DS 5230: Final Project Report

COVID Literature Clustering

Abhilash Hemaraj
hemaraj.a@northeastern.edu

Niyati Chopra
chopra.ni@northeastern.edu

Abstract

This project involves working on the CORD-19 dataset uploaded as a Kaggle competition by the Allen Institute of AI in collaboration with other institutions. It is available in the form of JSON files. For this project, paper id, title, authors, abstract, and body text were taken as data features. This project aims to cluster these literature articles available to help identify the research done on an array of topics related to COVID-19. The dataset was sampled to include 20% of the data randomly. Working exclusively on body text, they were converted to a TF-IDF vector alongside performing tokenization and stemming and removal of stop words. Then, this sparse TF-IDF feature matrix was used to perform dimensionality reduction using PCA and TSNE. With both the reduced datasets, two clustering algorithms, K-Means and DBSCAN were performed. K Means and DBSCAN both produced results much faster when using the TSNE reduced data. Due to computation limitations, DBSCAN did not give us clusters when implemented on the PCA reduced data. The most well-defined clusters were obtained after performing K Means on TSNE reduced data. To validate the model outputs, a Multiclass Naive Bayes model was fit using the clusters as labels. The project concludes by analysing the comparisons of various clustering models and their efficacy.

1 Introduction

Since December 2019, the new coronavirus or SARS COV-2 has been taking over the world [4]. Claiming over three hundred thousand lives in a year in the United States alone, this virus has claimed more than all of the flu seasons combined since 1918. Since then, thousands of articles, blogs, papers, studies, etc. have been published. Allen Institute for AI along with several other institutions released a dataset [3] containing over a hundred thousand such literary articles as a competition on Kaggle.

The goal of this project is to use unsupervised machine learning techniques on the dataset to organize these literary articles by the topics they cover and gain more insight on how much research has taken place for which topics. For example, an entire cluster would contain papers published on the vaccine, another one would have papers on patient demographics, another would have studies related to the spread of the virus, et cetera.

After obtaining the clusters,

1.1 The Dataset

The data set is called COVID-19 Open Research Dataset or CORD-19. It contains over two hundred thousand rows of articles of which over one hundred thousand have full body text about COVID-19, SARS-CoV-2, and related coronaviruses.

The dataset was uploaded in the form of JSON files. These files were structured to contain information like paper id, title, authors, abstract, body text, year, citation spans, reference spans, sections, back matter, pages, volume, etc. For this project, only paper id, title, authors, abstract, and body text were taken as features of the dataset.

Since the dataset was very large, only a random sample of 20% or twenty-five thousand rows were taken into consideration while implementing techniques. This was done due to computational limitations. Using the entire dataset would be a good way to improve the project if the computational power and resources are available.

1.2 Converting from text to numbers

The body text was converted to a TF-IDF matrix. TF-IDF stands for term frequency-inverse document frequency. This process also involves tokenization and stemming followed by the removal of stop words from the body text.

1.3 Dimensionality Reduction

After obtaining the TF-IDF matrix, dimensionality reduction was performed on it. Two methods of dimensionality reduction are implemented: Principal Component Analysis (PCA) and t distributed Stochastic Neighbour Embedding (TSNE).

1.4 Clustering

After getting the reduced data set after performing dimensionality reduction, two methods of clustering were performed on each, K Means and Density-Based Spatial Clustering of Applications with Noise (DBSCAN). Both these methods gave a list of labels for the assigned clusters for both the reduced datasets.

1.5 Multinomial naive Bayes classification

A multinomial naive Bayes classification model was fit for all four of the variations in clustering the data set. By testing various alpha values for each of the models, we got optimal accuracy scores and used this to check how accurate our clusters were. The highest accuracy score was obtained on the PCA reduced, K Means fit set of cluster labels.

2 Background

Some of the techniques covered in this project were not taught in the scope of the course. These included the work done related to the pre-processing techniques such as regular expressions, tokenization and stemming, and the multinomial naive Bayes classifier.

