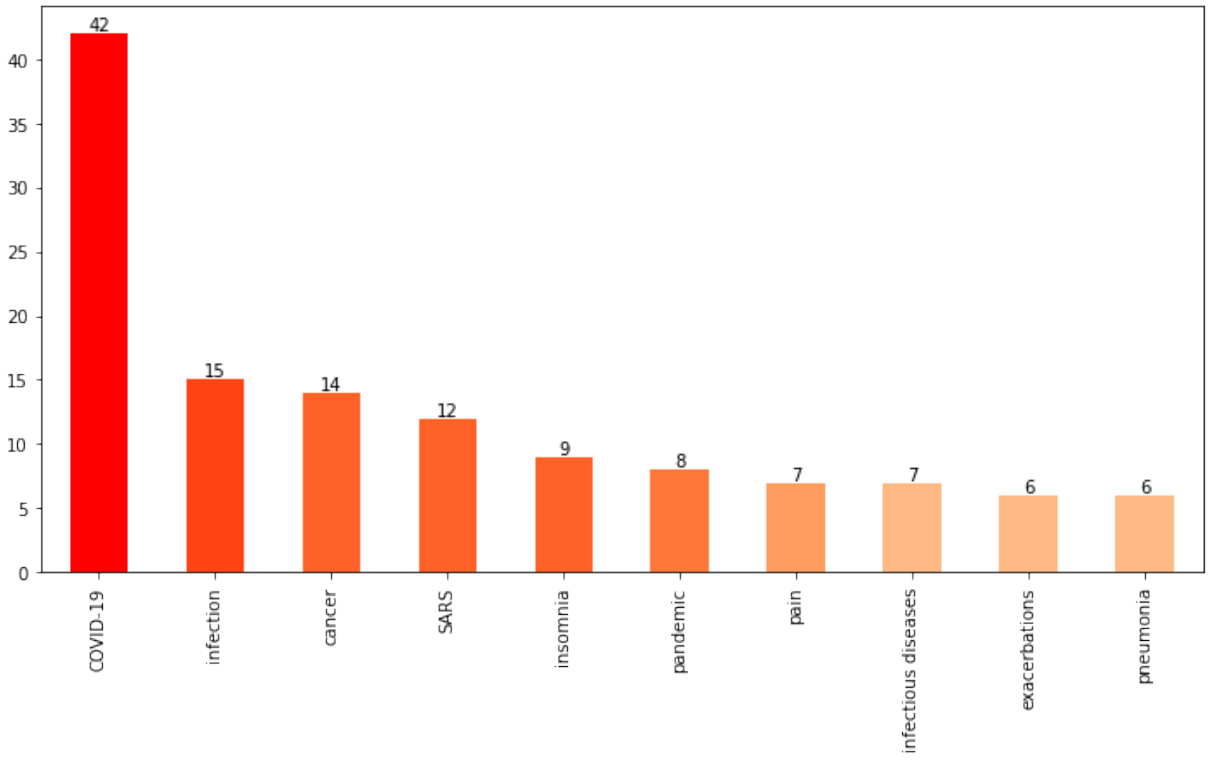

Figure 4: Top ten most frequent Named Entities in both "Lancet" and "Ind_Health" Journals

