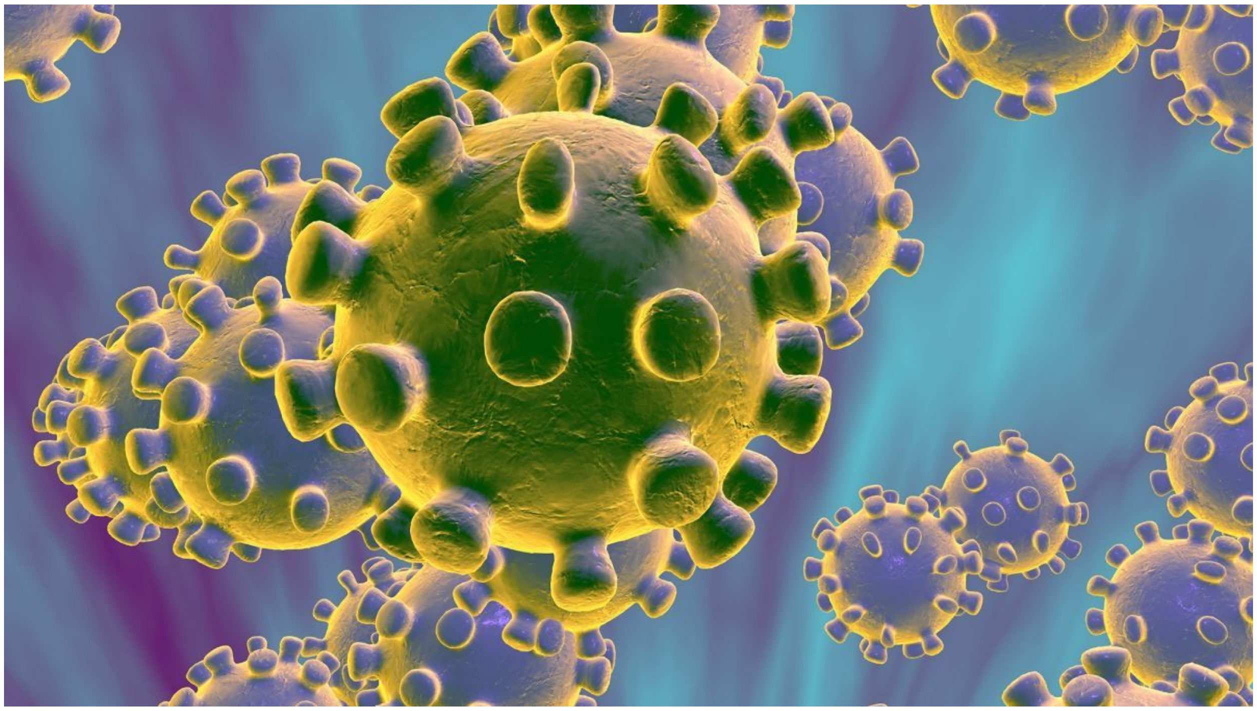


What do we know about COVID-19 risk factors?

04.03.2020

Daphney Carol Valiatingara	A20446937	dcarolvaliatingara@hawk.iit.edu
Neha Desai	A20402675	ndesai33@hawk.iit.edu
Ankit Patil	A20451742	apatil44@hawk.iit.edu

Table of Contents

1. Overview

As we are all aware, the COVID pandemic has violently and unexpectedly taken over all nations attacking everyone and sparing no one. It becomes necessary then we put our efforts from all sectors to fight this pandemic as one unit. We attempted to do just that by taking part in the COVID-19 Open Research DataSet Challenge. This challenge is open in order to find out more about this novel coronavirus from research papers and scholarly articles on this virus. The White House and a group of leading research groups have accumulated resources of 45000 scholarly articles and have posed 10 tasks/questions that they attempt to answer with these datasets in order to gain new insights. Questions that they attempt to answer range from details of its transmission, incubation, risk factors, vaccines, etc. Through this project, we attempt to answer the question of information regarding risk factors associated with this coronavirus by applying text and data mining techniques to attempt to find information on the same.

2. DataSets

We obtained our data set from Kaggle using the following link :

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

The dataset contains a resource of 45773 scholarly articles. In this set, over 33,000 have full-text information about COVID-19, SARS-CoV-2, and related coronaviruses. The articles are presented in JSON format.

It was created by Allen Institute for AI in partnership with the Chan Zuckerberg Initiative, Georgetown University's Center for Security and Emerging Technology, Microsoft Research, and the National Library of Medicine - National Institutes of Health, in coordination with The White House Office of Science and Technology Policy.

3. Research Problems

This project aims to cluster similar articles together and also focus on solving the research problem that answers this question: "What are the risk factors associated with COVID-19?

We want to tackle the above question because we want to understand who is at risk for this virus. Due to the volume of information available on this virus and COVID-19 spreading at an alarming rate, it is impossible for scientists to track through this vast information in search of answers in time to provide a solution.

These are the problems we attempted to solve:

3.1 Cluster similar articles together to understand the structure and look for underlying patterns in them.

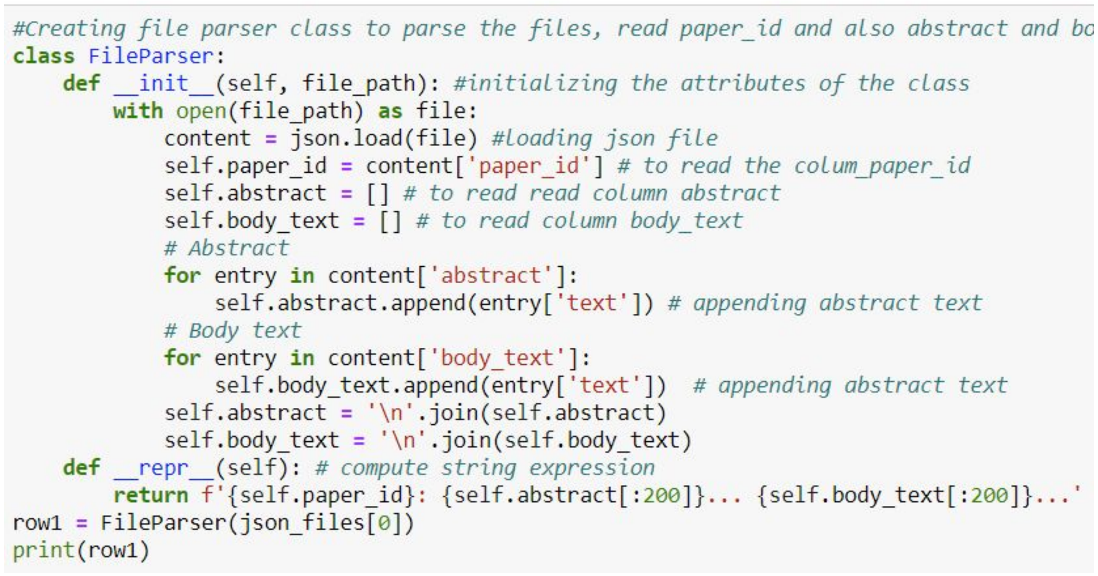
3.2 Scraping the given research articles to understand the risk factors associated.