Regular Expressions: Pre-processing the text corpus to retain only relevant information can be substantial in creating a model that produces better results. To match and replace unwanted characters such as URLs, User ID's, citation numbers, hyperlinks, and other symbols, regular expressions has proved to be quick and reliable. Below is a snapshot of how the matching works [6].

Fig 1: Visual e.g. of pattern matching using regular expressions

Since TF-IDF-vectorizer from sklearn was to be used for constructing a vector representation from the corpus, we needed custom functions designed specifically for tokenization and stemming, to be passed into the tokenizer.

2.1 Tokenization

This is the process of breaking down a sentence into individual words called tokens. So e.g. a string "I fit a model!" is converted to an array of tokens: ['I', 'fit', 'a', 'model', '!']

2.2 Stemming

Is the process of removing the word affixes; so, as the name suggests 'stem' a token to its base form. E.g. meet, meeting, meets gets converted to just meet.

The TF-IDF matrix was used as the basis for performing all the machine learning tasks on the dataset and also to obtain the word cloud. Whereas the Multinomial Naive Bayes classifier was used for verification of our obtained clusters.

2.3 Naive Bayes

Based on the Bayes theorem of conditional probability, it is one of the fastest and simplest to implement among the suite of classification algorithms. It works on the assumption that the features are independent. Since we are only aiming to get a baseline knowledge about whether the clustering labels makes sense or not, we chose to go with a very computationally efficient algorithm.

3 Related Works

Text Clustering is the field of cluster analysis where we are taking text documents as inputs and leveraging Natural Language Processing techniques for modeling and visualization. Herein we have focused on Document-based clustering, such that each row entry in the dataset serves as a document.

Several techniques have been implemented in this space and even on the CORD dataset; Since it presents a multi-array of challenges to deal with.

The present paper takes inspiration from the work done by MaksimEkin [1]. The paper works on the only PCA as the dimensionality reduction used for further modeling, with t-SNE used purely for visualization. The paper also goes on to perform topic modeling on each cluster and runs an SGD classifier to validate the clusters.

Since this is a Kaggle competition, there has been several attempts at the same task. One of them by Adel del Valle [2]; In his attempt, he trains a genism word2vec model to create embeddings and uses t-SNE for visualizations.

4 Project Description

The data itself which is majorly text is available in the form of JavaScript Object Notation (JSON) format. To convert the dataset into a usable tabular format, we parsed the individual JSON files using python scripting. The dataset rendered has dimensions (128915, 5). For further use, keeping computational limitations in mind, a smaller subset of the dataset is randomly sampled, accounting for 20% of total records.

Custom regex matching and substitution scripts in python were executed to clear the corpus of un-usable non-alphabetic characters. These include but are not limited to the citation- numbers, hyperlinks, URLs, and other separation clauses. The dataset is now ready to be looked at with an exploratory perspective. Subsequently, libraries were used to build a word cloud out of the dataset. A word cloud gives a sense of the prominent words after accounting for some stop-words.

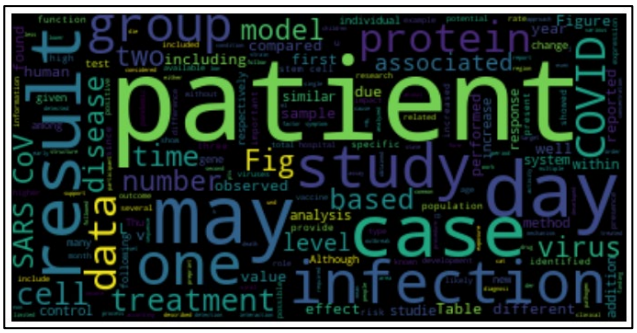

Figure 2: Word Cloud visualizing the body-text feature of data.

