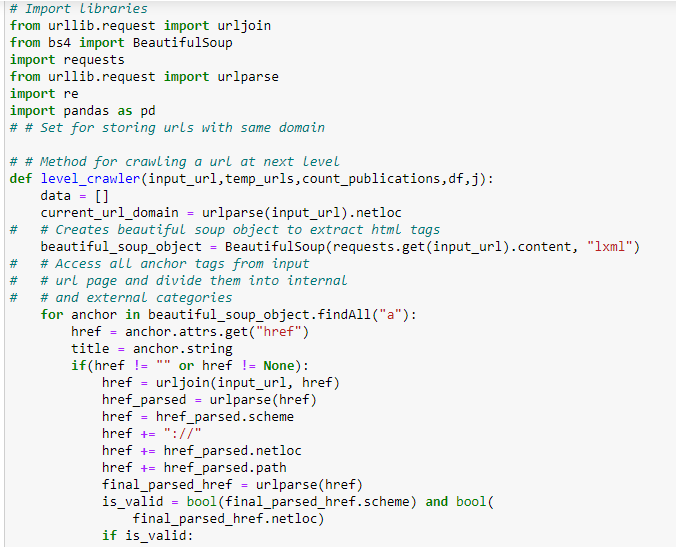
**Task 1. Search Engine**

**Part 1: Crawler:**

**1.2. Information collected about each publication (e.g. links, title, year, author or any additional part)**

The below code will crawl the input URL “<https://pureportal.coventry.ac.uk/en/organisations/school-of-economics-finance-and-accounting/publications/>“which is the Coventry university school of economics and accounting/ publications URL and fetch all the necessary details for each publications like publications links, publications title, publication date, author title, author name, abstract, content (which is abstract + author name).For all the publications I pulled all the above data and I added into a data frame and then at last the data frame is saved into a csv file. The csv file contains 638 rows and 7 columns. (Publications links, publications title, publication date, author title, author name, abstract, content (which is abstract + author name)).

**Screenshots of above code:**

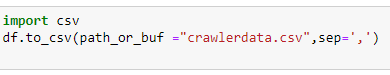






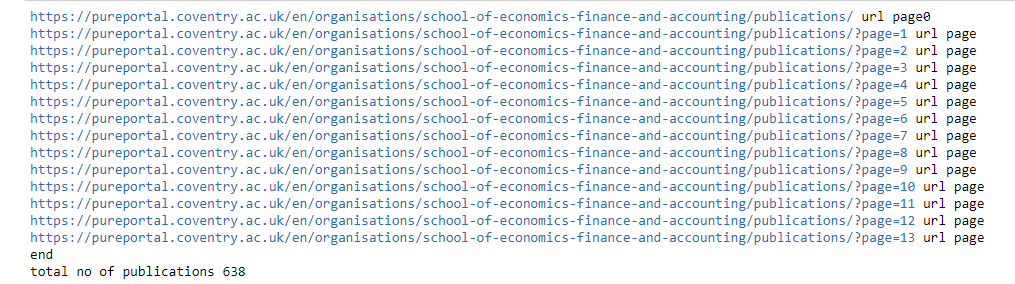




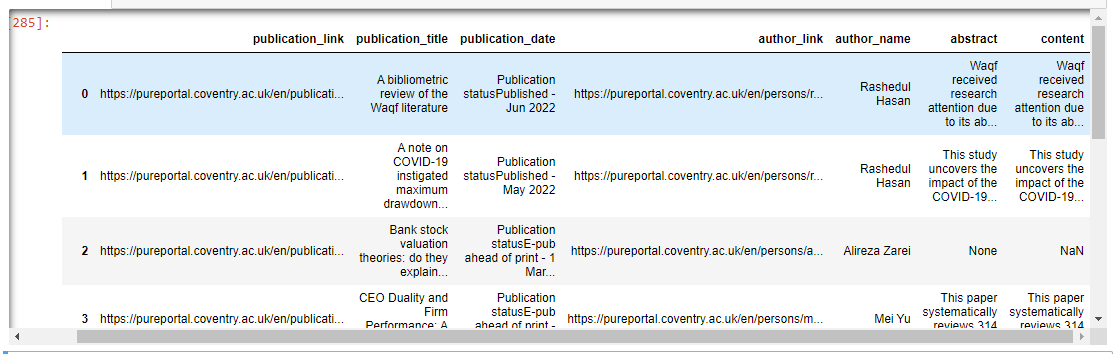


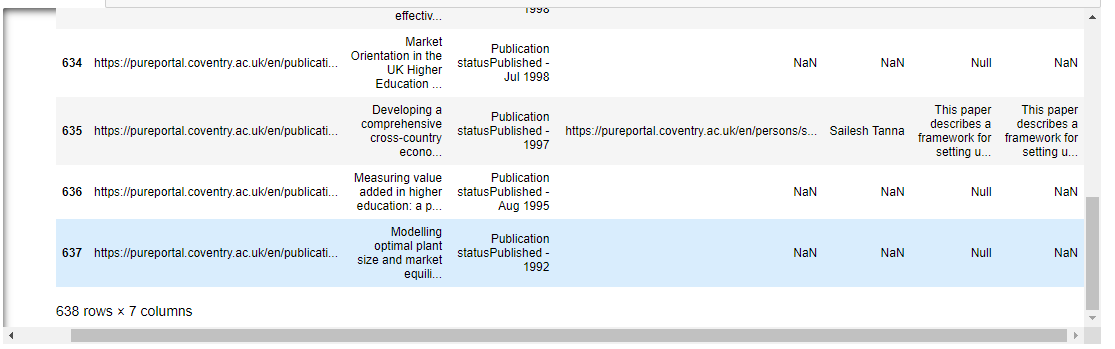
**Output screenshots:**

Below screenshots shows the output of crawler. There are total 638 publications in 13 web pages crawler crawls all the web pages. Added a print statement to check whether my crawler is accessing all web pages or not. Also added a count\_publications variable to count total number of publications being crawled and at the end of the program added a print statement to print the count\_publications.

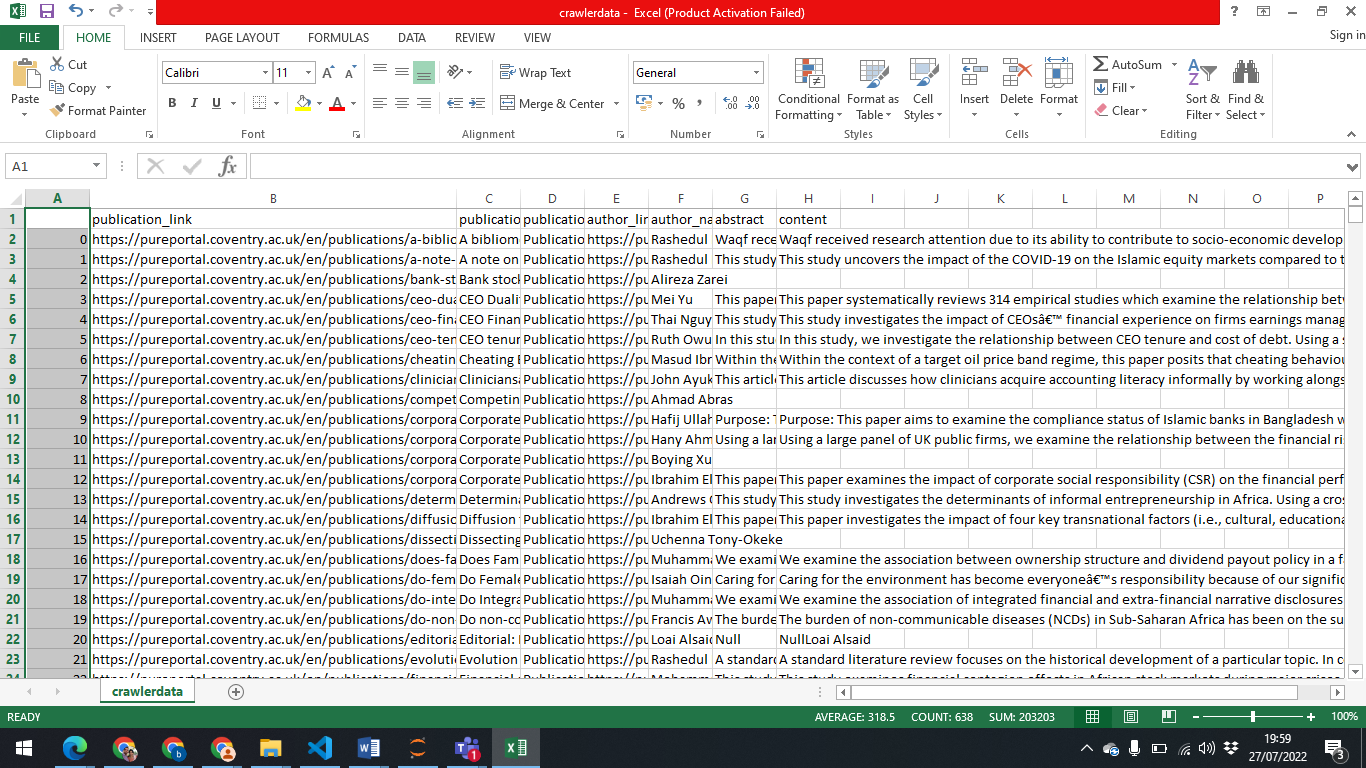


* Below screenshots shows the data frame containing all the publications details.





* Below screenshot shows all the publications data is being stored into a csv file. From now we can access the csv file every time instead of crawling all the time. Inorder to update the changes in the website, we can run the crawler program once in a week manually and update the data.

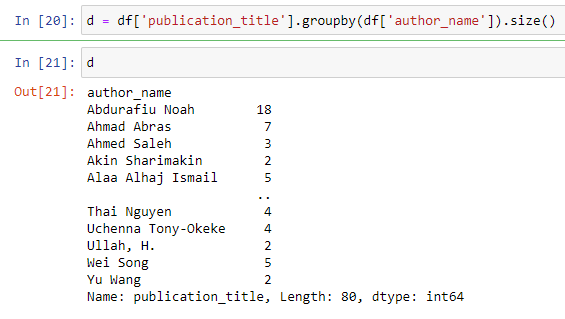


* 1. **Number of staff whose publications are crawled (approximately) and the maximum number of Publications per staff.**
* I fetched the below details from the data frame which I got after running the crawler program. Total 80 authors whose publications are crawled.





* Maximum number of publications per author



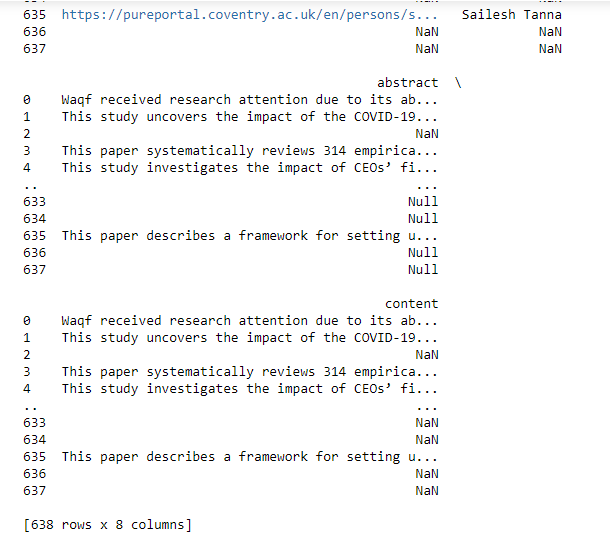
**1.3. Pre-processing tasks performed before passing data to Indexer.**

Removing stop words, tokenization and stemming are the three pre-processing tasks that are performed before passing data to indexer.