3.3 Extract text sections /articles that are relevant to the proposed question from research articles.

4. Potential Solutions - KDD

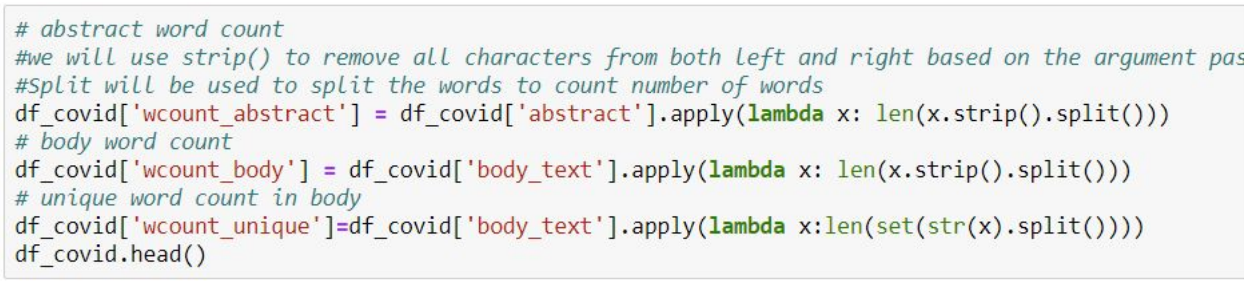
4.1 Clustering the data using different attributes in research papers.

- Used FileParser function to parse the data based on paper_id, body text and abstract.



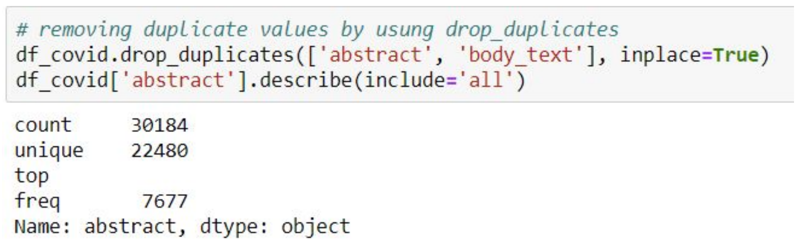
```
#Creating file parser class to parse the files, read paper_id and also abstract and bo
class FileParser:
    def __init__(self, file_path): #initializing the attributes of the class
        with open(file_path) as file:
            content = json.load(file) #loading json file
            self.paper_id = content['paper_id'] # to read the colum_paper_id
            self.abstract = [] # to read read column abstract
            self.body_text = [] # to read column body_text
            # Abstract
            for entry in content['abstract']:
                self.abstract.append(entry['text']) # appending abstract text
            # Body text
            for entry in content['body_text']:
                self.body_text.append(entry['text'])  # appending abstract text
            self.abstract = '\n'.join(self.abstract)
            self.body_text = '\n'.join(self.body_text)
    def __repr__(self): # compute string expression
        return f'{self.paper_id}: {self.abstract[:200]}... {self.body_text[:200]}...'
row1 = FileParser(json_files[0])
print(row1)
```

- Used lambda functions to count words in the text.



```
# abstract word count
#we will use strip() to remove all characters from both left and right based on the argument pas
#Split will be used to split the words to count number of words
df_covid['wcount_abstract'] = df_covid['abstract'].apply(lambda x: len(x.strip().split()))
# body word count
df_covid['wcount_body'] = df_covid['body_text'].apply(lambda x: len(x.strip().split()))
# unique word count in body
df_covid['wcount_unique']=df_covid['body_text'].apply(lambda x:len(set(str(x).split())))
df_covid.head()
```

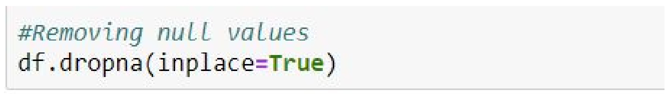
- Data Preprocessing
    1. Eliminating duplicate values



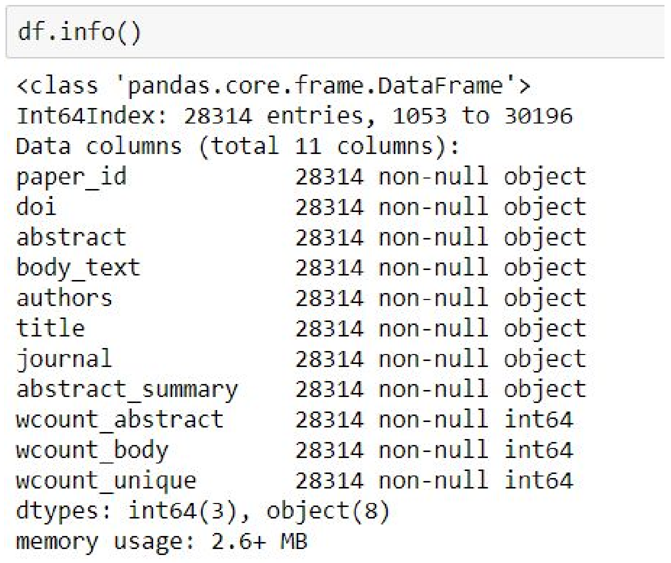
```
# removing duplicate values by usung drop_duplicates
df_covid.drop_duplicates(['abstract', 'body_text'], inplace=True)
df_covid['abstract'].describe(include='all')
```

```
count     30184
unique    22480
top
freq       7677
Name: abstract, dtype: object
```

2. Eliminating Null values



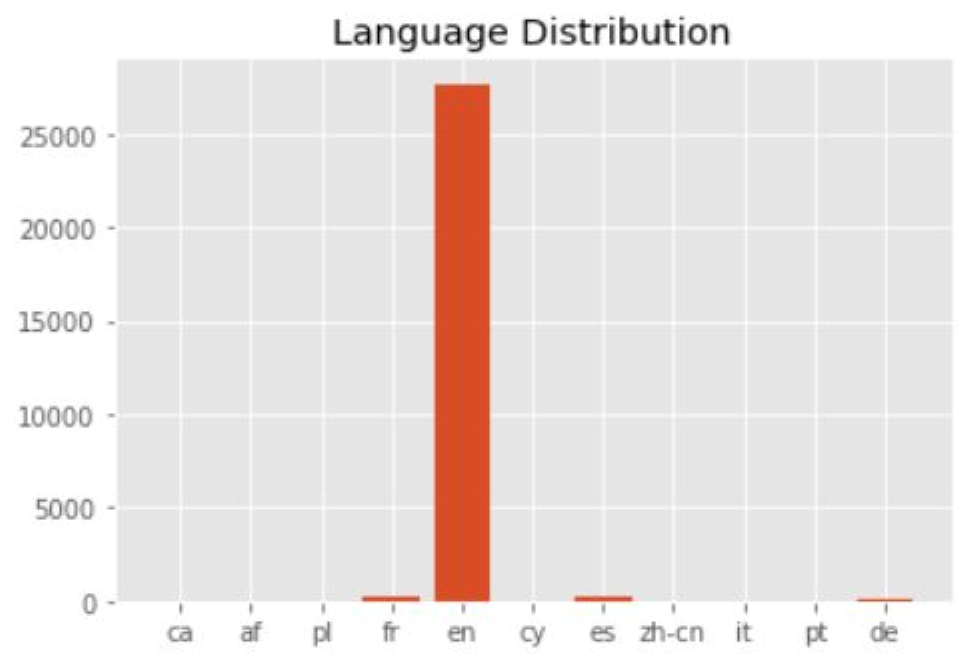
```
#Removing null values
df.dropna(inplace=True)
```



```
df.info()
```