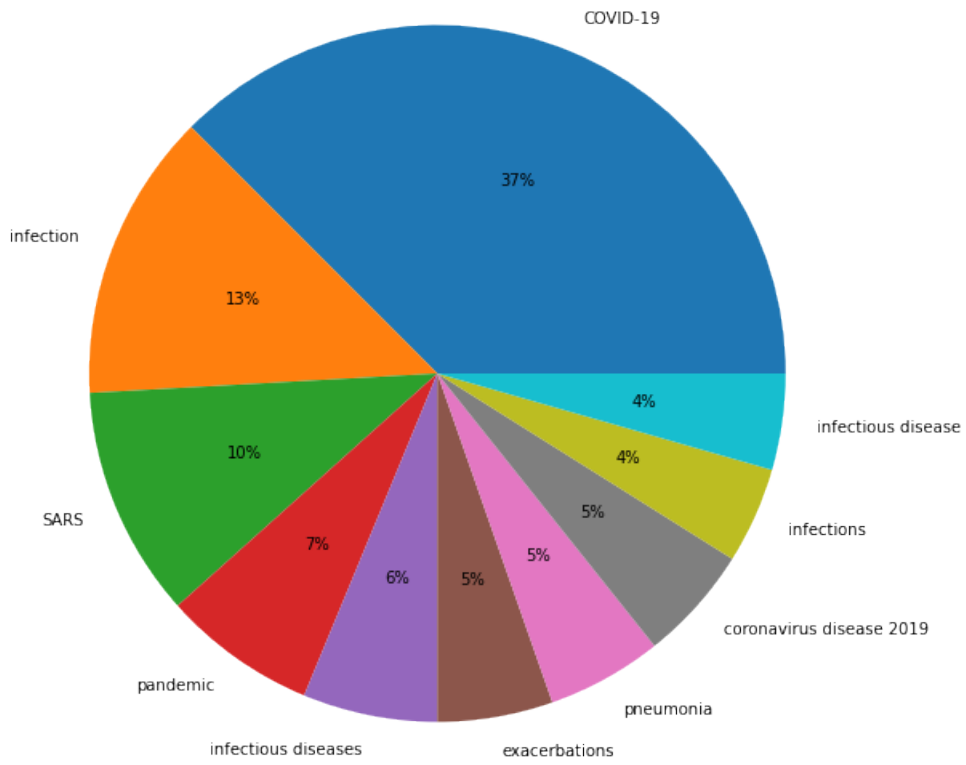

Figure 5: Percentage of each most frequent Named Entities in both "Lancet" and "Ind_Health" Journals