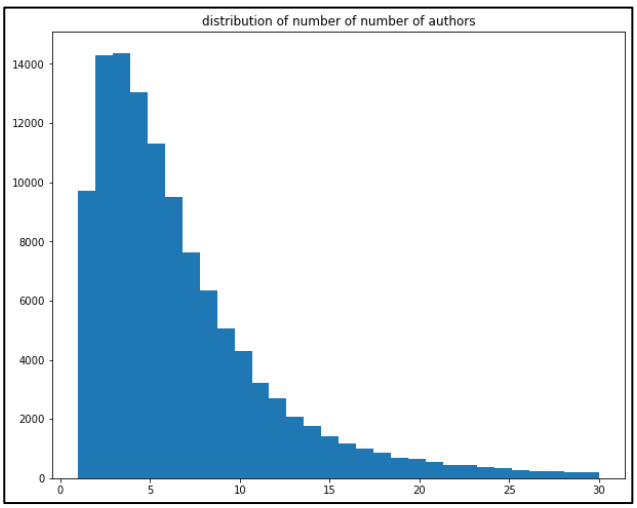

Figure 3: A histogram of the number of authors per document (upper limit of 30 authors)

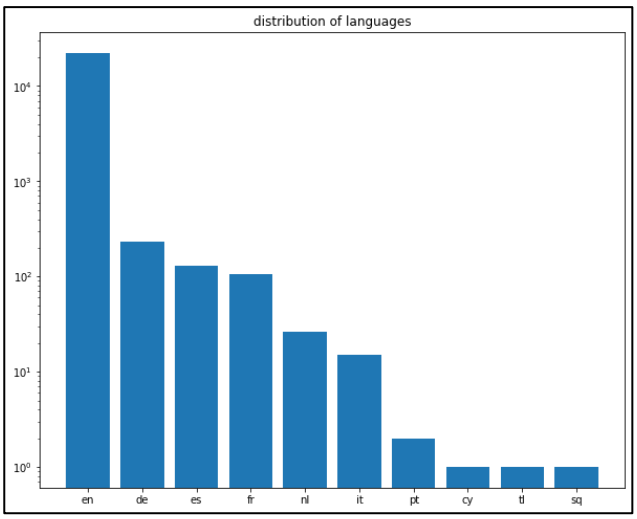

Figure 4: A bar-plot of top 10 languages present in the corpus.

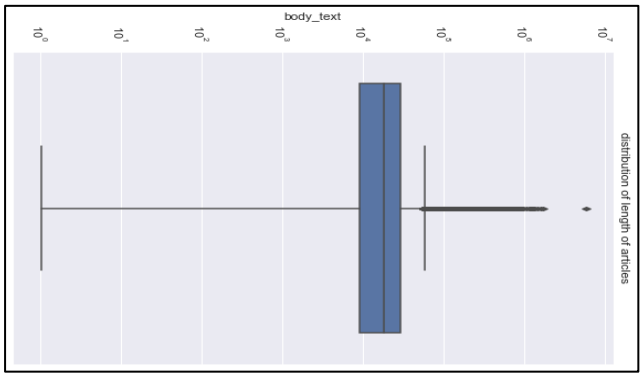

Figure 5: A boxplot of the length of articles (character count)

4.1 Term Frequency-Inverse Document Frequency

After Text processing and EDA is the part of using Natural Language Processing Techniques to further process the text contained in body text. NLTK's tokenization and stemming libraries are leveraged to build a function that can automatically convert each document into tokens. Now the corpus is ready to be converted into word vectors using sklearn's TfidfVectorizer.

The vectorizer works on the formula for TF-IDF that compares the uniqueness of words in a document scaled by the number of times the word appears in the whole corpus.

$$TFIDF = tf(t,d) * \frac{\log(N)}{df(d,t)}$$

Where t is the number of occurrences of the word in document d; df(d,t) is the number of documents containing the word t. And N is the number of documents. We customized the vectorizer to account for unigrams, bigrams, and trigrams. The run-time for the vectorizer was about 45 minutes for 25783 records.

The vector matrix is used for dimensionality reduction. As per the scope of the project, two techniques will be compared viz. PCA and t-SNE.