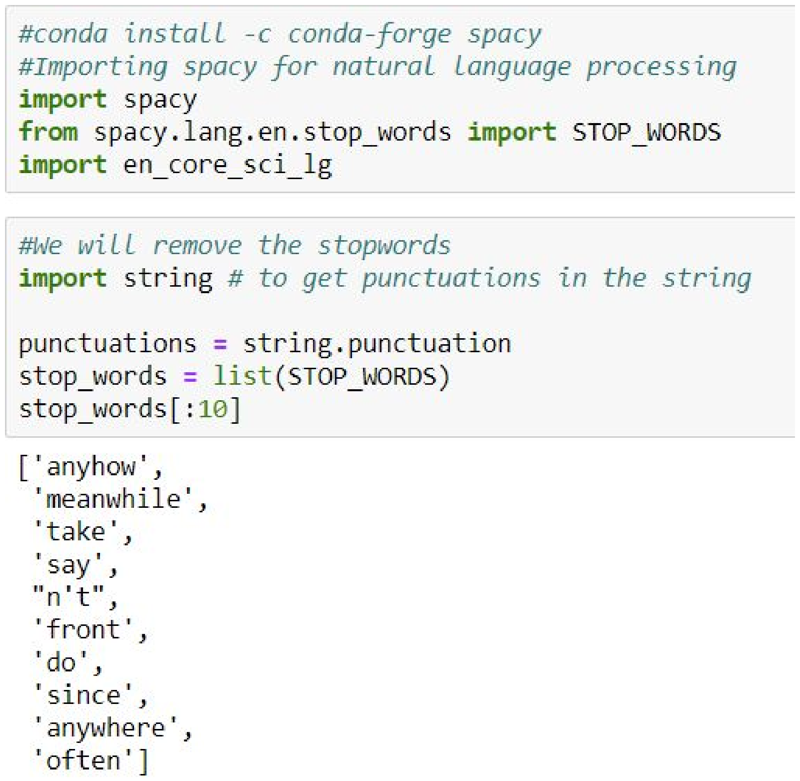
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 28314 entries, 1053 to 30196
Data columns (total 11 columns):
paper_id            28314 non-null object
doi                 28314 non-null object
abstract            28314 non-null object
body_text           28314 non-null object
authors             28314 non-null object
title               28314 non-null object
journal             28314 non-null object
abstract_summary    28314 non-null object
wcount_abstract     28314 non-null int64
wcount_body         28314 non-null int64
wcount_unique       28314 non-null int64
dtypes: int64(3), object(8)
memory usage: 2.6+ MB
```

- Used Langdetect module to find out different languages and then plot data based on different languages.



- We saw that english language was the most used language hence, we dropped all the other languages except english and further worked on English language.

- Used Spacy module for Natural Language Processing to eliminate Stop Words.



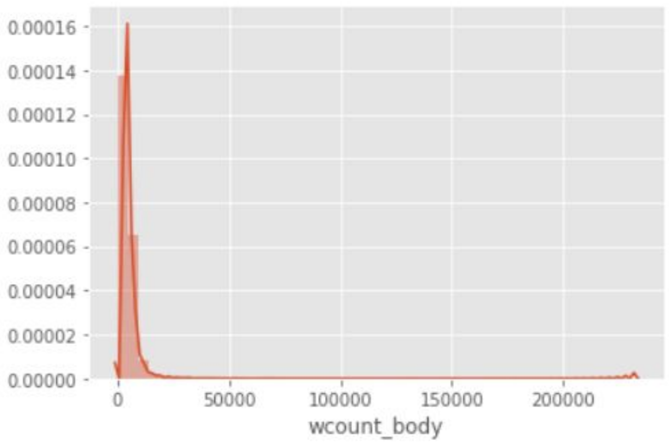
```
#conda install -c conda-forge spacy
#Importing spacy for natural language processing
import spacy
from spacy.lang.en.stop_words import STOP_WORDS
import en_core_sci_lg
```

```
#We will remove the stopwords
import string # to get punctuations in the string

punctuations = string.punctuation
stop_words = list(STOP_WORDS)
stop_words[:10]
```

```
['anyhow',
 'meanwhile',
 'take',
 'say',
 "n't",
 'front',
 'do',
 'since',
 'anywhere',
 'often']
```

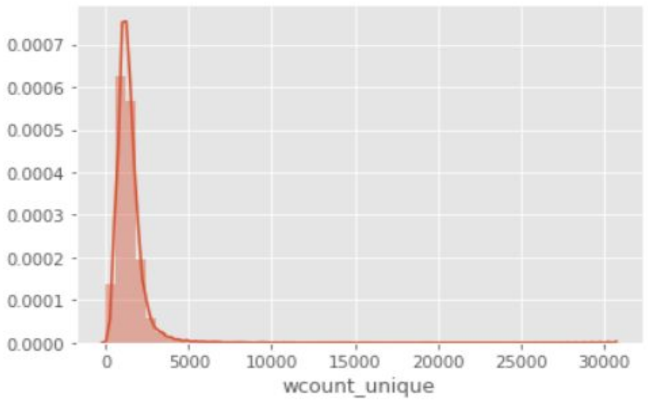
- By plotting the words we see that all of the research papers have word count less than 5000, i.e. each research paper has a maximum length of 5000 words.

```
count     27679.000000
mean       4672.184508
std        5165.128423
min          12.000000
25%        2595.000000
50%        3809.000000
75%        5528.000000
max      232431.000000
Name: wcount_body, dtype: float64
```

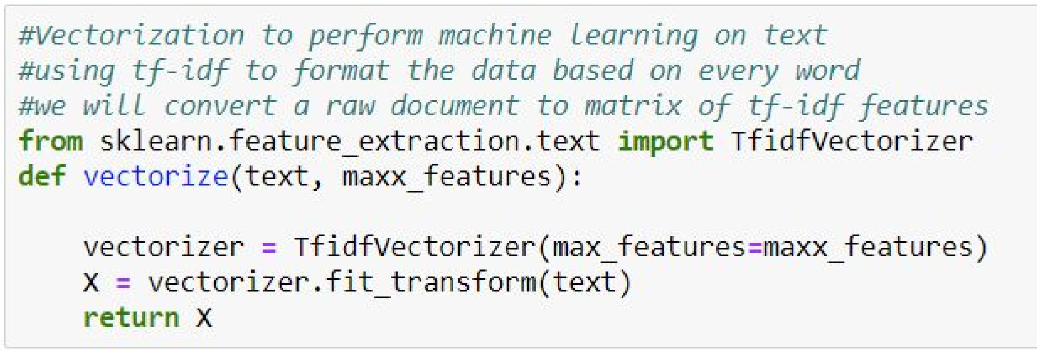


Unique word count:

```
count    27679.000000
mean      1431.322194
std        929.678683
min         12.000000
25%        960.000000
50%       1277.000000
75%       1689.000000
max      30523.000000
Name: wcount_unique, dtype: float64
```

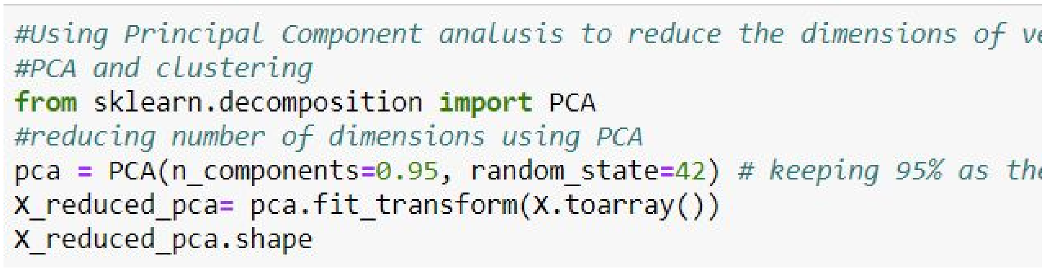


- Used vectorization process to apply machine learning algorithms on text. We converted the raw document to a matrix of Tf-idf features.



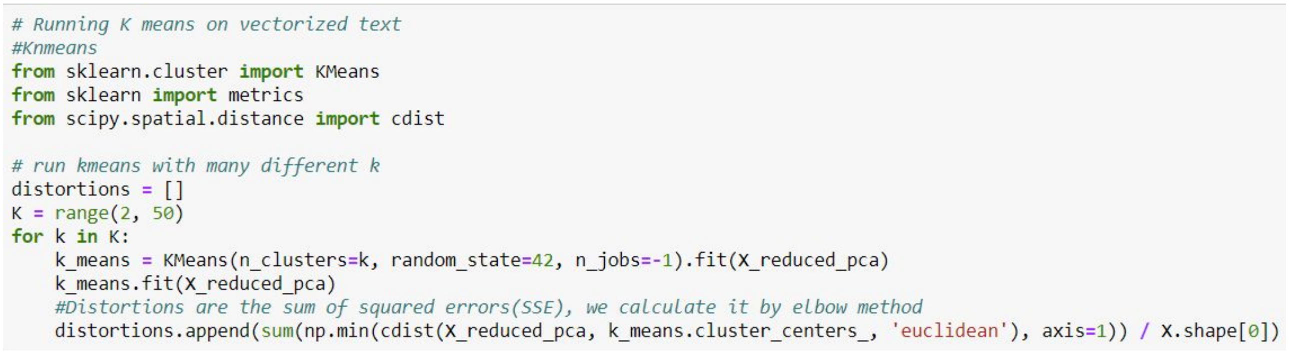
```python
#Vectorization to perform machine learning on text
#using tf-idf to format the data based on every word
#we will convert a raw document to matrix of tf-idf features
from sklearn.feature_extraction.text import TfidfVectorizer
def vectorize(text, maxx_features):

    vectorizer = TfidfVectorizer(max_features=maxx_features)
    X = vectorizer.fit_transform(text)
    return X
```

- Used Principal Component Analysis to reduce the dimension of the vectorized data.



```python
#Using Principal Component analusis to reduce the dimensions of ve
#PCA and clustering
from sklearn.decomposition import PCA
#reducing number of dimensions using PCA
pca = PCA(n_components=0.95, random_state=42) # keeping 95% as the
X_reduced_pca= pca.fit_transform(X.toarray())
X_reduced_pca.shape
```

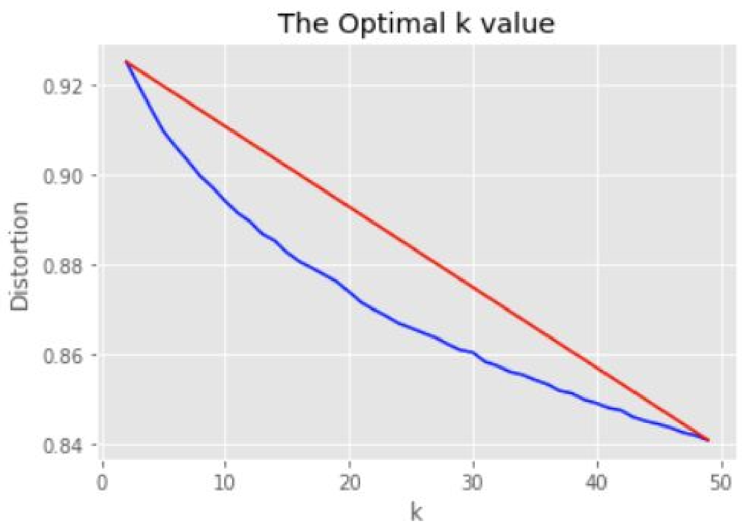
- To find the optimal value of K we used the elbow method. To find the optimal value we plotted the distortions.



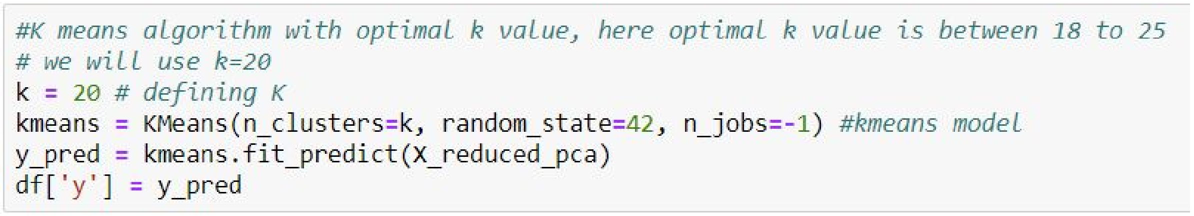
```python
# Running K means on vectorized text
#Knmeans
from sklearn.cluster import KMeans
from sklearn import metrics
from scipy.spatial.distance import cdist

# run kmeans with many different k
distortions = []
K = range(2, 50)
for k in K:
    k_means = KMeans(n_clusters=k, random_state=42, n_jobs=-1).fit(X_reduced_pca)
    k_means.fit(X_reduced_pca)
    #Distortions are the sum of squared errors(SSE), we calculate it by elbow method
    distortions.append(sum(np.min(cdist(X_reduced_pca, k_means.cluster_centers_, 'euclidean'), axis=1)) / X.shape[0])
```

We see that the optimal value is between 15 and 25, hence we choose 20 to be an optimal value for K.