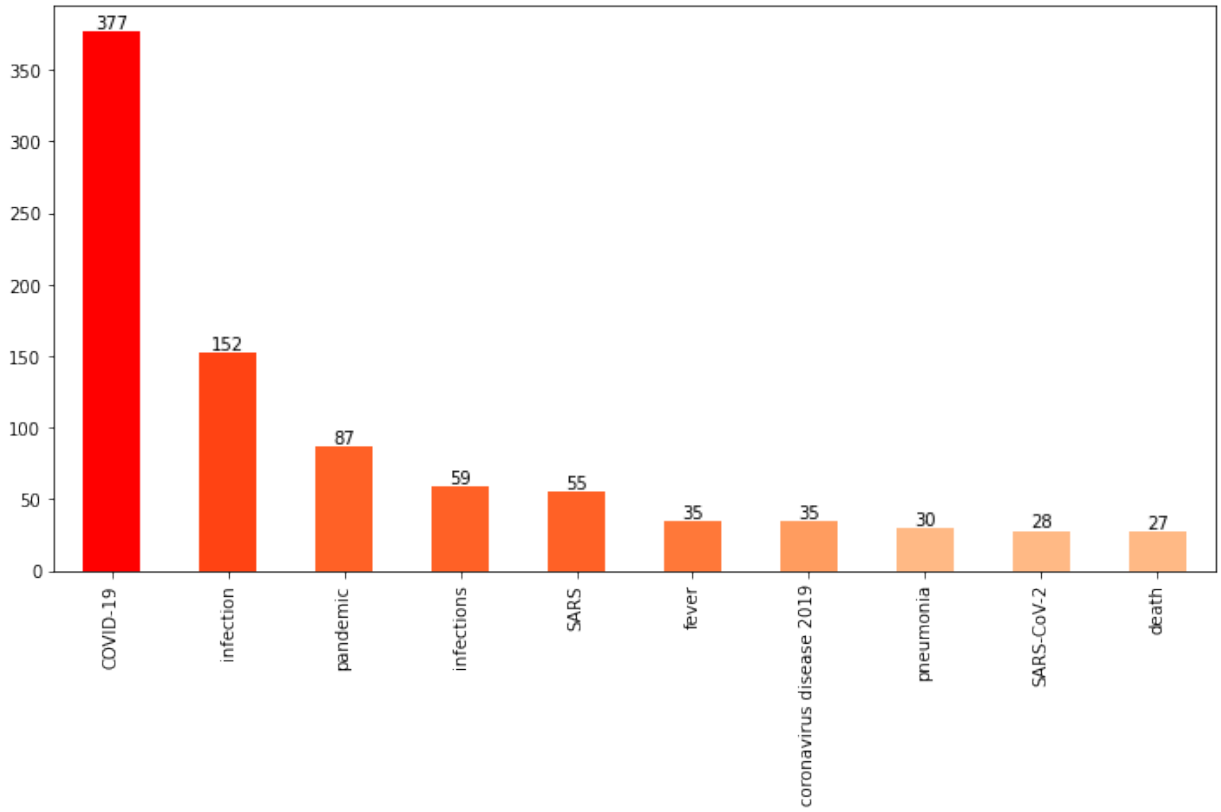

Figure 6: Top ten most frequent Named Entities in all Journals of thesubset

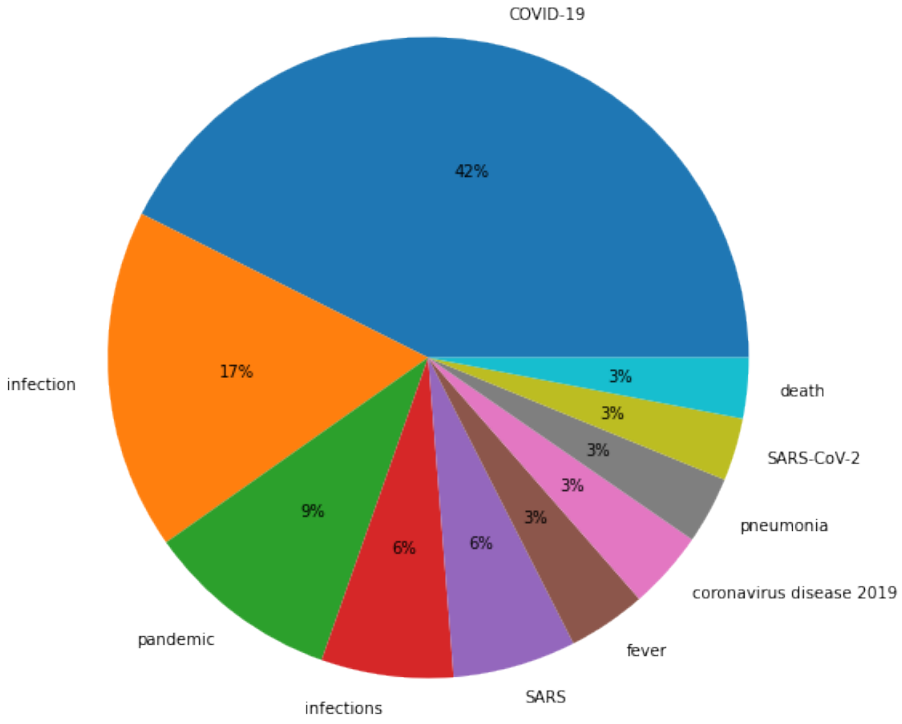

Figure 7: Percentage of each most frequent Named Entities in all Journals of the subset

5 Problems and Solutions

While doing the project I have sequentially stepped upon some problems which are mentioned as follows along with narration about how I approached to solve them:

5.1 Boilerplate utility coding

There were needed to write unnecessary boilerplate code. For example: File extraction, DOM element traversal in .nxml files, spliting dataset exclusively into training and testing dataset, doing simple text processing operation like removing stopwords, remaining punctuations, getting the base form of the words to commonize words i.e. lemmatization, utitity functions to shape and reshape data, doing arithmetic operation etc.

As a result, I took the liberty to import and use popular python package **at minimum level and only in obvious reasons.** For example: os, datetime, xml.dom.minidom, sklearn, numpy, pandas, itertools, counter etc.

5.2 Unnecessary and noise data

As per project description, in case of dataset classification, it was instructed to consider all the articles of all journal released in 2020 or the proportion of articles in any specified journal released in 2020 respectively. I have found that there were many articles prior to 2020. Thus, processing, normalizing and spliting them to train classifier seemed unnecessary and just noise to training data of the classifier.

As a result, I did a one-time operation to remove all the articles in all journal of the used dataset which has been prior of 1st January, 2020 and then start extraction of title and abstract for the remaining articles.