4.2 Principal Component Analysis

Principal Component Analysis or PCA is a technique to project the original data points onto principal component axes. The idea is to reduce the dimensionality of the features by losing minimal information. Initially, a covariance matrix is built from the vectors; and the eigenvalues and corresponding eigenvectors are calculated. Selecting k number of components from eigenvectors ordered by decreasing eigenvalues, the data is projected.

$$Z_t = U^t_{reduced} * x_t$$

$$where\ Z_t\ is\ the\ linear\ transformation\ of\ x_t$$

First, sklearn's PCA library is instantiated keeping 95% of the variance. The number of components that took to reach the threshold of explained variance is 644 components.

4.3 T-Distributed Stochastic Neighbourhood Embedding

Furthermore, the same word vectors are fed into sklearn's TSNE library. The parameters of TSNE include:

Perplexity: It is an estimation of the number of neighbourhoods points each data point would have. Usually ranges from 0-50, but since the document is fairly large we are using a perplexity parameter of 100. This computation took around 1000s.

With the reduced dimensional data handy, the next step would be to build clustering algorithms. In this project, two such clustering algorithms will be compared by executing it on both the dimensionality reduction techniques.

4.4 K-means Clustering

K-means is a Centroid-based clustering technique. It uses distance measures to calculate the closeness of the data points. The type of distance measure can be altered. Most commonly used is the Euclidean distance measure. The basic algorithm works as follows:

1. Choose K number of clusters and initialize as many centroids.
2. Calculate the distance of each data point to every centroid, forming K clusters.
3. Calculate centroid of each cluster
4. Repeat till convergence.

The sklearn's K-means employs kmeans++, which is a method of choosing initial centroids. The logic is:

1. Choose a centre at random.
2. Calculate the distance of other points with the centroid
3. Choose the next centroid such that the probability of choosing a point as the next centroid is directly proportional to the distance between the two.
4. Repeat till k centroids are calculated.
5. Run the K-means algorithm.

4.5 Density Based Spatial Clustering of Application with Noise (DBSCAN)

DBSCAN or Density-Based Spatial Clustering of Applications with Noise is an unsupervised clustering technique. It works on two parameters: epsilon and min points. Epsilon defines the maximum distance between two points for them to be a part of the same cluster. Min points define the minimum number of points to form a dense region.

The main idea behind DBSCAN is to classify points into three groups: core point, border point, and noise point.
A core point is one that has min points number of points within epsilon distance. A boundary point is a point that is a neighbour of a core point but does not have min points number of points within epsilon distance. A noise point is one that is neither a core point nor a boundary point.
Algorithm:

1. Choose a point in the space. Find the distance to all other points in that space. If there are at least min points number of points within at most epsilon distance of the point, mark this point as a core point and all its neighbors within epsilon distance as part of this cluster.
2. If this point is not a core point itself but is in the vicinity of another core point, then mark this point as a boundary point of that cluster
3. If this point is neither a core point nor a boundary point, then mark it as a noise point
4. Repeat for all points in the space till each is not classified

It is a very straightforward algorithm that can be used to identify clusters of arbitrary shapes and sizes.

4.6 Naïve Bayes Classifier

A multinomial naïve Bayes classifier is a popular classifier for text data. It works on Bayes' rule of probability which is:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

Where $P(A|B)$ represents the probability of A happening, given B has already taken place.
Multinomial naïve Bayes classifier taken in the TF-IDF vector as the feature vector. Then, taking the vector value of each word, we calculate the probability of the word being in the cluster. Here, $P(A|B)$ will represent the probability of a word occurring given we are given a cluster label. Based on the values of the probabilities obtained, each word is classified into a cluster with the maximum conditional probability.
Multinomial naïve Bayes classifier was chosen for verification because it works well with text data and TF-IDF vectors. The metrics used to evaluate NB model is accuracy score.

5 Empirical Results

After performing all the above-mentioned tasks, the TSNE dimensions were used to generate a plot representing the dataset.