```
#Plotting the distortions
X_line = [K[0], K[-1]]
Y_line = [distortions[0], distortions[-1]]

# Plot the elbow
plt.plot(K, distortions, 'b-')
plt.plot(X_line, Y_line, 'r')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Optimal k value')
plt.show()
```

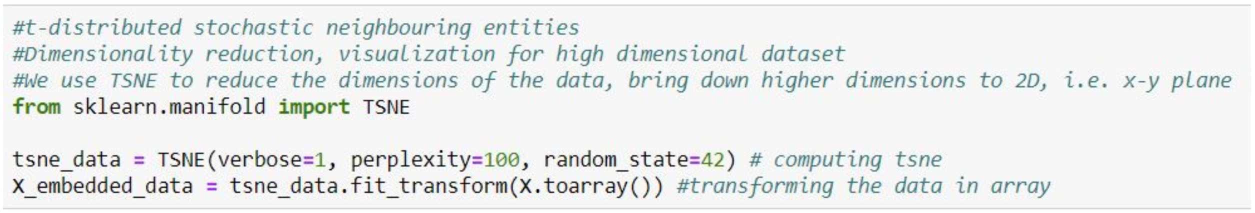


- K-Means clustering



```
#K means algorithm with optimal k value, here optimal k value is between 18 to 25
# we will use k=20
k = 20 # defining K
kmeans = KMeans(n_clusters=k, random_state=42, n_jobs=-1) #kmeans model
y_pred = kmeans.fit_predict(X_reduced_pca)
df['y'] = y_pred
```

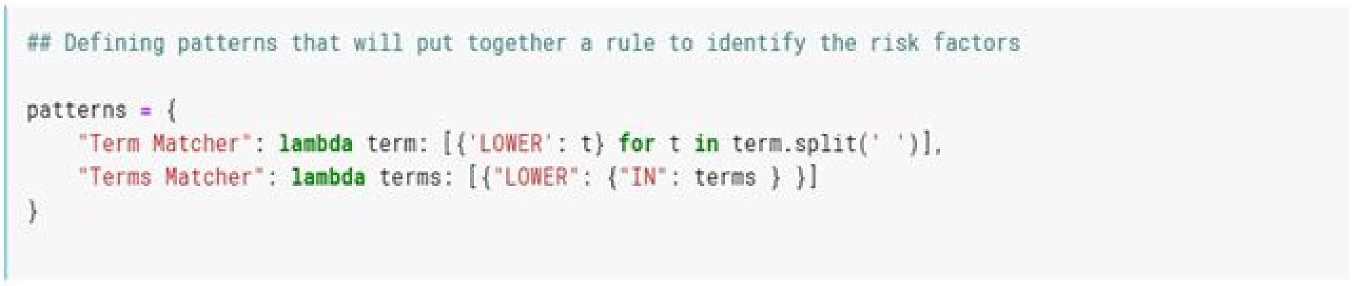
- Used T Distributed Stochastic Neighbouring Entities for dimensionality reduction, i.e. we brought down higher dimensions to 2- dimensional X-Y plane.



```
#t-distributed stochastic neighbouring entities
#Dimensionality reduction, visualization for high dimensional dataset
#We use TSNE to reduce the dimensions of the data, bring down higher dimensions to 2D, i.e. x-y plane
from sklearn.manifold import TSNE

tsne_data = TSNE(verbose=1, perplexity=100, random_state=42) # computing tsne
X_embedded_data = tsne_data.fit_transform(X.toarray()) #transforming the data in array
```

4.2 Used SpaCy pattern Matching to identify Risk factors associated

- Used SpaCy for Python(an Open source Natural language processing library) that has a rule matching engine called the “Matcher” that operates over a set of tokens.
- Loaded the language class with a predefined English model “ en_core_web_sm” which is then used to instantiate Matcher Class Object.
- Spacy pattern matcher that operates over a set of tokens and identifies risk factors in the articles

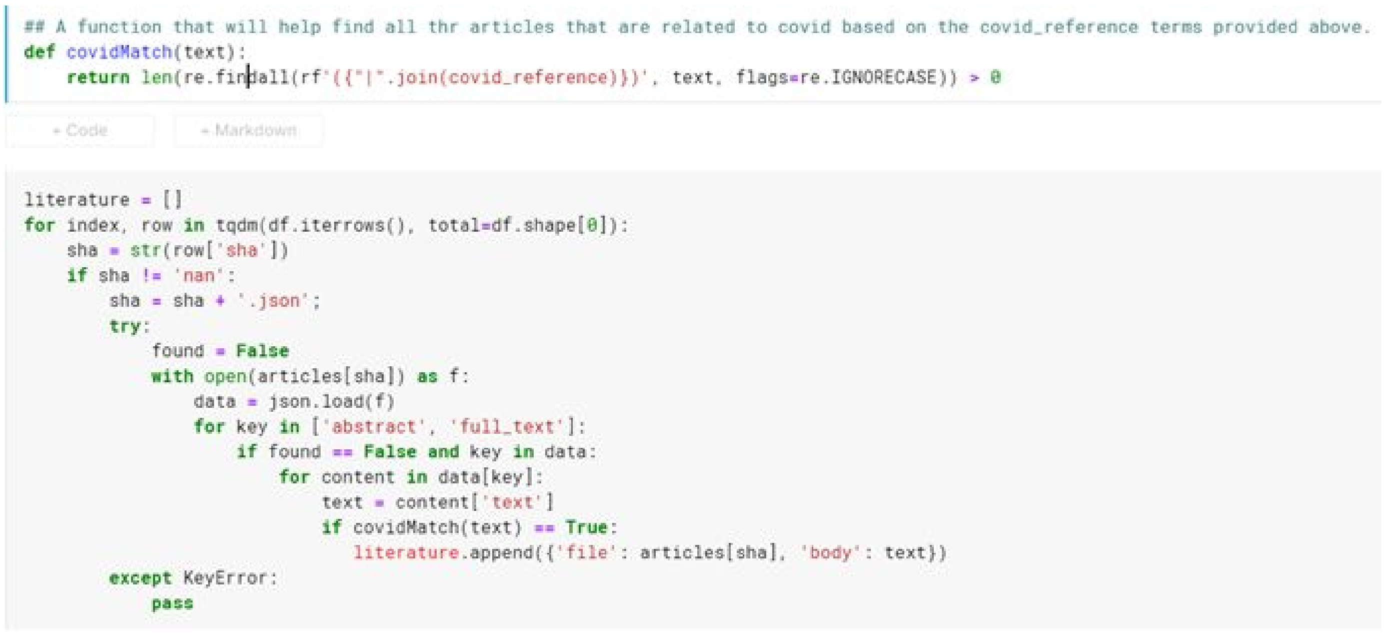