5.3 Limitation of computational resource

Even though we were allowed to constraint our tasks to the archive files named "comm_use.I-N.xml.tar.gz" and "non_comm_use.I-N.xml.tar.gz", both of them have total size of 17 GB approximately. Given my laptop configuration (Intel Core i5 1.90 Ghz and 8 GB RAM) and despite the one-time delete operation of avoidable articles in different jour-

nal, it was not feasible to start the intital text preprocessing tasks by iterating through all the thousands of articles in hundreds of journal on the dataset.

As a result, I choose to use the smaller XML archive file i.e. "non_comm_use.I-N.xml.tar.gz" and used a subset/ chunk of it comprising over 17,000 articles in 270+ journals respectively. But my code is immune to expontentially very large dataset, offcourse if it is run on computers with better configuration [1].

5.4 Lack of rigid structure of XML based on journals

After scanning and even skimming some of the articles in the used dataset from the project description, I have found that in many cases there are no abstract and if they are present, they are mostly located in XML tags named "journal-title" or "abstract-title". There were many child tags under them which had to be consider to avoid as many of them were not of TEXT_NODE type.

As a result, I wrote two utility functions where get_Text(nodelist) takes the abstract related XML tags and traverse through the child nodes and send it to get_all_text(node) for recursively getting the text content from the child node if the child node is of TEXT_NODE type only. After the dataset speculation, I found out that XML tag named "article-title" always contained the title of the article. Thus, I always try to extract title from that tag assuming its there.

5.5 Gathering bio-medical COVID-19 related keywords

There are no labelling or pre-stated knowledge for any article to understand whether they are COVID-19 related or not. Also it is very time consuming to scan through all the articles to create custom COVID-19 vocabulary. Also being a non-biomedical student, it is difficult to know or get relevant medical keywords related to COVID-19 so that all the articles can be labelled by it.

As a result, I tried to do web-search for finding both medical and non-medical COVID-19 keywords and comprised a custom list of keywords for labeling articles by restricting the domain of keywords to only core COVID-19 related as much possible. If there are any keywords present in either text or abstract of any article, I labelled the article as 1 i.e. Positive and vice-versa [2] [3] [4] [5].

5.6 Choosing reusability over static normalization pipeline

I wanted to make a reusable text normalization pipeline where the methods are flexible to be used individually and also be used to create custom flow of text normalization.

Hence, I created a Text Normalizer class to containing most common text normalization processes and then created a seperate custom pipeline named text_normalize(sentence) by picking up methods from the Text Normalizer class as I felt necessary.

5.7 Selecting the necessary steps

There are many steps we can make sentence undergo for doing the text normalization. But I have to consider which steps were actually beneficial for the task of document classification. Some of the concerns are mentioned as follows:

1) I did not perform any dedicated cleaning operation because scientific publication often contains clean words, very fewer spelling mistakes, contracted words such as "could've", "won't", "must've" etc. Still I have added many of them in stopwords just in case.

2) NLTK library has a small set of stopwords which I thought is insufficient for appropriate coverage in stop words detection. Also there are some unique stop words which can be found in both the used XML and other scientific publication in general. For e.g: "study", "journal", "statistic", "analysis", "group", "", "̂" etc. As a result, I have curated another custom list of stopwords and combined them with the default NLTK stopwords to ensure more efficient stopwords removal [6].

5.8 Dilemma in normalization steps

One of the toughest dilemma was whether to choose lemmatization or stemming and if both then in which order the text should be processed:

5.8.1 Scenario 1:

If I do stemming(Porter Stemming) before lemmatization, it cuts out letters from many words making it unable to lemmatize properly. The indiscriminate cutting of prefix or

suffix during stemming are not always successful to get the main word The accuracy of the classifier decreases to 93.8%

5.8.2 Scenario 2:

If I do stemming after lemmatization, it does the same as above by making training set difficult to learn. As a result, the accuracy of the classifier slightly decreases to same i.e. 93.8%

5.8.3 Scenario 3:

If I do not perform any stemming and lemmatization, the training data have very high but needless variation of same words which creates more instances of same words making training set bigger too. As a result, overfitting occurs and the accuracy decrease to 91.7%

5.8.4 Final scenario:

In the end, I chose to perform only lemmatization because it extract the base form of word which makes the same word in different parts-of-speech identical. Thus its more helpful to control high variance while not affecting the biasness of the classifier. Also, stemming may increases recall but lowers precision. The accuracy of the classifier becomes 93.9%

Important learning outcome:
The accuracy really illustrate the fact that, despite changes in data normalization or standardization, Naive Bayes tends to perform very well as it works with the probability of data in given class where the probability of each data (word or sentence) are considered independent of each other. For example, the classifier consider that probability of word "corona" does not depends on that of "covid19" respecitvely.