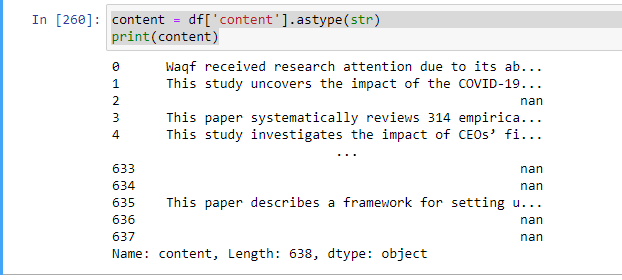
**Screen shots of the code and output:**

* Below screenshot shows reading data into a csv file and storing in a data frame df and prints the df. Data frame is of size 638 rows X 7 columns

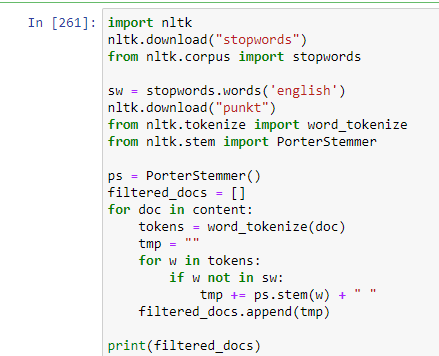




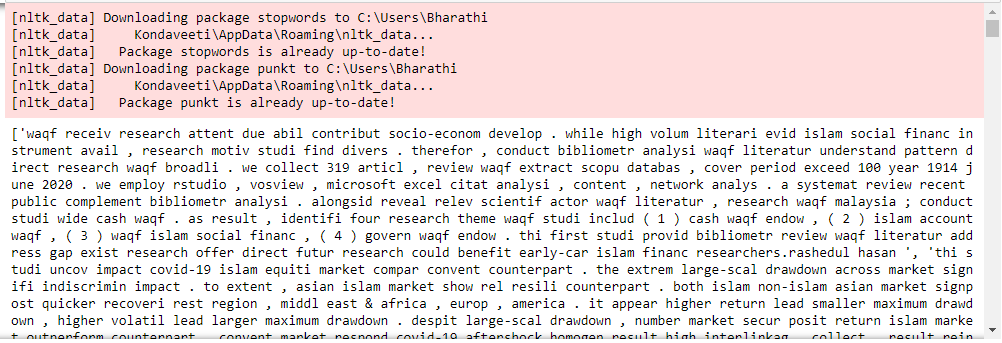
* Below screenshot shows the content column in df is stored in a local variable called content and prints the same to perform pre-processing tasks.



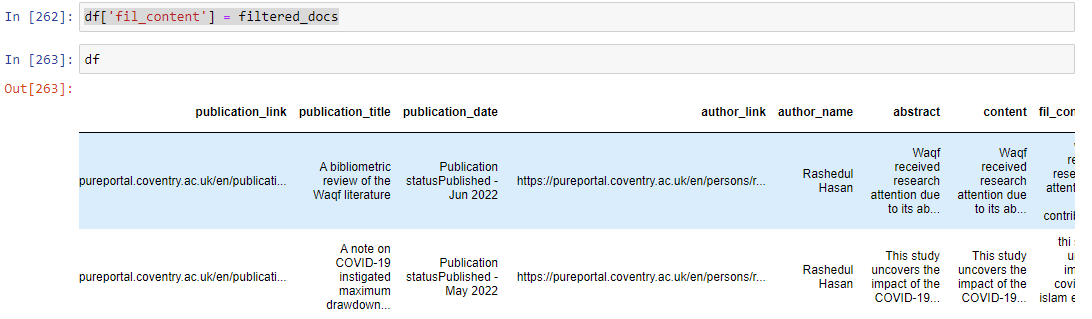
* The pre-processing tasks like stop words removal, tokenizing and stemming are being performed to the variable content.

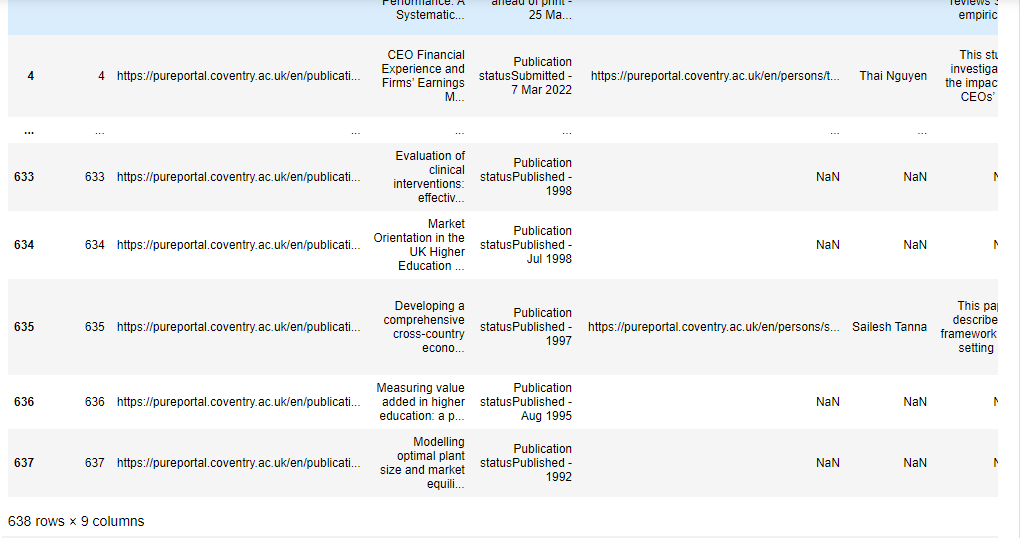


* The output is stored in filtered\_docs which is after processing content variable. Below screenshot shows the filtered\_doc list.



* The filtered\_docs is stored into a new column named ‘Fil\_content’ in the data frame.





1.4. The crawler operates, e.g. scheduled or run manually

* My crawler program will run manually by the programmer.

1.5. Explanation of crawler it works.

* iniitially the user gives input url “<https://pureportal.coventry.ac.uk/en/organisations/school-of-econ>omics-finance-and-accounting/publications/” and sends input to “level\_crawler” function.
* Created a data frame columns = ["publication\_link","publication\_title","publication\_date","author\_link","author\_name","abstract","content"] to store all publication details in the data frame.
* Then using beautiful\_soup\_object.findAll("a") it fetches the publications link and publications title and stores in a data frame and the each publications link is sent as input to “get\_author\_details” function.
* In get\_author\_details function, using the each publication link as input, to fetch author details I used soup.findAll('a',{'class':'link person'}) and got author name and author link. With this we will get all the author details.
* Inorder to fetch only SEFA authors, we need to check for (sefa == 'School of Economics, Finance and Accounting') and when it’s true only we will store the author details and author link in data frame.
* Then to get abstract I used (soup.find('div',{'class':'textblock'}) and fetched abstract from the each Publications link and sent to a data frame .
* Also fetched publication status and date using soup.find('tr',{'class':'status'}) for each publication and Store in a data frame.
* After fetching and writing all the details of first publication in to the first row of data frame , the code automatically fetches the 2nd publication details and so on till the end of the input page.
* After fetching all 50 publications in the first page, the input url will be updated automatically With the 2nd page URL using the pages logic. And it will continues to crawl all the pages till the last publication.
* After completing the crawler program, I added abstract, author\_name and publications\_title data and stored to the content column [‘content'] = ['abstract'] + ['author\_name']+[‘publication\_title’].
* Now the data frame is stored into an excel file named “crawlerdata.csv”. As all the crawled data is sent and stored in to csv file, from now we can access csv file and use the publications data.
* To perform pre-processing tasks, first we need to read the csv file in to a data frame df which contains all the publication details.
* I am implementing pre-processing like stop word removal, tokenization and stemming to the ‘content’ column which contains abstract and author name. So I extracted content column and stored in a list content.
* Then implemented the stop word removal, tokenization and stemming to the content variable and stored in the data frame into a new column name “fil\_content”. From now we can use the fil\_content column in the data frame.

**2. Indexer**

2.1. Whether you implemented the index or used Elastic Search (note that if Elastic Search is used you will lose the 15 marks for index construction, but the project becomes easier).

* In this project I implemented indexer on the fil\_content column of data frame df and stored in a dictionary.

2.2. If you implemented it, which data structure is used (for example, incidence matrix or inverted index)

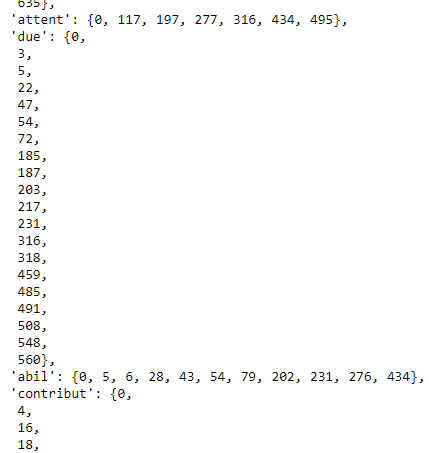
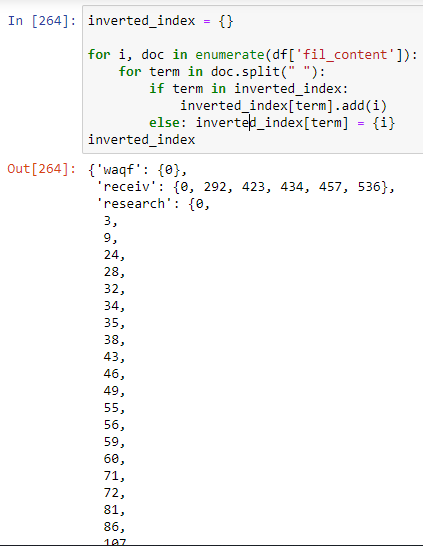
* In this project I implemented inverted index data structure

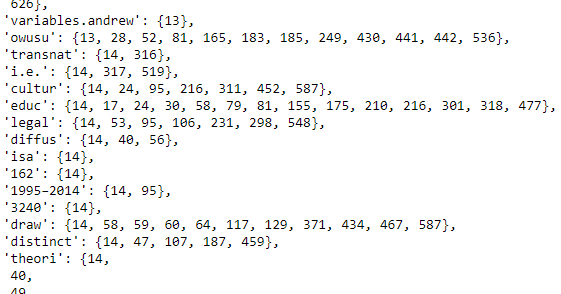
2.3. If you implemented it, whether it is incremental, i.e. it grows and gets updated over the time, or it is constructed from scratch every time your crawler is run

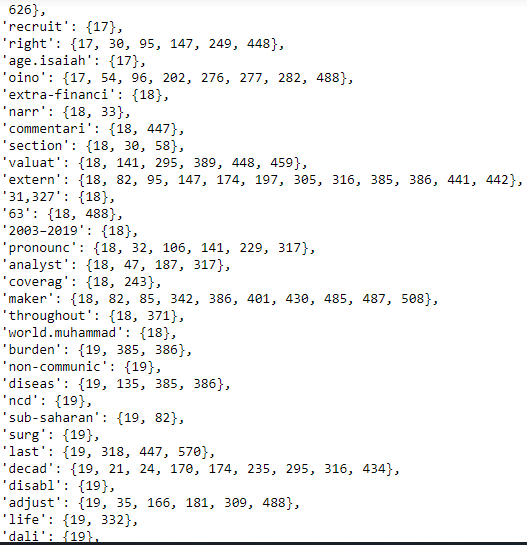
* For each time I run the crawler, the csv file gets updated and the inverted indexer code is executed manually. It is implemented from the scratch every time I run the crawler.