```
## Defining patterns that will put together a rule to identify the risk factors

patterns = {
    "Term Matcher": lambda term: [{'LOWER': t} for t in term.split(' ')],
    "Terms Matcher": lambda terms: [{"LOWER": {"IN": terms } }]
}
```

- Data preprocessing

  Created a new literature data set by parsing through all articles and extracting only the information from “abstract” and “full text body.”

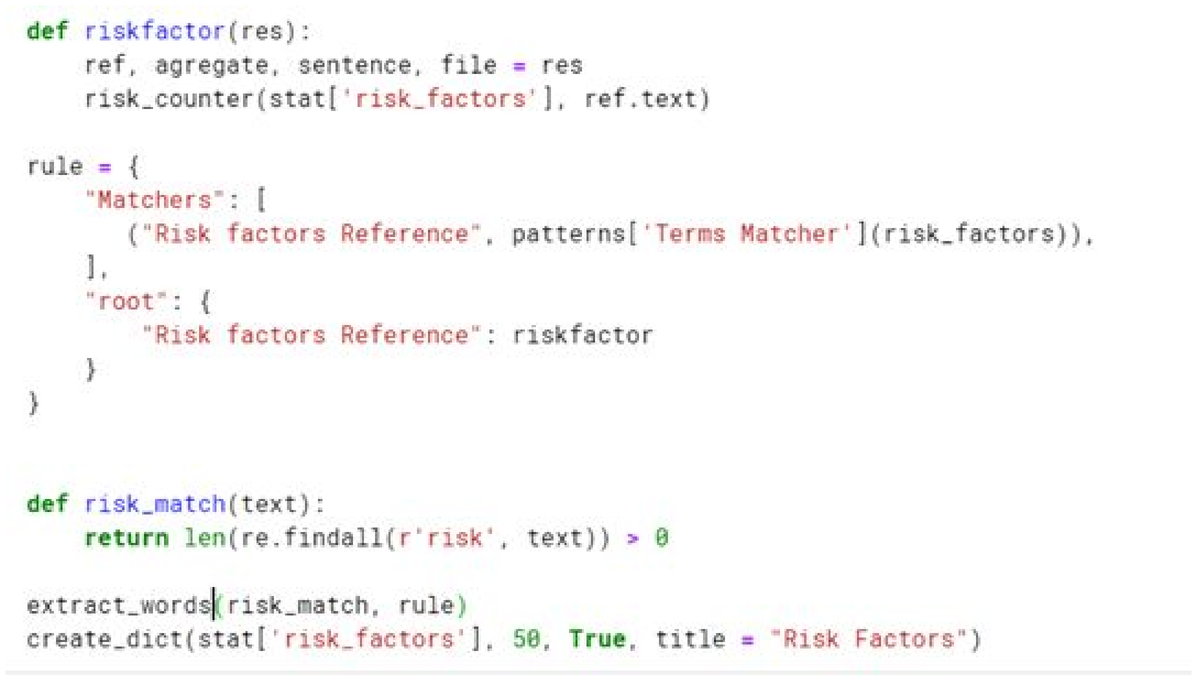
  Filtering out articles that do not have covid references in it.



```
## A function that will help find all thr articles that are related to covid based on the covid_reference terms provided above.
def covidMatch(text):
    return len(re.findall(rf'({"|".join(covid_reference)})', text, flags=re.IGNORECASE)) > 0
```

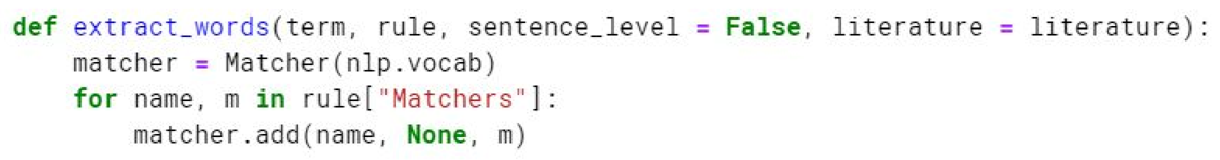
```
literature = []
for index, row in tqdm(df.iterrows(), total=df.shape[0]):
    sha = str(row['sha'])
    if sha != 'nan':
        sha = sha + '.json';
        try:
            found = False
            with open(articles[sha]) as f:
                data = json.load(f)
                for key in ['abstract', 'full_text']:
                    if found == False and key in data:
                        for content in data[key]:
                            text = content['text']
                            if covidMatch(text) == True:
                                literature.append({'file': articles[sha], 'body': text})
        except KeyError:
            pass
```

- ❖ Defined a matcher that will search for virus references and risk factors in all articles and in the known terms library defined. Uses the keyword “risk” to search for patterns in articles.



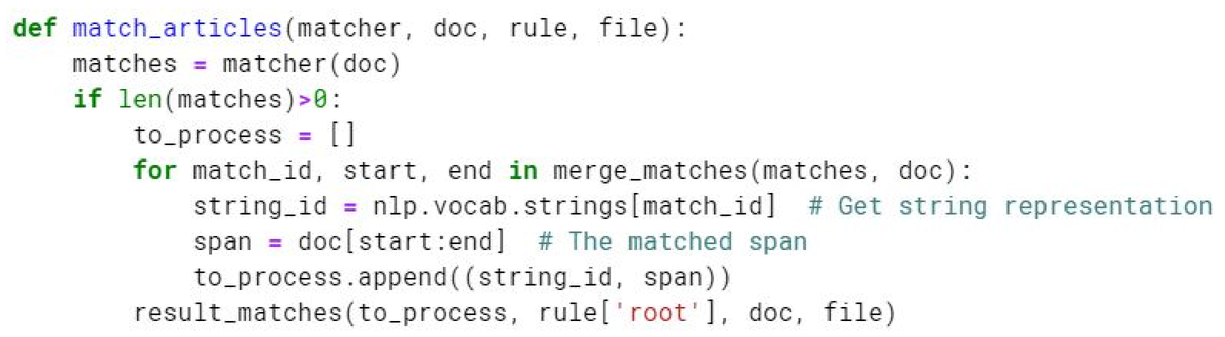
```python
def riskfactor(res):
    ref, agregate, sentence, file = res
    risk_counter(stat['risk_factors'], ref.text)

rule = {
    "Matchers": [
        ("Risk factors Reference", patterns['Terms Matcher'](risk_factors)),
    ],
    "root": {
        "Risk factors Reference": riskfactor
    }
}