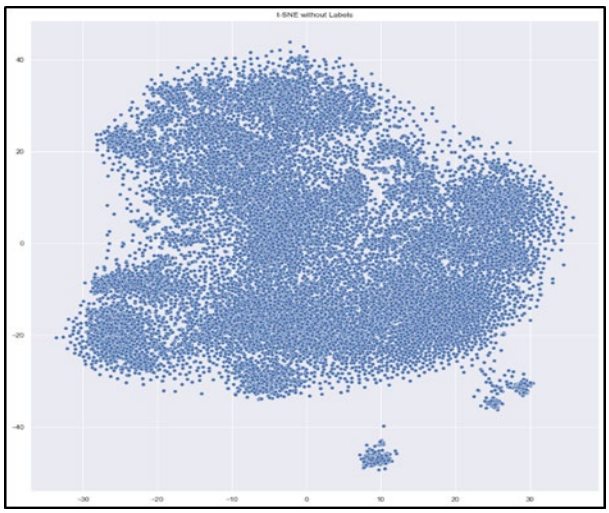

Figure 6: t-SNE's high-level representation of body text. (without labels)

The next step was to perform clustering. First, K Means clustering was implemented.

On the PCA reduced data, this is what the distortion (or inertia) vs K value graph looked like:

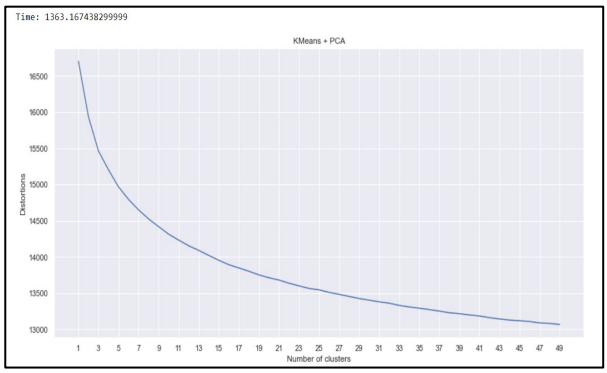

Figure 7: Distortion plot obtained running K-means

As observed, the elbow value is at point 15 and that was chosen as the optimal number of clusters. For model comparison, the time taken to execute the graph was noted to be 1363.17 seconds.

These labels were then fit on the plot generated using TSNE and this was the plot obtained:

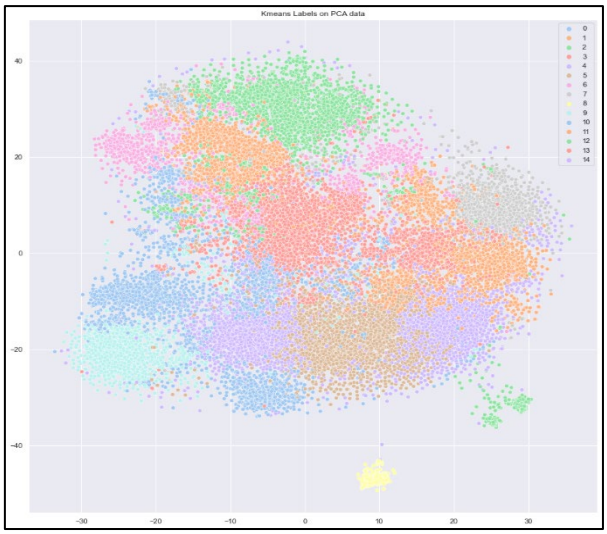

Figure 8: labels obtained from kmeans-pca mapped on tsne

Similarly, K Means was run on the TSNE reduced dataset to generate the following elbow curve:

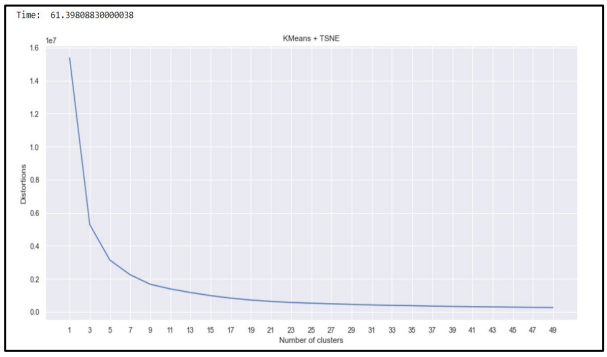

Figure 9: Distortion plot of Kmeans run on tsne data.