2.4. If you implemented it, show some part of its content (e.g. the constructed dictionary).

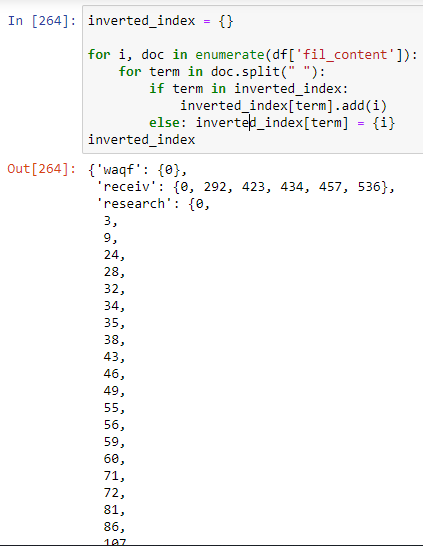
**Screenshots of code:**







2.5. Explanation of how it works



* The output from the crawler program and pre-processing tasks which is the csv file, the column ‘fil\_content’ is used as input to the indexer.
* An empty dictionary named inverted\_index is created. The for loop is implemented on fil\_content column for each row of fil\_content and splits by space. The term in each doc is checked in inverted index.
* If it’s already present in inverted index, it adds the row number (i.e., publication number) to the list of values in the dictionary where key is the term.
* If the term is not present in the inverted index dictionary, the term will be added as a key and value is the row number.
* By the end of the for loop, the inverted index dictionary is created having key as words of the fil\_content from all publications and values as list of rows in which the word exists.

3. Query processor

3.1. Pre-processing tasks are applied to a given query

For query processor, I applied pre-processing tasks like stop words removal, tokenization and stemming.

3.2. Do you only support Boolean queries (using AND, OR, NOT, etc.) or accept keywords like Google does (without any need for AND, OR, NOT etc.)

I did not used Boolean queries. My query processor accepts keywords like google does.

3.3. If Elastic Search is used, how you convert a user query to an appropriate query for Elastic Search

Did not used elastic search.

3.4. If Elastic Search is NOT used, whether or not you perform ranked retrieval; if yes, specify whether or not you used vector space and the method used to calculate the ranks

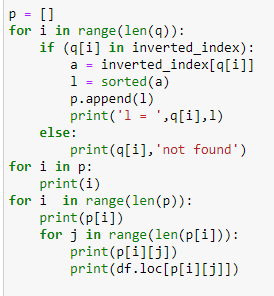
I did not perform ranked retrieval.

3.5. Demonstration of the running system (use screenshots in you report and run your software in your viva). You must run your system on numerous and various input queries to prove the accuracy and robustness of your system. For example, you must use appropriate queries to prove your system performs stop-word removal and stemming and ranked retrieval.

**Code and output screenshots of query processor.**

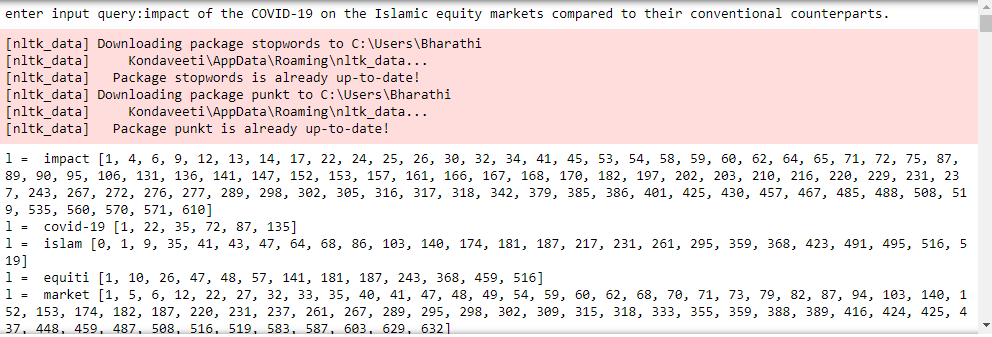
**Screenshots of code:**



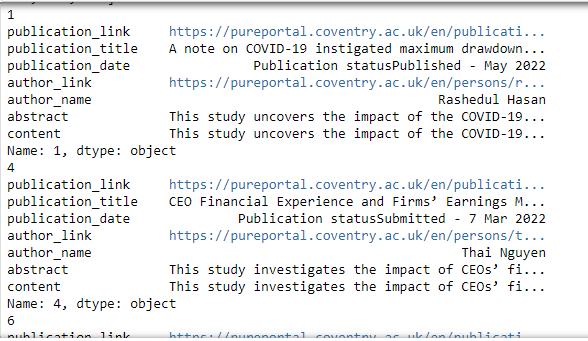


Output screenshots for long and short queries:

* **Long Query 1:**
* User entered query as input.



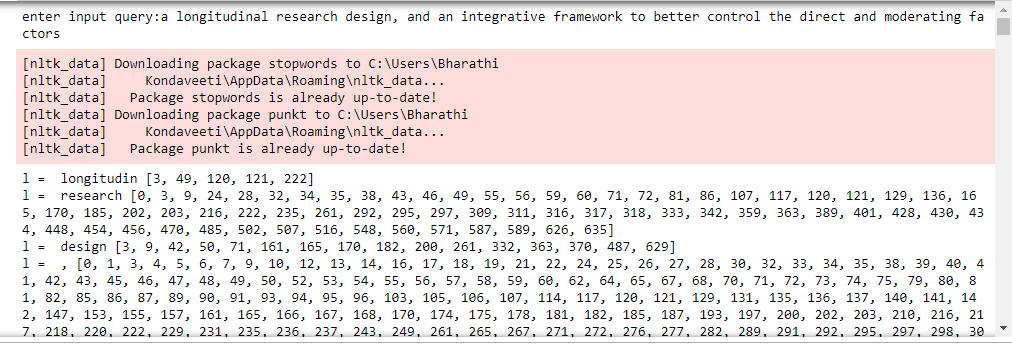
* My query processor fetches all the publications related to the query. The first result is the most related query.



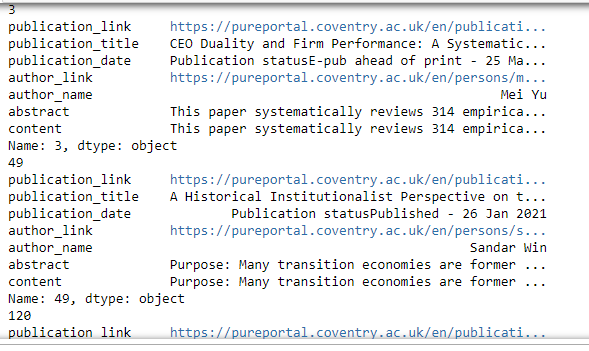
* Input query sentence is taken from the below publication and got the same publication as output to the query and all the below outputs are the related information.



* **Long Query 2:**
* User entered query as input.



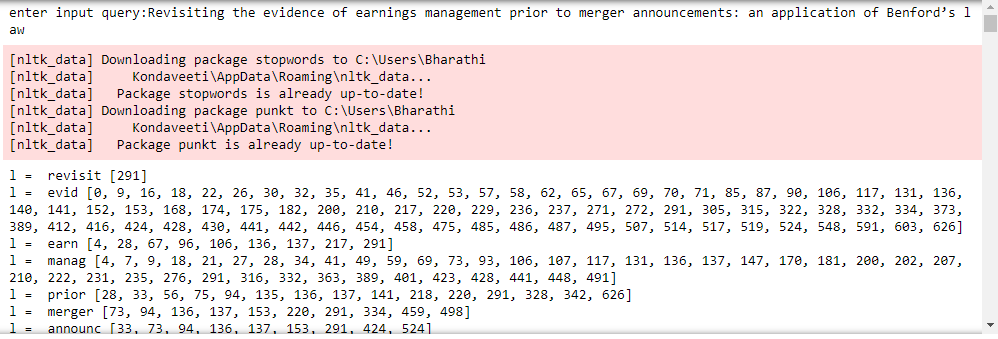
* My query processor fetches all the publications related to the query. The first result is the most related query.

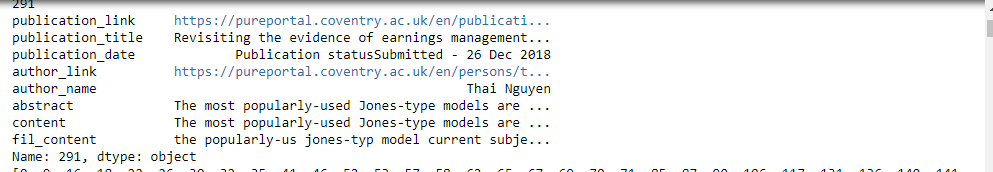


* Input query sentence is taken from the below publication and got the same publication as output to the query and all the below outputs are the related information.



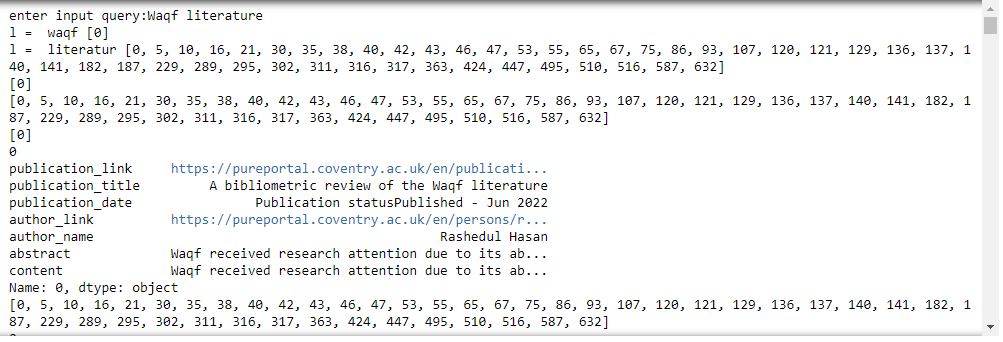
* Long Query 3:







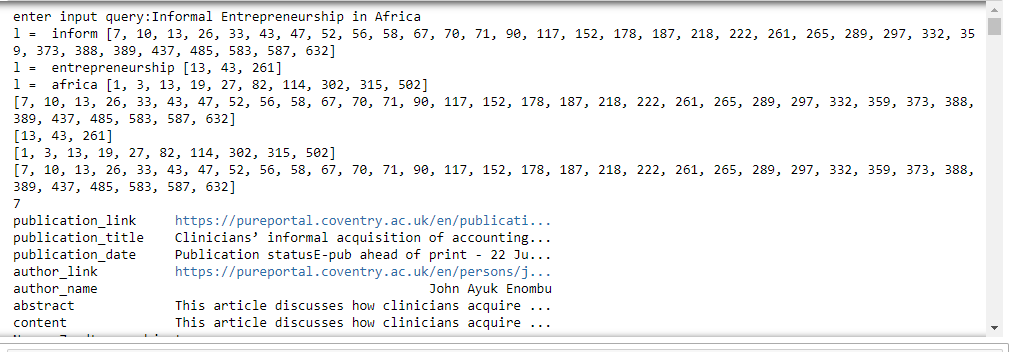
* **Short Query 1:**
* User entered query as input. My query processor fetches all the publications related to the query. The first result is the most related query.