def risk_match(text):
    return len(re.findall(r'risk', text)) > 0

extract_words(risk_match, rule)
create_dict(stat['risk_factors'], 50, True, title = "Risk Factors")
```

- ❖ Executing the rule and adding patterns defined to the object to match patterns.
  1. Creating the matcher class object and adding pattern to the matcher object created



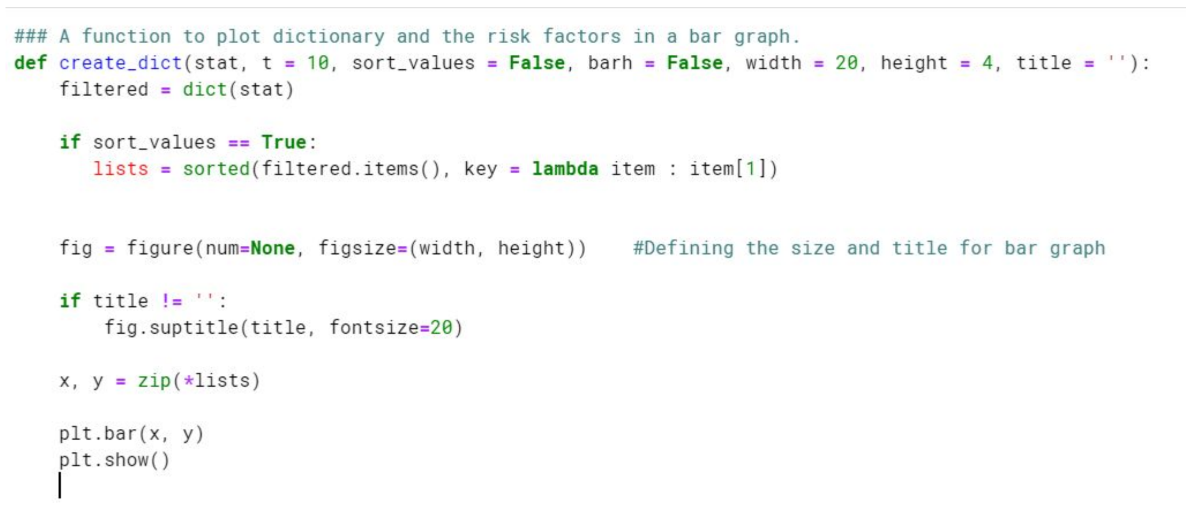
```python
def extract_words(term, rule, sentence_level = False, literature = literature):
    matcher = Matcher(nlp.vocab)
    for name, m in rule["Matchers"]:
        matcher.add(name, None, m)
```

  2. Converting text articles to document object using **doc = nlp(text)**
  3. We call the matcher object with the document object as the argument. We get a span class with all the matched results.



```
def match_articles(matcher, doc, rule, file):
    matches = matcher(doc)
    if len(matches)>0:
        to_process = []
        for match_id, start, end in merge_matches(matches, doc):
            string_id = nlp.vocab.strings[match_id]  # Get string representation
            span = doc[start:end]  # The matched span
            to_process.append((string_id, span))
        result_matches(to_process, rule['root'], doc, file)
```

❖ Plotting the graph for Risk factors



```
### A function to plot dictionary and the risk factors in a bar graph.
def create_dict(stat, t = 10, sort_values = False, barh = False, width = 20, height = 4, title = ''):
    filtered = dict(stat)

    if sort_values == True:
       lists = sorted(filtered.items(), key = lambda item : item[1])


    fig = figure(num=None, figsize=(width, height))    #Defining the size and title for bar graph

    if title != '':
        fig.suptitle(title, fontsize=20)

    x, y = zip(*lists)

    plt.bar(x, y)
    plt.show()
```

4.3 Identify the top papers that identify the risk factors using the Neural language model.

We use the feed forward Neural Network to identify the top 5 papers associated with the risk factors query.

Link:https://tfhub.dev/google/nnlm-en-dim128/2.

We link the above model into our python code.The module takes a batch of sentences in a 1-D tensor of strings as input.Model splits on the basis of spaces. Word embeddings are combined into sentence embeddings using sqrtn combiner.

In order to do this, we divide the papers into a list of sentences and put it into a sentence list. Initially we will apply the FileReader function which is also defined and used in module 4.1. We then read all of the data from the Jsons and filtered out any values that are null or empty adding titles and breaks wherever needed. Next, we divide it into sentences and ultimately put it into a list that is suitable to feed it into the NLM model.

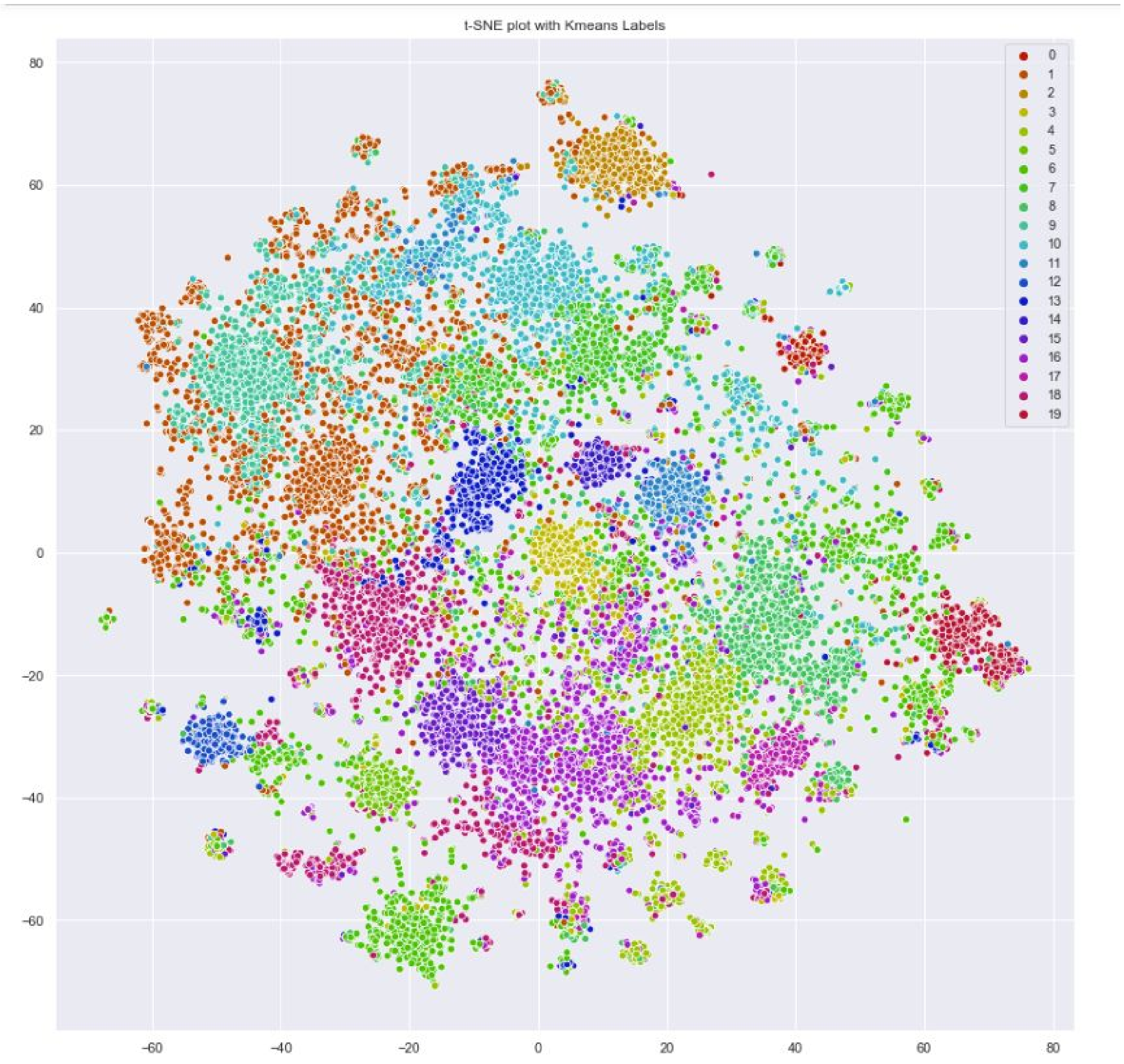
We will then embed the sentences using the feed-forward Neural-Net Language Models and we will also query into 128 dimensional embedding vectors.

We will then check for the similarity between the query and the sentences using cosine similarity. The top 5 results are displayed.

5. Evaluation and Results

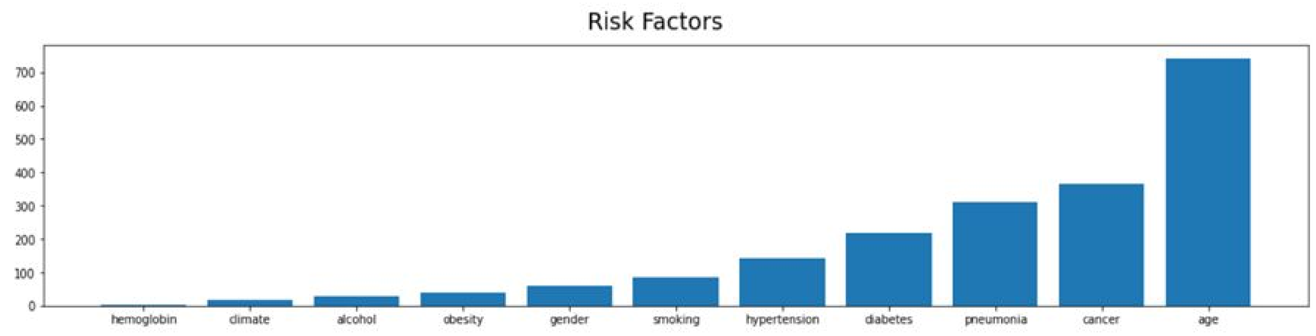
Results and Findings

5.1 The results obtained after clustering the data using K-means and t-SNE and a scatter plot visualizing it.

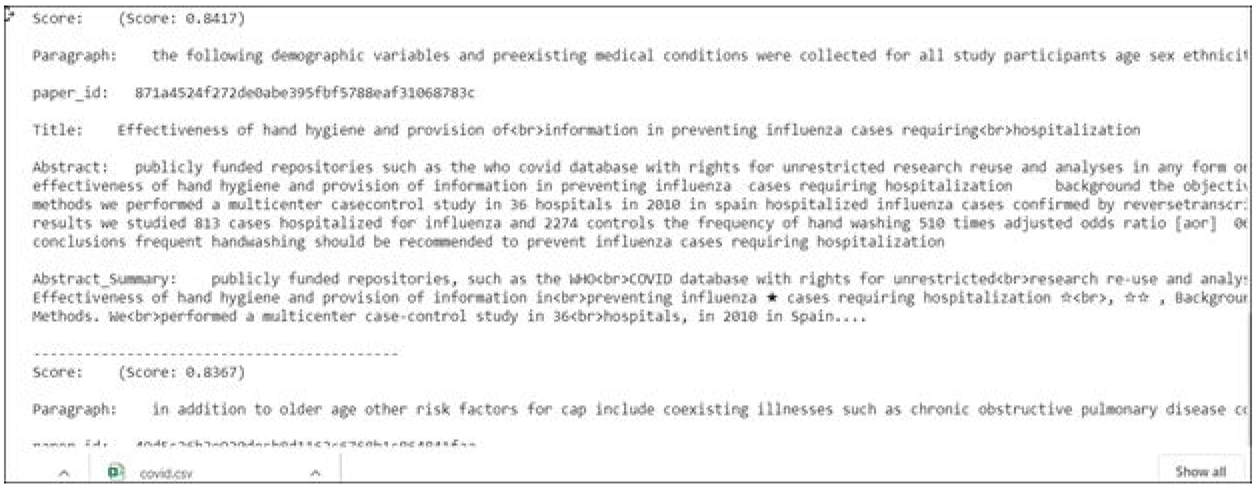


5.2

Results obtained from spaCy pattern matching are a list of risk factors and a bar graph visualizing it.



5.3 Output of the top papers that identify the risk factors using Neural Language model.