As observed, the graph has a much sharper elbow at k = 9. The time is taken (61.39 seconds) was also far less as compared to K Means on PCA data.

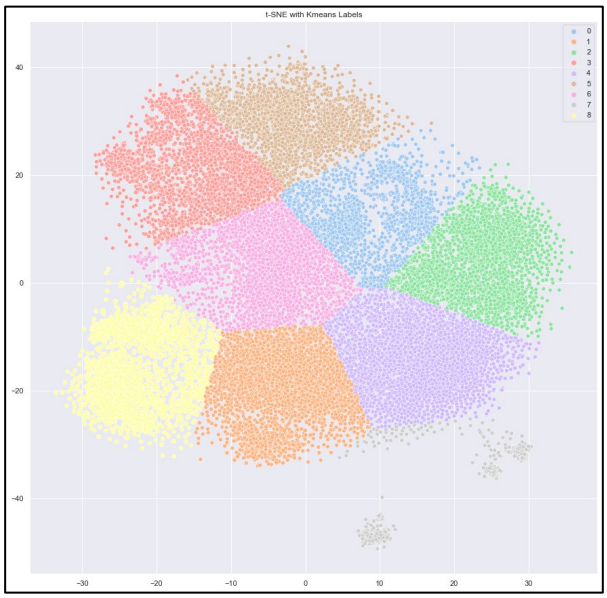

Figure 10: The plot generated using labels on the TSNE generated graph

The clusters obtained are very distinct, they have very clear boundaries.
Further model comparison was done using multinomial naive Bayes classification.

The next clustering algorithm used was DBSCAN. The hyperparameters for DBSCAN are epsilon and min_samples. Epsilon is the maximum distance between two samples for them to be considered neighbours of each other. Min_samples represents the minimum number of points in an epsilon radius for the point to be considered as a core point.
After a lot of trial and error, it was found that for the TSNE reduced data, the values epsilon = 4 and min_samples = 500 give the least number of noise points.

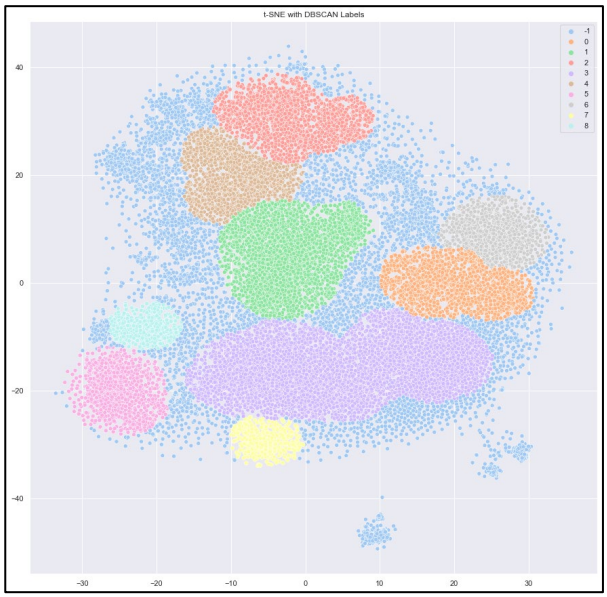

Figure 11: The plot generated using the DBSCAN labels on the TSNE reduced dataset

This is the distribution of points in the clusters. -1 represents noise points.

Labels	Label Counts
-1	6717
3	5940
1	3285
2	2164
0	1882
4	1843
5	1448
6	1231
8	651
7	622

The time taken to run this algorithm on TSNE reduced data was 1.64 seconds.