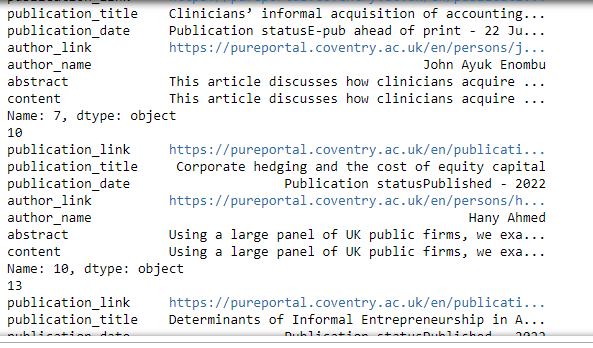
****

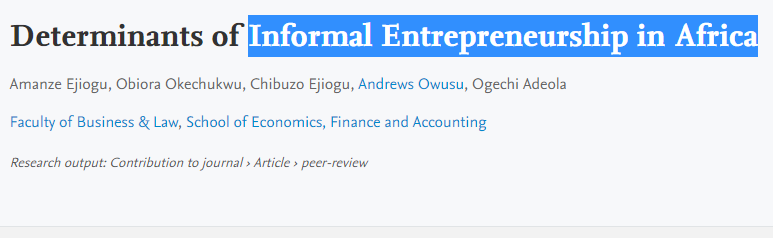
* Input query sentence is taken from the below publication and got the same publication as output to the query and all the below outputs are the related information.



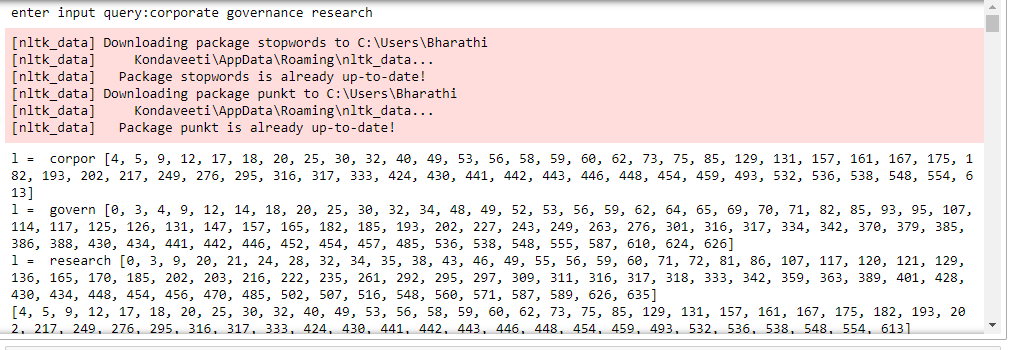
* **Short Query 2:**

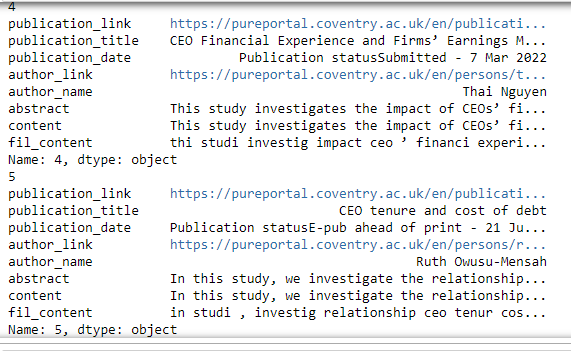
****

****

****

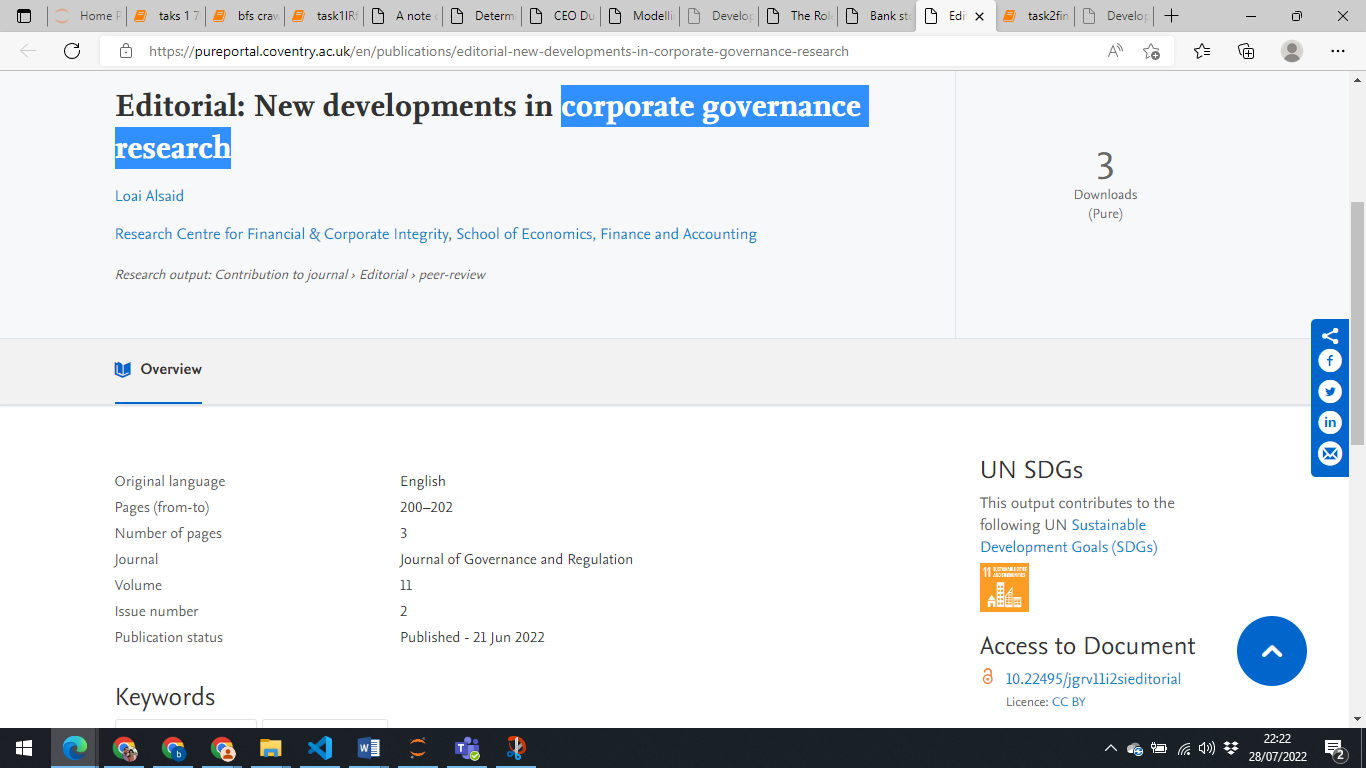
* **Short Query 3:**



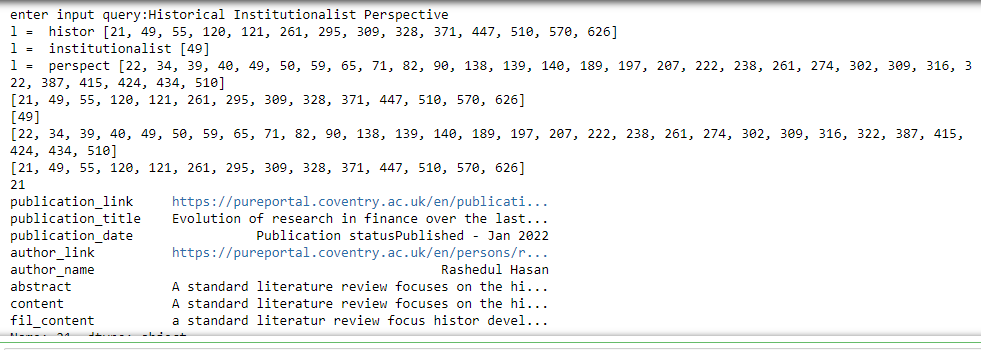


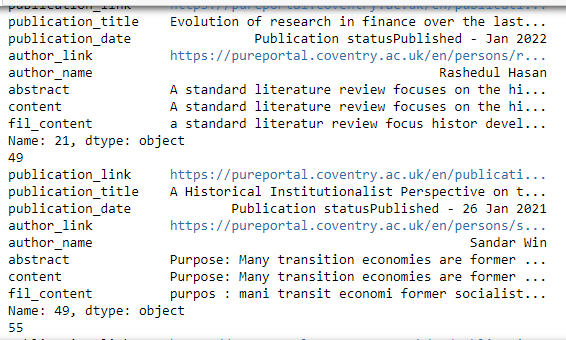


* Input is taken from the below publication with no abstract. As the fil\_content column contains abstract + publication name + author name , I can able to fetch details of publications.



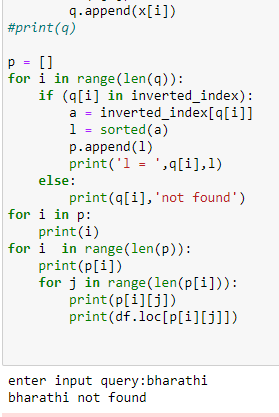
* **Short Query 3:**

****

****

****

* **Short Query 4:**
* When I gave a word which is not in any of the publications my query is printing not found message.

****

3.6. Brief explanation of how it works

I took input from the user whether it’s long or short query, the pre- processing tasks are applied to the input query and splits all the words in to a list and removes any empty spaces in the list as words. Then the final list with all words will be searched in the dictionary which is inverted\_index and gets the row numbers as list which are the publication details of the data frame. Then after retrieving row information I printed all the details of the publications. In my query processor, once the user enters the query it will automatically fetches the related publications. If the user enters a query which is not in any of the publications, user will get the output message “not found“.

**References:**

[Web crawling using Breadth First Search at a specified depth - GeeksforGeeks](https://www.geeksforgeeks.org/web-crawling-using-breadth-first-search-at-a-specified-depth/?ref=gcse)

<https://anvil.works/blog/how-to-build-a-search-engine>

<https://www.youtube.com/user/thenewboston/videos>

<https://github.com/Dev-Elie/Web-Crawler-for-Google-Scholar/blob/main/app.py>

<https://elhamamini.medium.com/how-to-build-a-vertical-search-engine-using-python-f09b137b5db>

<https://towardsdatascience.com/create-a-simple-search-engine-using-python-412587619ff5>

<http://aakashjapi.com/fuckin-search-engines-how-do-they-work/>

<https://www.youtube.com/watch?v=bFrO8piASKg>

<https://www.scrapingbee.com/blog/crawling-python/>

<https://www.scrapingbee.com/blog/crawling-python/>

<http://www.cs.put.poznan.pl/alabijak/ezi/lab1/Lab1-Crawling-Python.pdf>

<https://github.com/wesdoyle/Javelin/blob/master/Notebooks/Inverted%20Index.ipynb>

Task 2. Document Clustering

Q1: How and how many input documents are collected

* I implemented web crawling using Beautiful Soup to pull the text data that belongs to three different categories namely Sport, Health and Politics.
* I manually collected BBC news website links from the three categories and gave inputs in my crawler function to pull the data automatically from those links and stored all the documents (each document is atleast one sentence) in a list and names as docs.
* Total number of documents collected are above 110.

Q2: Which document clustering method (e.g. K-means with appropriate K value) has been used and how its performance is measured?

