In [1]:

```
import pandas as pd
import numpy as np
import json
import math
import glob
```

In [2]:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from nltk.stem.snowball import SnowballStemmer
from scipy.spatial import distance
from matplotlib import pyplot as plt
```

In [3]:

```
import ipywidgets as widgets

from IPython.display import Image
from IPython.display import display, HTML
```

In [4]:

```
df = pd.read_csv('C:\\Users\\asus\\Covid Data\\metadata.csv')
doc_paths = 'C:\\Users\\asus\\Covid Data\\pdf_json.json'
df.sha.fillna("", inplace=True)

#get text for articles that are available
```

```
C:\Users\asus\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:316
5: DtypeWarning: Columns (1,4,5,6,13,14,15,16) have mixed types.Specify dtype o
ption on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

In [5]:

```
def get_text(sha):
    if sha == "":
        return ""
    document_path = [x for x in doc_paths if sha in x]
    if not document_path:
        return ""
    with open(document_path[0]) as f:
        file = json.load(f)
        full_text = []
        #iterate over abstract and body part
        for part in ['abstract', 'body_text']:
            # iterate over each paragraph
            for text_part in file[part]:
                text = text_part['text']
                # remove citations from each paragraph
                for citation in text_part['cite_spans']:
                    text = text.replace(citation['text'], "")
                full_text.append(text)

        return str.join(' ', full_text)
```

In [6]:

```
%time df['text'] = df.apply(lambda x: get_text(x.sha), axis=1)
```

```
Wall time: 9.46 s
```

In [7]:

```
stemmer = SnowballStemmer("english")
analyzer = CountVectorizer().build_analyzer()

def preprocess(doc):
    doc=doc.lower()
    return str.join(" ", [stemmer.stem(w) for w in analyzer(doc)])

def preprocess_row(row):
    text = str.join(' ', [str(row.title), str(row.abstract), str(row.text)])
    return preprocess(text)
```

In [8]:

```
%time df['preprocessed'] = df.apply(lambda x: preprocess_row(x), axis=1)
```

```
Wall time: 25min 45s
```

In [9]:

```
cv = CountVectorizer(max_df=0.95, stop_words='english')
%time word_count = cv.fit_transform(df.preprocessed)
tfidf_tr = TfidfTransformer(smooth_idf=True, use_idf=True)
%time tfidf_tr.fit(word_count)
```

```
Wall time: 1min 14s
Wall time: 231 ms
```

Out[9]:

```
TfidfTransformer()
```

In [29]:

```
def get_word_vector(document):
    w_vector = tfidf_tr.transform(cv.transform([document]))
    return w_vector
```

In [20]:

```
%time df['word_vector'] = df.preprocessed.apply(get_word_vector)
```

```
Wall time: 52min 36s
```

In [30]:

```
df.iloc[1].word_vector.data
```

Out[30]:

```
array([291298, 280730, 269169, 266548, 256016, 252893, 241753, 240632,
       237200, 229073, 228644, 227893, 227418, 226677, 217178, 216010,
       215990, 208635, 208442, 205563, 196853, 188228, 183187, 179915,
       176332, 176108, 170079, 167616, 167109, 159598, 150808, 148498,
       147793, 147118, 147102, 146249, 146042, 133105, 122338, 116867,
       116301, 109224, 104946, 101745,  97055,  93182,  88773,  88539,
        76432,  74944,  74519,  74227,  73749,  73701,  54398,  51118,
        50087,  44728,  42470,  39420,  35973,  30425], dtype=int32)
```

feature_names = cv.get_feature_names() def get_words_with_value(w_vector): return sorted([(feature_names[ind], val) for ind, val in zip(w_vector.indices, w_vector.data)], key=lambda x: x[1], reverse=True)

In [56]:

```
def calculate_distance_between_words_vectors(search_words_indices, search_vec, do
    document_vec = document_vector[0, search_words_indices].toarray()
    return distance.euclidean([search_vec], document_vec)
```

In [80]:

```
def display_friendly_results(df_result):
    display_columns = ["title", "doi", "pmcid", "authors"]
    display(df_result[display_columns].reset_index(drop=True))
```

In [58]:

```
topic="What do we know about COVID-19 ?"
search_vector = get_word_vector(preprocess(topic))
```

In [59]:

```
search_words_indices = search_vector.indices
search_vec = search_vector.data
```

In [61]:

```
distance_idx = calculate_distance_between_words_vectors(search_words_indices,sear
distance_idx
```

Out[61]:

```
1.0
```

In [62]:
```
distance_idx = df.apply(lambda x: calculate_distance_between_words_vectors(searc
```

In [82]:
```
relevant_indexes = distance_idx.sort_values().head(10).index
result_columns = ["title", "doi", "pmcid", "license", "authors"]
result = df[result_columns].iloc[relevant_indexes].fillna("")
```

In [83]:
```
display_friendly_results(result)
```

	title	doi	pmcid	authors
0	All about COVID-19 what do we know?			Kandel, Dipendra
1	COVID-19: What do we know?			Marshall, Steve; Duryea, Michael; Huang, Greg;...
2	COVID-19: What do we know?			Marshall, Steve; Duryea, Michael; Huang, Greg;...
3	COVID-19: Knowing the Data			Stewart, Mary W
4	COVID-19: Knowing the Data			Stewart, Mary W
5	What you should know about COVID-19 to protect...			Prevention, Centers for Disease Control and
6	COVID-19 management: What we need to know?			Dhanushkodi, Manikandan; Kulkarni, Padmaj
7	COVID-19—What we know and what we need to know...	10.1007/s00059-020-04929-9	PMC7179372	Maisch, Bernhard; Dörr, Rolf
8	COVID-19 and cardiovascular disease: What we k...	10.1016/j.yjmcc.2020.04.026	PMC7180349	Dhawan, Rahul; Gundry, Rebekah L.; Brett-Major...
9	COVID-19 and cardiovascular disease: What we k...			Dhawan, Rahul; Gundry, Rebekah L; Brett-Major,...

```
for i in df['word_vector']:
    if i[1]>=val:
        print(1)
```

search_vector = get_word_vector(ptopic)

word_frequency = dict(get_words_with_value(search_vector)) k=max(word_frequency, key=word_frequency.get) val=word_frequency[k]