To run DBSCAN on the PCA reduced data, there was a lot of backlog in terms of computation capacity. Running a single iteration on DBSCAN took at least 1600 seconds. As the trial and error method was being used here, no proper plot could be generated for DBSCAN on the PCA reduced data. This was the plot generated using similar hyperparameters as DBSCAN on PCA reduced data. As observed, all points are being classified into a single cluster.

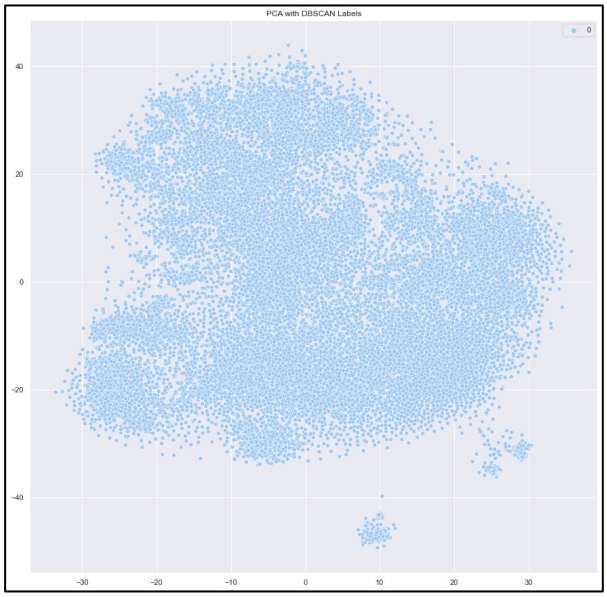

Figure 12: The plot generated using the DBSCAN labels on the TSNE reduced dataset:

After obtaining these results for clustering, a multinomial naive Bayes classifier was fit on the TF-IDF matrix and with ‘y’ values as the predicted class labels. The model was passed a range of alpha values to get the highest possible accuracy score.
These were the alpha values and their corresponding accuracy scores:

Clustering technique	Alpha value	Accuracy score
PCA + K Means	0.3	0.85
TSNE + K Means	0.3	0.78
TSNE + DBSCAN	0.3	0.76

6 Conclusion and future scope

While comparing dimensionality reduction algorithms, it was found that PCA was better as it retained 95% variance in data. TSNE was a good method to reduce the data down to two dimensions to get an approximate visualization of the multidimensional data in a two-dimensional space. Due to the lower number of dimensions, clustering algorithms were executed in much less time as compared to PCA reduced data.

It can also be concluded from the result of the naive Bayes classifier that the clusters were more accurately predicted by the K Means performed on the PCA reduced dataset despite the plot for K Means labels on TSNE reduced dataset being more distinct.

For the future scope of this project, if computation resources are available, the DBSCAN algorithm can be implemented using hyperparameter tuning techniques, or trying variations of the algorithms, like OPTICAL. Some other clustering techniques (like hierarchical clustering) can be implemented as well. Other words to vector techniques can be used in place of TF-IDF to see if those perform better with clustering techniques.

Other pre-processing techniques, like using more stop words, performing lemmatization, paying more attention to the different languages present, using a bigger sample of the dataset, et cetera.

This project has a lot of potential to help in the recognition of patterns and similarities between literatures related to SARS-COV-2. The results obtained could be useful in finding how earlier studies can be used to fight this pandemic.

7 References

[1] Maksim Ekin Eren, Nick Solovyev, Edward Raff, Charles Nicholas, and Ben Johnson. 2020. COVID-19 Kaggle Literature Organization. In Proceedings of the ACM Symposium on Document Engineering 2020(DocEng '20). Association for Computing Machinery, New York, NY, USA, Article 15, 1–4. DOI:https://doi.org/10.1145/3395027.3419591

[2] Adel del Valle https://www.kaggle.com/adelfdelvalleperez/cord-19-fasttext-words-clustering

[3] Dataset: https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge

[4] https://elemental.medium.com/us-covid-19-deaths-compared-to-diseases-pandemics-wars-2a7495a43280

[5]https://stats.stackexchange.com/questions/263539/clustering-on-the-output-of-t-sne/264647#264647

[6] https://regex101.com/