5.9 Exclusive separation of training and test set

To check whether the classifier indeed is learning and applying its learning to find out relevancy in the document or not, we need pure test set of which the classifier is completely unaware of. Otherwise, we are detecting not topic but just the probability

of the text comes from a COVID-19 related or non-related articles.

In order to ensure the purity of the separation of testing articles, I have used train_test_split method of popular sklearn library with a fixed random state to get reproducible shuffling of the dataset and splitted 33% of the whole dataset as test data which are kept aside and NOT USED for training and analysing respectively [7].

5.10 Understanding the classification problem

As its a supervised machine learning problem because we know the labeling of each data, I had to consider one of the supervised ML algorithm. I also had to keep in mind for achieving good performance even though I used less training data compared to original dataset and spend less training time because there are other time-consuming operations to undergo before and after classifcation, such as extracting title and abstract of articles in the directories, loading and running Named Entity Recognition(NER) model for each COVID-19 related text etc.

As a result, I have implemented Naive Bayes classifier that classifies based on probabilities of events. It is commonly applied and performs well in many types of classification problem with less training data.

5.10.1 Calculating Document Term Matrix

It required to have a list of word frequencies appearing in all the articles in training set i.e. Document Term Matrix for each class in order to find the probability of each word of each row in test dataset for a given class i.e. label.

Therefore, to find the total number of times a word appears in that class in the training set I used CountVectorizer from sklearn library which can give the list i.e. Document Term Matrix [8].

5.10.2 Tackling Zero Frequency Problem

If a word from the sentence of test set does not occur in the class within the training set, the equation becomes zero. This is know as "zero frequency problem"

An approach to overcome this problem in a Bayesian setting, I used one of the additive smoothing technique known as "Laplace smoothing" is used where it basically adds 1

to the count for every attribute value-class combination when an attribute value does not occur with every class value.

5.11 Analysis of Named Entity Recognition (NER)

There are bio-medical corpus available from which Name Entity Recognition (NER) can extract different types of Named Entities such as proteins, genes and medical condition, diseases, chemicals, DNA, RNA, cellular components, amino acid etc. However, I could not find any official NER model trained on COVID-19 Open Research Dataset (CORD-19) [9].

I was consider one of two options.
1) To use Python implementation of GENIA tagger library trained on GENIA corpus
2) To explore and use one of pre-trained ScispaCy model

5.11.1 GENIA tagger library

In option 1, I could only get Named Entities if they are of type Protein, DNA, RNA, Cell line and Cell type respectively. There were no Entity type to identify diseases or related entities

5.11.2 ScispaCy NER Models

In case of option 2, The Spacy NER system contains a word embedding strategy using sub word features along with "Bloom" embed, and a deep convolution neural network with residual connections [10].

Nevertheless, as I have to detect entities related to COVID-19 which is a disease, the only relevant ScispaCy model which I could consider is named "en_ner_bc5cdr_md". It is trained on BC5CDR corpus and can detect two major Entity types namely "chemical"and "disease" respectively [11].

In the end, I chose the 2nd option because spaCy NER system is designed to give a good balance of efficiency, accuracy and adaptability. Spacy has support for word vectors, so it's fast and accurate [10].