* Document clustering is a sub area of data clustering which includes concepts from information retrieval, natural language processing, and machine learning. The main aim of document clustering method is find out natural groupings of documents from a given collection of documents.
* The main goal is to minimize the computational overheard by creating more accurate clusters. There are many kinds of techniques for achieving these desired properties. The two main algorithms that are used in clustering are Hierarchical clustering and K-means clustering techniques. Hierarchical clustering is slower than K-means and sometimes combination of these two also used for good results.
* Hierarchical clustering techniques produce a nested sequence of partitions or a cluster of hierarchy or tree of clusters. This structure is also called as dendrogram. In this structure every node has child and sibling clusters. The main advantages of hierarchical clustering are their flexibility and ease of handling any forms of similarity. But they suffer from vagueness in the termination criteria.
* K-means is a partitioning relocation clustering method which divides data into several subsets. When we are using K-means we are using a centroid which is the mean value of all points within the cluster. This centroid represents the cluster formed and this helps the K-means methods to produce clusters in a faster rate than hierarchical methods.
* K-means provides a faster way to create cluster from a set of random documents. It calculates the vector value for each document from the vector space and based their value new clusters are formed. Since it uses the TDIDF and cosine measure the final produced clusters are always good in terms of both intra and inter cluster similarity
* So I have used the K-Means algorithm here to generate clusters. K-means clustering is a type of unsupervised learning method, which is used when we don’t have labelled data as in our case, we have unlabelled data (means, without defined categories or groups). The goal of this algorithm is to find groups in the data, whereas the no. of groups is represented by the variable K. The data have been clustered on the basis of high similarity points together and low similarity points in the separate clusters.
* Document clustering can be split in to 3 steps. Pictorial representation and the explanation of the same is shown in the below table.

|  |  |
| --- | --- |
|  | Cleaning and tokenizing data: this involves lowercasing text, removing non alpha numeric characters/stop words, and stemming the data.  Generating Vector representation of the documents: This concerns with mapping the documents from words into numerical vectors. We can use bag-of-words model (CountVectorizer) or TF.IDF (TfidfVectorizer) to construct vector. Here we used TF.IDF to construct vector.  Applying K-mean algorithm on the document vectors: This requires selecting and applying clustering algorithm to find the best possible groups using the document vectors. |

* Performance Measured: Evaluating the performance of a clustering algorithm is not as trivial as counting the number of errors or the precision and recall of a supervised classification algorithm. In particular any evaluation metric should not take the absolute values of the cluster labels into account but rather if this clustering define separations of the data similar to some ground truth set of classes or satisfying some assumption such that members belong to the same class are more similar than members of different classes according to some similarity metric.
* Given the knowledge of the ground truth class assignments labels\_truth and our clustering algorithm assignments of the same samples labels\_pred the **(adjusted or unadjusted) Rand index** is a function that measures the **similarity** of the two assignments, ignoring permutations. Perfect labelling is scored 1.0.
* I got rand index = 0.7142857142857143 in this task.

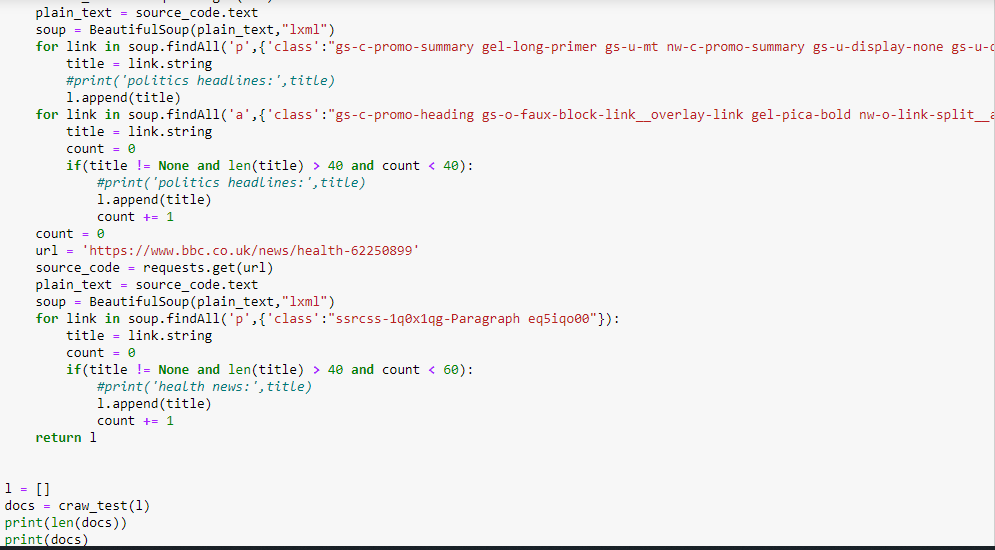
Q3: Which type of clustering is used (hierarchical/flat and hard/soft)

* For this task I used Hard clustering as each document belongs to only one cluster andflat clustering as a cluster won’t split itself into several documents.
* Two types of clustering’s
* Hard (document belongs to only one cluster) /
* Soft (document belongs to more than one cluster).
* Two types of clustering’s
* Flat (a cluster won’t split itself into several clusters)
* Hierarchical (a cluster itself split into two or more clusters)

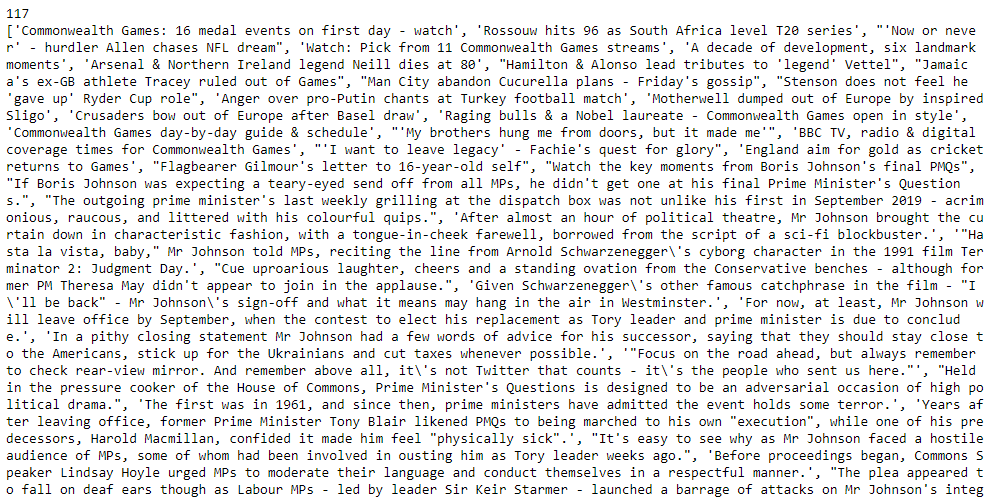
Q4: Screenshot and demonstration of its accuracy and robustness for numerous and various inputs

* Crawled BBC news website and pulled some text related to sports, health and politics. I stored all the Crawled text into a list. Each value in a list is a document. Total length of the list is 121, that means total 121 documents are collected.
* Below screenshot shows the code for crawling data from news websites.

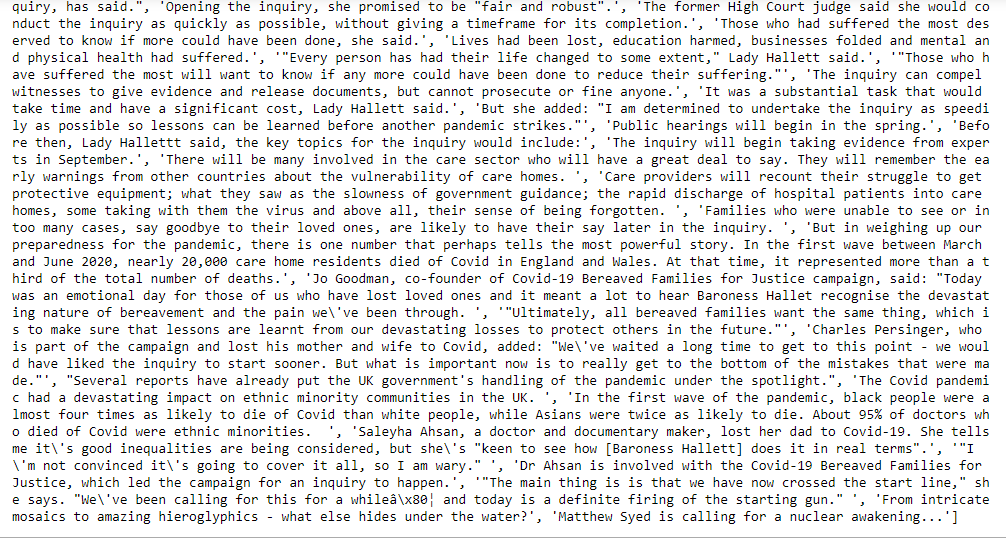




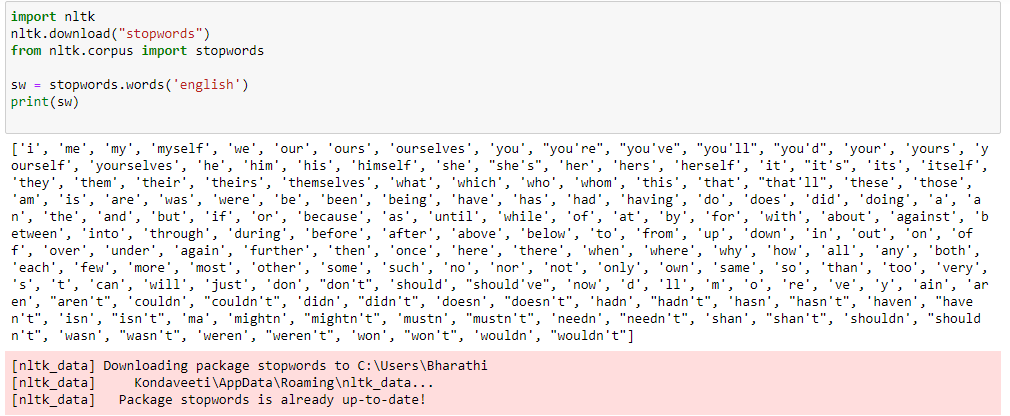
* Added print statements in the code to show the length of the list of documents and also printed the docs list. Below screenshots shows the output of the above code.
* In the below screenshot, it shows the starting of the list.



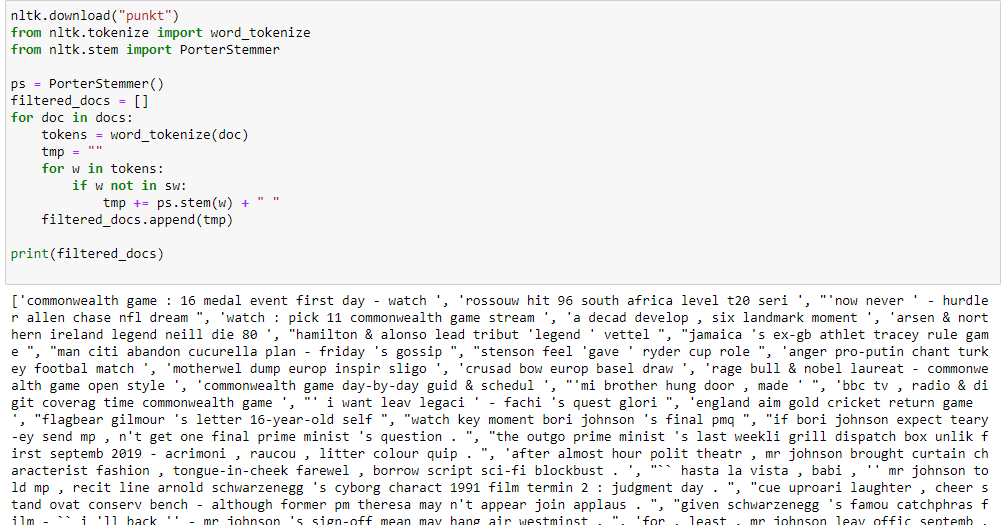
* Next screenshot shows the ending of the list.