```
Score:    (Score: 0.8417)

Paragraph:    the following demographic variables and preexisting medical conditions were collected for all study participants age sex ethnicit

paper_id:   871a4524f272de0abe395fbf5788eaf31068783c

Title:    Effectiveness of hand hygiene and provision of<br>information in preventing influenza cases requiring<br>hospitalization

Abstract:   publicly funded repositories such as the who covid database with rights for unrestricted research reuse and analyses in any form or
effectiveness of hand hygiene and provision of information in preventing influenza  cases requiring hospitalization     background the objectiv
methods we performed a multicenter casecontrol study in 36 hospitals in 2010 in spain hospitalized influenza cases confirmed by reversetranscr
results we studied 813 cases hospitalized for influenza and 2274 controls the frequency of hand washing 510 times adjusted odds ratio [aor]  0
conclusions frequent handwashing should be recommended to prevent influenza cases requiring hospitalization

Abstract_Summary:    publicly funded repositories, such as the WHO<br>COVID database with rights for unrestricted<br>research re-use and analys
Effectiveness of hand hygiene and provision of information in<br>preventing influenza ★ cases requiring hospitalization ☆<br>, ☆☆ , Backgrou
Methods. We<br>performed a multicenter case-control study in 36<br>hospitals, in 2010 in Spain....

-------------------------------------------
Score:    (Score: 0.8367)

Paragraph:    in addition to older age other risk factors for cap include coexisting illnesses such as chronic obstructive pulmonary disease c
```

6. Conclusions and Future Work

6.1 Conclusions

- **Conclusions obtained after clustering the data are:**
  - Different clusters are grouped together based on body texts.
  - All research papers are clustered based on 20 labels that were labelled using the K-means algorithm.
  - It will be easier for the scientists and doctors to find relevant research papers.

- **Conclusions from Risk Factor visualization obtained by performing Spacy Pattern matching are as below :**
    - Factors like climate , alcohol , obesity and gender(Male /Female) are less likely to affect the spread of coronavirus.
    - People who smoke or those suffering from hypertension or diabetes are more likely to get affected by the virus.
    - People having pneumonia or cancer are at high risk and have a higher chance of getting affected.
    - An aged person has the highest risk of getting affected with the virus.

Conclusions of the top papers that identify the risk factors using neural Net language model

- The model will now help the scientists to go through the papers to identify the top papers to refer as per their required query.

6.2 Limitations

- The clustered data do not show any specific data, we have created labels based on 20 numbers.
- We are yet to identify a method that will help us identify which is a better model to search for top N results.

6.3 Potential Improvements /Future Work

- For future work, we will define these labels based on different topics.
- We will use Latent Dirichlet Allocation Model to define these 20 topics so that it will be easier for the doctors and scientists to identify the clusters just by scrolling on them.
- Use spaCy pattern matcher to perform more complicated pattern matching in articles by providing a list of terms to be searched. Also observing the results without providing the set of terms.
- For the part 3, we also plan to use sentence embeddings using TF-IDF and then find a way to compare the two methods and find out which method provides a better result.