5.11.3 Underlying behaviour of SciSpaCy NER Model:

The structured prediction framework is transition-based i.e instead of having each word attached tag for making it object of interest, they use the artificial recurrent neural network (RNN) architecture known as Long short-term memory (LSTM) which takes each words in the state where it can take some actions that moves the state from the current configuration to another or previous configuration i.e. instead of feedforward it feedback the connections. All the actions can differ based on the configuration making this transitional approach very flexible [12]. The action fix the label at the start of the entity for which when making custom NER model spaCy requires a .spacy format where each entity is given its entity type and its position of starting and ending character in sentences respectively.[13]

Transition	Output	Stack	Buffer	Segment
	[]	[]	[Mark, Watney, visited, Mars]	
SHIFT	[]	[Mark]	[Watney, visited, Mars]	
SHIFT	[]	[Mark, Watney]	[visited, Mars]	
REDUCE(PER)	[(Mark Watney)-PER]	[]	[visited, Mars]	(Mark Watney)-PER
OUT	[(Mark Watney)-PER, visited]	[]	[Mars]	
SHIFT	[(Mark Watney)-PER, visited]	[Mars]	[]	
REDUCE(LOC)	[(Mark Watney)-PER, visited, (Mars)-LOC]	[]	[]	(Mars)-LOC

Figure 8: Transition sequence for "Mark Watney visited Mars" with Stack-LSTM Model

For example: Suppose, we need to find Named Entity on the sentence "Mark Watney visited Mars".

The steps spaCy NER model will work is as follows:
0) Intialize empty stack and output arrays. Buffer array holds each token in the sentence
1) It takes the first word Mark from the buffer array and then do action "Shift"
2) We get "Mark Watney" in the stack and then do action/transition name "Reduce" to an combined entity where Mark is F-PER and Watney is L-Per and the combined entity is PER
3) If there is an entity i.e. reduce is successful, then its placed in the output array making stack empty
4) Repeat the process mentioned in Step 1 to 3

It might looks like a tagging task but due to transition based structure recognition framework, it can invalid an action/ transition even though it has high score. For example: "This is Mark Google" will have 2 entities Mark(PER) and Google(ORG) even though Google is placed after Mark as a L-PER (LastName of Person) Entity and has

high score as L-NER because the transition is invalid. If spaCy used greedy heuristic than it would have been a problem [12].
Importantly, even though it can project the most frequent entities with acceptable accuracy, it misclassify the most important entity "COVID-19" as chemical due to the lack of COVID-19 related text in the corpus its trained upon and the naming convention of it as a virus disease.

5.11.4 Potential optimization of NER Models

There are many ways I could have done the NER better. For example, I could have custom trained a spacy model with CORD-19 dataset by labeling each word if its required entity with entity type and its position of starting and ending character but it would have been very time consuming given the scope of the project[13]. Also, I could have used BERT-based neural network model implementation trained on CORD-19 dataset but it also needs a lot of time for parameter tuning[14].

6 Conclusion

In the end, I would say that I got sufficient idea how to face any kind of text classification challenges and initially approach it . If had more time, I would have made my code more robust in training and processing texts.

For example: If the training set do not have any COVID-19 related articles, the classifier cannot learn due to arithmetic error during intermediate probability calculations.

I would also make a sciSpaCy NER model based on CORD-19 dataset using custom neural network parameter configuration and more importantly with as much large dataset labeled with entity and its type as possible [15] [16].

References

[1] https://ftp.ncbi.nlm.nih.gov/pub/pmc/oa_bulk/non_comm_use.I-N.xml.tar.gz.

[2] https://libguides.rcsi.ie/covid19/searchstrategy.

[3] https://www.nps.org.au/glossary.

[4] https://www.everydayhealth.com/coronavirus/coronavirus-glossary-key-terms-about-t

[5] https://www.tmc.edu/news/2020/05/covid-19-crisis-catalog-a-glossary-of-terms/.

[6] https://countwordsfree.com/stopwords.

[7] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html.

[8] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html.

[9] https://github.com/allenai/cord19.

[10] https://blog.vsoftconsulting.com/blog/understanding-named-entity-recognition-pre-

[11] https://allenai.github.io/scispacy/.

[12] https://www.youtube.com/watch?v=sqDHBH9IjRU.

[13] https://medium.com/analytics-vidhya/custom-named-entity-recognition-ner-model-wit

[14] https://aistairc.github.io/BENNERD/.

[15] https://github.com/dmis-lab/biobert.

[16] http://prm-ezcatdb.cbrc.jp/bennerd/.