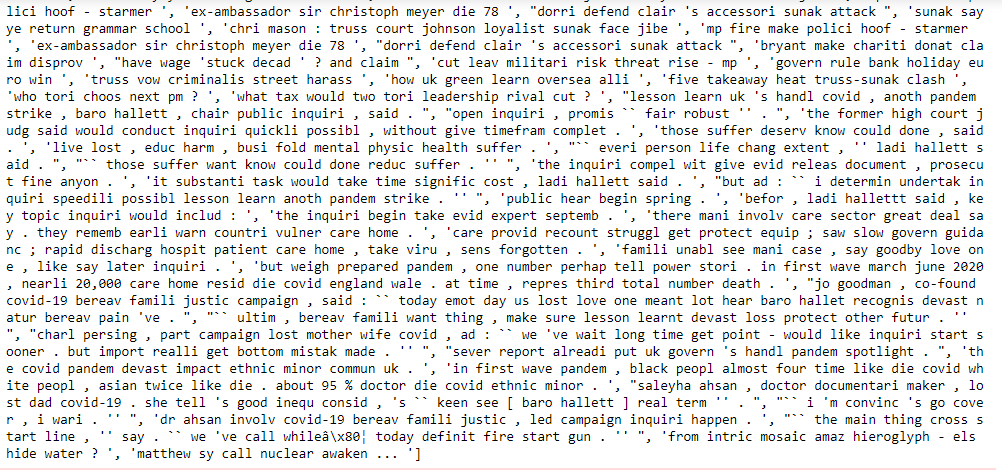


* Downloaded stop words

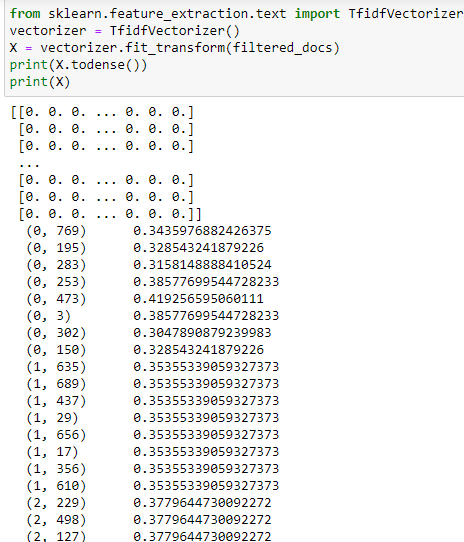


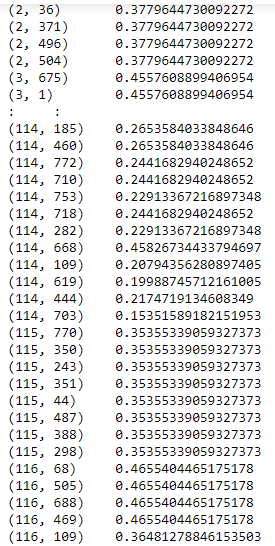
* Implemented tokenizer and stemming to the input list of docs.

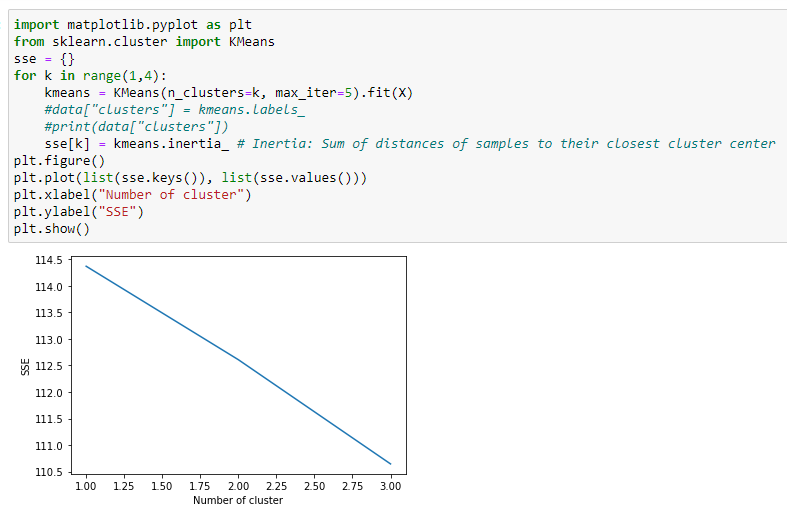


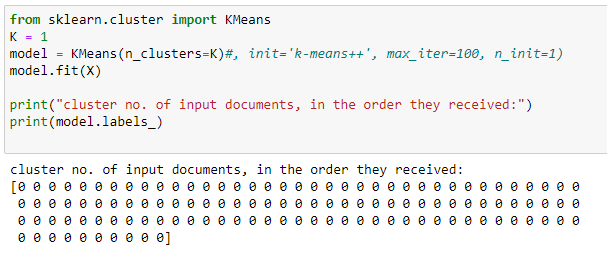


* Implemented TFIDF vectorizer to the above output docs.

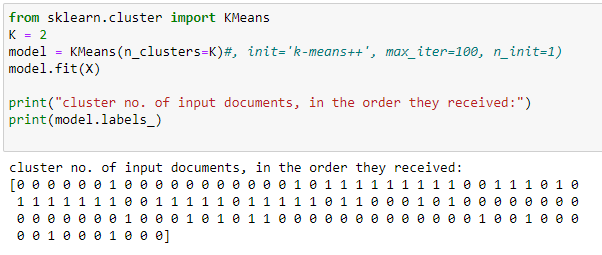




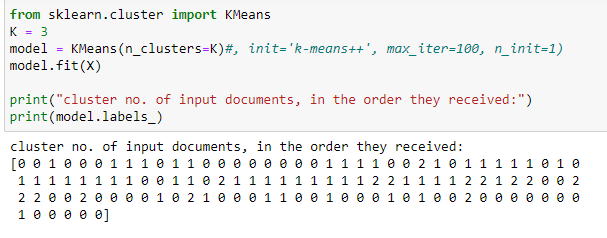
* Elbow method gives us an idea on what a good k number of clusters would be based on the sum of squared distance (SSE) between data points and their assigned clusters’ centroids. We pick k at the spot where SSE starts to flatten out and forming an elbow. 
* Implemented K-means to the TFIDF vectorizer output for k = 1 means all the input data is into one cluster ‘0’.



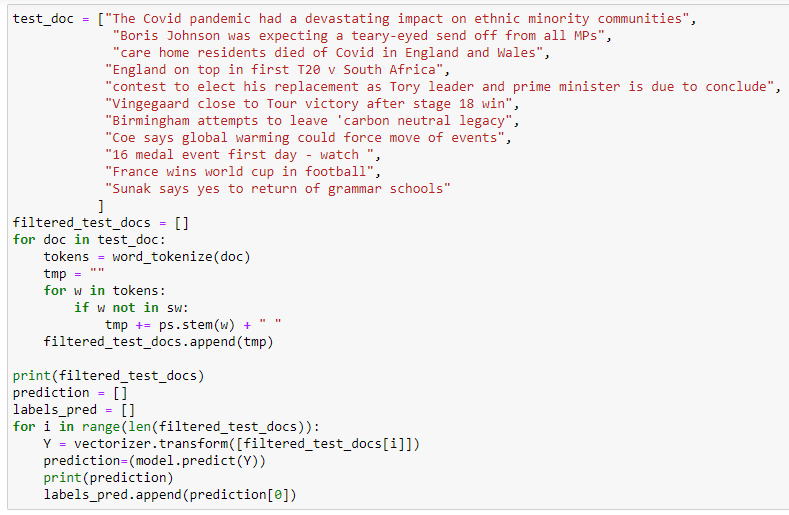
* Implemented K-means to the TFIDF vectorizer output for k = 2.it means all the input data is into two cluster ‘0 and 1’.

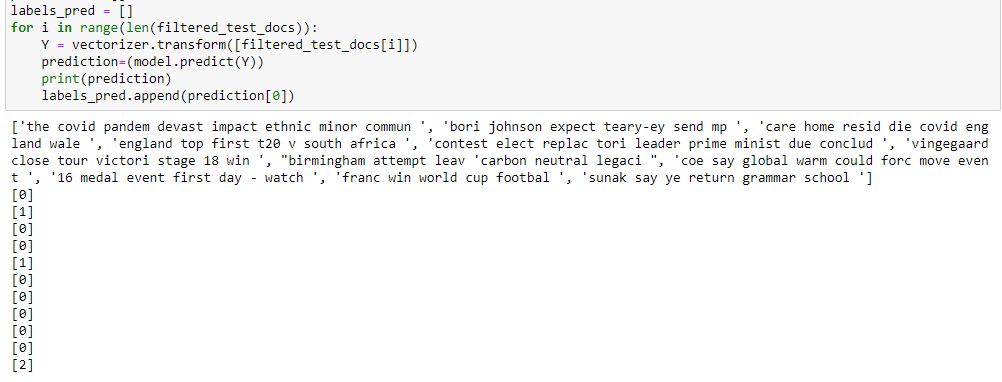


* Implemented K-means to the TFIDF vectorizer output for k = 2.it means all the input data is into two cluster ‘0, 1 and 2’.

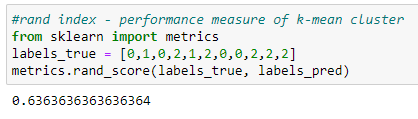


* Below code predicting the incoming text documents by using above K-means clustering Output. The list test\_doc is the input documents for the predicting to which cluster the input documents belong to. For the test\_doc list, I applied all the pre-processing tasks (removing stop words, tokenizing and stemming) which were applied to the clustering input list. Then applied vectorization and model prediction on the processed list. Then we will get the prediction of each document i.e.; to which cluster the document belongs to.





* To measure the performance of K-mean cluster there are many ways, here I calculated rand index to measure the performance of document clustering, I imported metrics from sklearn and applied rand index to get rand score. Input to the rand score are predicted output from above prediction and labels\_true which I manually gave input in a list of integer values to which cluster it belongs to. I got 0.714 as rand index. The value of rand index ranges from 0 to 1. Perfect rand index is 1.



5. Brief explanation of how k-means document clustering works.

* The main objective of the K-Means algorithm is to minimize the sum of distances between the points and their respective cluster centroid.
* The first step in k-means document clustering is to collect the input documents.
* The second step is to perform pre-processing tasks like removing stop words, tokenizing and stemming to the input documents.
* The third step is to perform TF.IDF which means converting text document into numerical form called vectorization.
* The fourth step is to implement K-mean clustering for the vector form of our input document where k is equal to the number of clusters you choose. Given K = 1, 2 and 3 as the input document contains data from three categories.
* The fifth step is predicting of some incoming documents to the cluster they belongs to.
* The sixth step is to measure the performance of k-mean i.e.; finding rand index.it ranges from 0 to 1 (perfect).

References:

<https://realpython.com/k-means-clustering-python/#understanding-the-k-means-algorithm>

<https://www.ijert.org/research/improved-clustering-of-documents-using-k-means-algorithm-IJERTV5IS070358.pdf>

<https://scikit-learn.org/stable/auto_examples/text/plot_document_clustering.html>

<https://research.ijcaonline.org/volume110/number11/pxc3900929.pdf>

<https://towardsdatascience.com/performance-metrics-in-machine-learning-part-3-clustering-d69550662dc6>

<https://www.tutorialspoint.com/scikit_learn/scikit_learn_clustering_performance_evaluation.htm>

[K-means Clustering Elbow Method & SSE Plot - Python - Data Analytics (vitalflux.com)](https://vitalflux.com/k-means-elbow-point-method-sse-inertia-plot-python/#:~:text=Elbow%20method%20requires%20drawing%20a,decreasing%20in%20a%20linear%20fashion).)

Appendix:

Task 1:

**Source Code:** crawled author details with all his publication details.

import requests

from bs4 import BeautifulSoup

def craw\_test(url,max\_pages):

page = 0

total\_publications = 0

total\_activities = 0

count\_authors = 0

Q = {}

while page <= max\_pages:

if (page >0):

new\_url = url+'?page='+str(page)

else:

new\_url = url

source\_code = requests.get(new\_url)

plain\_text = source\_code.text

soup = BeautifulSoup(plain\_text,"lxml")

for link in soup.findAll('a',{'class':'link person'}):

links = []

href = link.get('href')

title = link.string

#print('author profile link:',href)

#print('author name:',title)

links.append(href)

count\_authors += 1

links,no\_of\_publications,total\_publications = get\_single\_author\_publications(links,href+'/publications/',total\_publications)

Q[title] = links

print('total no of publications by',title,'are',no\_of\_publications)

page += 1

print('total no of authors in School of Economics, Finance and Accounting -Coventry University:' ,count\_authors)

print('total research publications by School of Economics, Finance and Accounting -Coventry University:',total\_publications)

print('Dictionary',Q)

def get\_single\_author\_publications(q,item\_url,total):

source\_code = requests.get(item\_url)

plain\_text = source\_code.text

soup = BeautifulSoup(plain\_text,"lxml")

count = 0

list\_of\_results = soup.findAll('div',{'class':'result-container'})

for each in list\_of\_results:

h3\_tag = each.find('h3')

link = h3\_tag.find('a').get('href')

title = h3\_tag.text.strip()

date = each.find('span',{'class':'date'})

#print('title of publication:',title)

#print('publications link:',link)

#print("date of publication:",date.text.strip())

q.append(link)

count +=1

total +=1

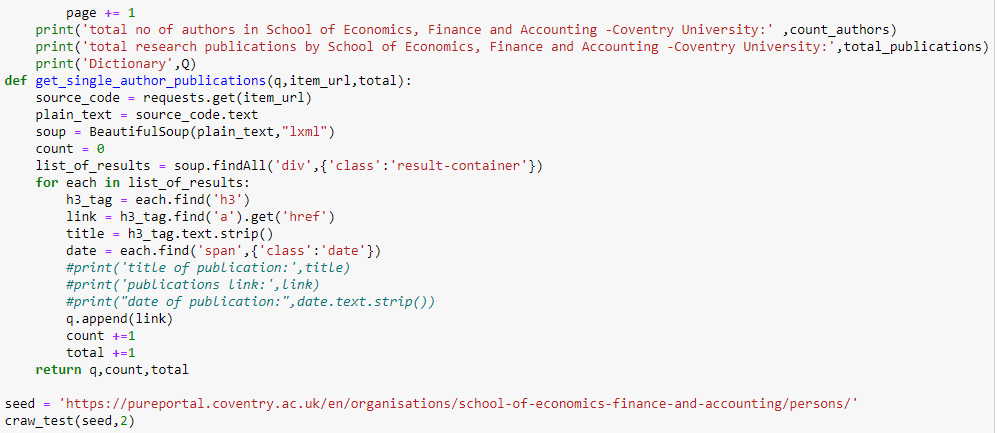
return q,count,total

seed = 'https://pureportal.coventry.ac.uk/en/organisations/school-of-economics-finance-and-accounting/persons/'

craw\_test(seed,2)

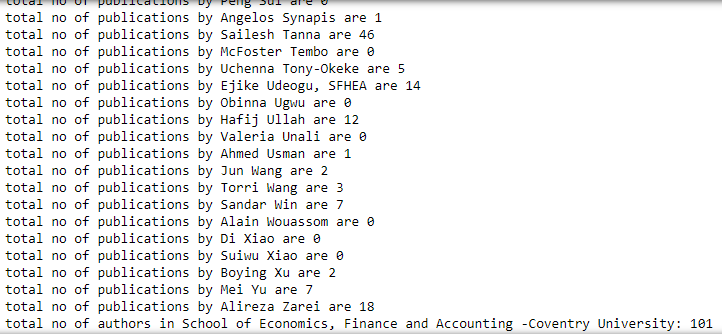
**Screenshots of code:**





**Output:**









Stored the data in a dictionary having key as name of the author/staff and values are a list Containing author’s link, and all his publications.

**Source Code:** Information collected about each publication

# Import libraries

from urllib.request import urljoin

from bs4 import BeautifulSoup

import requests

from urllib.request import urlparse

import re

import pandas as pd

# # Set for storing urls with same domain

# # Method for crawling a url at next level

def level\_crawler(input\_url,temp\_urls,count\_publications,df,j):

data = []

current\_url\_domain = urlparse(input\_url).netloc

# # Creates beautiful soup object to extract html tags

beautiful\_soup\_object = BeautifulSoup(requests.get(input\_url).content, "lxml")

# # Access all anchor tags from input

# # url page and divide them into internal

# # and external categories

for anchor in beautiful\_soup\_object.findAll("a"):

href = anchor.attrs.get("href")

title = anchor.string

if(href != "" or href != None):

href = urljoin(input\_url, href)

href\_parsed = urlparse(href)

href = href\_parsed.scheme

href += "://"

href += href\_parsed.netloc

href += href\_parsed.path

final\_parsed\_href = urlparse(href)

is\_valid = bool(final\_parsed\_href.scheme) and bool(

final\_parsed\_href.netloc)

if is\_valid:

if current\_url\_domain not in href and href not in links\_extern:

#print("Extern - {}".format(href))

links\_extern.add(href)

if current\_url\_domain in href and href not in links\_intern:

#if re.search("^https://pureportal.coventry.ac.uk/en/publications/.\*$", href):

#re.search("^The.\*Spain$", href)

if '/publications/' in href[36:50] and href[50:] != '':

#print("Intern - {}".format(href))

count\_publications += 1

links\_intern.add(href)

temp\_urls.append(href)

data.append(href)

#print('publications title:',title)

df.loc[j, ['publication\_link']] = href

df.loc[j, ['publication\_title']] = title

df,j = get\_author\_details(href,df,j)

j += 1

return temp\_urls,count\_publications,df,j,data

def get\_author\_details(link,df,j):

source\_code = requests.get(link)

plain\_text = source\_code.text

soup = BeautifulSoup(plain\_text,"lxml")

for link in soup.findAll('a',{'class':'link person'}):

href = link.get('href')

name = link.string

source\_code1 = requests.get(href)

plain\_text1 = source\_code1.text

soup1 = BeautifulSoup(plain\_text1,"lxml")

if(soup1.find('a',{'class':'link primary'}) != None):

sefa = soup1.find('a',{'class':'link primary'}).string

elif(soup1.find('a',{'class':'link school'}) != None):

sefa = soup1.find('a',{'class':'link school'}).string

if (sefa == 'School of Economics, Finance and Accounting'):

df.loc[j, ['author\_link']] = href

df.loc[j, ['author\_name']] = name

publi = soup.find('tr',{'class':'status'})

date = publi.text.strip('date')

#print('date:',date)

#print(abstract,'abs..........')

if (soup.find('div',{'class':'textblock'})):

abstract = soup.find('div',{'class':'textblock'}).string

else:

abstract = 'Null'

#print(abstract,'abs..........')

if type(abstract) == 'NoneType':

abstract = ' '

df.loc[j, ['publication\_date']] = date

df.loc[j, ['abstract']] = abstract

#df.loc[j, ['content']] = abstract + name

return df,j

links\_intern = set()

url = 'https://pureportal.coventry.ac.uk/en/organisations/school-of-economics-finance-and-accounting/publications/'

depth = 1

p = 0

max\_pages = 12

count\_publications = 0

# # Set for storing urls with different domain

df = pd.DataFrame(columns = ["publication\_link","publication\_title","publication\_date","author\_link","author\_name","abstract","content"])

j = 0

links\_extern = set()

list\_url = []

if(depth == 0):

print("Intern - {}".format(input\_url))

elif(depth == 1):

while p >= 0:

if p == 0:

list = []

print(url,'url page0')

list\_url,count\_publications,df,j,data = level\_crawler(url,list,count\_publications,df,j)

p += 1

else:

new\_url = url+'?page='+str(p)

print(new\_url,'url page')

list\_url,count\_publications,df,j,data = level\_crawler(new\_url,list\_url,count\_publications,df,j)

p += 1

if (data == []):

p = -1

else:

# # We have used a BFS approach

# # considering the structure as

# # a tree. It uses a queue based

# # approach to traverse

# # links upto a particular depth.

queue = []

queue.append(input\_url)

for j in range(depth):

for count in range(len(queue)):

url = queue.pop(0)

urls = level\_crawler(url)

for i in urls:

queue.append(i)

print('end')

print('total no of publications',count\_publications)

#print(list\_url)

df['content']=df['abstract']+df['author\_name'] +df[‘publication\_Title']

df

import csv

df.to\_csv(path\_or\_buf ="crawlerdata.csv",sep=',')

**Source code for pre-processing tasks:**

Reading a csv file and stored in a data frame which contains all the publications data which is being stored at the end of crawler program.

df = pd.read\_csv("crawlerdata.csv")

print(df)

content = df['content'].astype(str)

print(content)

import nltk

nltk.download("stopwords")

from nltk.corpus import stopwords

sw = stopwords.words('english')

nltk.download("punkt")

from nltk.tokenize import word\_tokenize

from nltk.stem import PorterStemmer

ps = PorterStemmer()

filtered\_docs = []

for doc in content:

tokens = word\_tokenize(doc)

tmp = ""

for w in tokens:

if w not in sw:

tmp += ps.stem(w) + " "

filtered\_docs.append(tmp)

print(filtered\_docs)

**Inverted index Source code:**

inverted\_index = {}

**for** i, doc **in** enumerate(df['fil\_content']):

**for** term **in** doc.split(" "):

**if** term **in** inverted\_index:

inverted\_index[term].add(i)

**else**: inverted\_index[term] = {i}

inverted\_index

**Query Processor Source code:**

query = input('enter input query:')

**import** **nltk**

nltk.download("stopwords")

**from** **nltk.corpus** **import** stopwords

sw = stopwords.words('english')

nltk.download("punkt")

**from** **nltk.tokenize** **import** word\_tokenize

**from** **nltk.stem** **import** PorterStemmer

ps = PorterStemmer()

filtered\_docs = []

tokens = word\_tokenize(query)

tmp = ""

**for** w **in** tokens:

**if** w **not** **in** sw:

tmp += ps.stem(w) + " "

filtered\_docs.append(tmp)

#print(filtered\_docs)

x = filtered\_docs[**0**].split(" ")

q = []

**for** i **in** range(len(x)):

**if** len(x[i]) > **0**:

q.append(x[i])

#print(q)

p = []

**for** i **in** range(len(q)):

**if** (q[i] **in** inverted\_index):

a = inverted\_index[q[i]]

l = sorted(a)

p.append(l)

print('l = ',q[i],l)

**else**:

print(q[i],'not found')

**for** i **in** p:

print(i)

**for** i **in** range(len(p)):

print(p[i])

**for** j **in** range(len(p[i])):

#print(p[i][j])

print(df.loc[p[i][j]])

**Task 2:**

**Source Code for crawling input docs related to sports, health and politics from BBC news website:**

import requests

from bs4 import BeautifulSoup

def craw\_test(l):

count = 0

url = 'https://www.bbc.co.uk/sport'

source\_code = requests.get(url)

plain\_text = source\_code.text

soup = BeautifulSoup(plain\_text,"lxml")

for link in soup.findAll('p',{'class':"ssrcss-6arcww-PromoHeadline e1f5wbog4"}):

title = link.string

if(title != None and len(title) > 40 and count < 20):

#print('news headlines:',title)

l.append(title)

count += 1

url = 'https://www.bbc.co.uk/news/uk-politics-62239950'

source\_code = requests.get(url)

plain\_text = source\_code.text

soup = BeautifulSoup(plain\_text,"lxml")

for link in soup.findAll('p',{'class':"ssrcss-1q0x1qg-Paragraph eq5iqo00"}):

title = link.string

if(title != None and len(title) > 40 and count < 60):

#print('news headlines:',title)

l.append(title)

count += 1

count = 0

url = 'https://www.bbc.co.uk/news/politics'

source\_code = requests.get(url)

plain\_text = source\_code.text

soup = BeautifulSoup(plain\_text,"lxml")

for link in soup.findAll('p',{'class':"gs-c-promo-summary gel-long-primer gs-u-mt nw-c-promo-summary gs-u-display-none gs-u-display-block@m"}):

title = link.string

#print('politics headlines:',title)

l.append(title)

for link in soup.findAll('a',{'class':"gs-c-promo-heading gs-o-faux-block-link\_\_overlay-link gel-pica-bold nw-o-link-split\_\_anchor"}):

title = link.string

count = 0

if(title != None and len(title) > 40 and count < 40):

#print('politics headlines:',title)

l.append(title)

count += 1

count = 0

url = 'https://www.bbc.co.uk/news/health-62250899'

source\_code = requests.get(url)

plain\_text = source\_code.text

soup = BeautifulSoup(plain\_text,"lxml")

for link in soup.findAll('p',{'class':"ssrcss-1q0x1qg-Paragraph eq5iqo00"}):

title = link.string

count = 0

if(title != None and len(title) > 40 and count < 60):

#print('health news:',title)

l.append(title)

count += 1

return l

l = []

docs = craw\_test(l)

print(len(docs))

print(docs)

**Source Code:**

import nltk

nltk.download("stopwords")

from nltk.corpus import stopwords

sw = stopwords.words('english')

print(sw)

**Source Code for pre-processing task:**

nltk.download("punkt")

from nltk.tokenize import word\_tokenize

from nltk.stem import PorterStemmer

ps = PorterStemmer()

filtered\_docs = []

for doc in docs:

tokens = word\_tokenize(doc)

tmp = ""

for w in tokens:

if w not in sw:

tmp += ps.stem(w) + " "

filtered\_docs.append(tmp)

print(filtered\_docs)

**Tfidf vectorization Code:**

from sklearn.feature\_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer()

X = vectorizer.fit\_transform(filtered\_docs)

print(X.todense())

print(X)

**Elbow graph Code:**

**import** **matplotlib.pyplot** **as** **plt**

**from** **sklearn.cluster** **import** KMeans

sse = {}

**for** k **in** range(**1**,**4**):

kmeans = KMeans(n\_clusters=k, max\_iter=**5**).fit(X)

#data["clusters"] = kmeans.labels\_

#print(data["clusters"])

sse[k] = kmeans.inertia\_ # Inertia: Sum of distances of samples to their closest cluster center

plt.figure()

plt.plot(list(sse.keys()), list(sse.values()))

plt.xlabel("Number of cluster")

plt.ylabel("SSE")

plt.show()

**k-means Code:**

from sklearn.cluster import KMeans

K = 3

model = KMeans(n\_clusters=K)#, init='k-means++', max\_iter=100, n\_init=1)

model.fit(X)

print("cluster no. of input documents, in the order they received:")

print(model.labels\_)

**k-means test Code:**

test\_doc = ["The Covid pandemic had a devastating impact on ethnic minority communities",

"Boris Johnson was expecting a teary-eyed send off from all MPs",

"care home residents died of Covid in England and Wales",

"England on top in first T20 v South Africa",

"contest to elect his replacement as Tory leader and prime minister is due to conclude",

"Vingegaard close to Tour victory after stage 18 win",

"Birmingham attempts to leave 'carbon neutral legacy",

"Coe says global warming could force move of events",

"16 medal event first day - watch ",

"France wins world cup in football",

"Sunak says yes to return of grammar schools"

]

filtered\_test\_docs = []

for doc in test\_doc:

tokens = word\_tokenize(doc)

tmp = ""

for w in tokens:

if w not in sw:

tmp += ps.stem(w) + " "

filtered\_test\_docs.append(tmp)

print(filtered\_test\_docs)

prediction = []

labels\_pred = []

for i in range(len(filtered\_test\_docs)):

Y = vectorizer.transform([filtered\_test\_docs[i]])

prediction=(model.predict(Y))

print(prediction)

labels\_pred.append(prediction[0])

rand index performance measure Code:

#rand index - performance measure of k-mean cluster

**from** **sklearn** **import** metrics

labels\_true = [**0**,**1**,**0**,**2**,**1**,**2**,**0**,**0**,**2**,**2**,**2**]

metrics.rand\_score(labels\_true, labels\_pred)

input docs list for clustering:

['Commonwealth Games: 16 medal events on first day - watch', 'Rossouw hits 96 as South Africa level T20 series', "'Now or never' - hurdler Allen chases NFL dream", 'Watch: Pick from 11 Commonwealth Games streams', 'A decade of development, six landmark moments', 'Arsenal & Northern Ireland legend Neill dies at 80', "Hamilton & Alonso lead tributes to 'legend' Vettel", "Jamaica's ex-GB athlete Tracey ruled out of Games", "Man City abandon Cucurella plans - Friday's gossip", "Stenson does not feel he 'gave up' Ryder Cup role", 'Anger over pro-Putin chants at Turkey football match', 'Motherwell dumped out of Europe by inspired Sligo', 'Crusaders bow out of Europe after Basel draw', 'Raging bulls & a Nobel laureate - Commonwealth Games open in style', 'Commonwealth Games day-by-day guide & schedule', "'My brothers hung me from doors, but it made me'", 'BBC TV, radio & digital coverage times for Commonwealth Games', "'I want to leave legacy' - Fachie's quest for glory", 'England aim for gold as cricket returns to Games', "Flagbearer Gilmour's letter to 16-year-old self", "Watch the key moments from Boris Johnson's final PMQs", "If Boris Johnson was expecting a teary-eyed send off from all MPs, he didn't get one at his final Prime Minister's Questions.", "The outgoing prime minister's last weekly grilling at the dispatch box was not unlike his first in September 2019 - acrimonious, raucous, and littered with his colourful quips.", 'After almost an hour of political theatre, Mr Johnson brought the curtain down in characteristic fashion, with a tongue-in-cheek farewell, borrowed from the script of a sci-fi blockbuster.', '"Hasta la vista, baby," Mr Johnson told MPs, reciting the line from Arnold Schwarzenegger\'s cyborg character in the 1991 film Terminator 2: Judgment Day.', "Cue uproarious laughter, cheers and a standing ovation from the Conservative benches - although former PM Theresa May didn't appear to join in the applause.", 'Given Schwarzenegger\'s other famous catchphrase in the film - "I\'ll be back" - Mr Johnson\'s sign-off and what it means may hang in the air in Westminster.', 'For now, at least, Mr Johnson will leave office by September, when the contest to elect his replacement as Tory leader and prime minister is due to conclude.', 'In a pithy closing statement Mr Johnson had a few words of advice for his successor, saying that they should stay close to the Americans, stick up for the Ukrainians and cut taxes whenever possible.', '"Focus on the road ahead, but always remember to check rear-view mirror. And remember above all, it\'s not Twitter that counts - it\'s the people who sent us here."', "Held in the pressure cooker of the House of Commons, Prime Minister's Questions is designed to be an adversarial occasion of high political drama.", 'The first was in 1961, and since then, prime ministers have admitted the event holds some terror.', 'Years after leaving office, former Prime Minister Tony Blair likened PMQs to being marched to his own "execution", while one of his predecessors, Harold Macmillan, confided it made him feel "physically sick".', "It's easy to see why as Mr Johnson faced a hostile audience of MPs, some of whom had been involved in ousting him as Tory leader weeks ago.", 'Before proceedings began, Commons Speaker Lindsay Hoyle urged MPs to moderate their language and conduct themselves in a respectful manner.', "The plea appeared to fall on deaf ears though as Labour MPs - led by leader Sir Keir Starmer - launched a barrage of attacks on Mr Johnson's integrity and record in as PM. ", '"Inflation is up again this morning and millions are struggling with a cost of living crisis, and he\'s decided to come down from his gold wallpapered bunker for one last time to tell us that everything\'s fine," Sir Keir said.', 'Mr Johnson paid little heed to Sir Lindsay\'s plea either, branding Sir Keir a "great pointless human bollard".', 'It was the kind of unconventional one-liner Mr Johnson has reeled off time and again during his 93 PMQs duels, most of them against Sir Keir.', '"Captain Hindsight" was one of Mr Johnson\'s favourite nicknames for Sir Keir, often used in the context of his calls for lockdown restrictions during the pandemic.', 'Many of their fiercest clashes at PMQs were over Covid-19 rules, and the breach of them by Mr Johnson and others in Downing Street.', 'A more recent addition to Mr Johnson\'s jibes at Sir Keir has been "Captain Crasheroonie Snoozefest", and he once described former Labour leader Jeremy Corbyn as a "chlorinated chicken" in an exchange about post-Brexit trade with the US.', 'When asked why the Tory leadership candidates had pulled out of a televised debate this week, Mr Johnson once again conjured an vivid image with his rhetoric.', 'The candidates would "wipe the floor" with Sir Keir, Mr Johnson said, comparing his Tory colleagues to "household detergent".', 'Perhaps not quite what eliminated Tory leadership candidate Tom Tugendhat had in mind when he offered a "clean start".', 'BBC political correspondent Ione Wells sat in the press gallery of the Commons watching the spectacle unfold.', 'The atmosphere was "pretty jovial overall", she said. She said even Mr Johnson\'s critics couldn\'t help but chuckle at his gags, with Labour and Liberal Democrat MPs laughing at his notorious rhetorical flourishes, which have arguably got him in trouble over the years.', "Mr Johnson's valedictory speech came after veteran Conservative backbencher Sir Edward Leigh praised his record.", 'Labour MPs could be heard shouting "no" as Sir Edward said: "On behalf of the House may I thank the prime minister for his three years\' record of service."', "Stepping up to the dispatch box for the last time, Mr Johnson thanked his staff and MPs before giving a nod to Schwarzenegger's Terminator character.", 'As the heat of PMQs cooled, Mr Johnson left the chamber, receiving pats on the back and handshakes as he went.', 'Some tears were apparently shed after all by Conservative minister Andrea Jenkyns.', "These were the end credits of Mr Johnson's swansong PMQs, or, to evoke his Terminator reference, the final parliamentary Judgement Day of his premiership.", 'In his first PMQs on 4 September 2019, Mr Johnson called Mr Corbyn a "great big girl\'s blouse" when the then-Labour leader challenged the PM about parliamentary scrutiny.', '"Call an election, you great big girl\'s blouse"', 'On 12 January 2022, Mr Johnson offered "heartfelt apologies" for attending drinks in Downing Street\'s garden on 20 May 2020, when lockdown restrictions were in force.', 'The prime minister was accused of "body shaming" the SNP\'s Westminster leader Ian Blackford during a PMQs clash on 26 January 2022.', 'Ian Blackford: "The impending National Insurance tax hike hangs like a guillotine while they eat cake."', 'On 25 May 2022, Mr Johnson repeated his apology for parties held in Downing Street during lockdown, after senior civil servant Sue Gray published a report into breaches of Covid-19 rules.', 'Prime Minister Boris Johnson: I am humbled and I have learned a lesson', 'But his team later clarifies Mr Sunak was only backing the expansion of existing grammar schools.', 'Liz Truss continues to emphasise her loyalty to the outgoing prime minister, while Rishi Sunak stands accused of stabbing him in the back', 'The Labour leader insists he did not sack his transport spokesman for joining a union picket line.', 'The former ambassador to the US became famous for his colourful turn of phrase - and colourful socks.', 'The culture secretary also accuses the ex-chancellor of leading a "ruthless coup" against Boris Johnson.', 'Rampant inflation is leading to demands for higher pay offers and promises of coordinated industrial action.', 'But his team later clarifies Mr Sunak was only backing the expansion of existing grammar schools.', 'Liz Truss continues to emphasise her loyalty to the outgoing prime minister, while Rishi Sunak stands accused of stabbing him in the back', 'The Labour leader insists he did not sack his transport spokesman for joining a union picket line.', 'Sunak says yes to return of grammar schools', 'Chris Mason: Truss courts Johnson loyalists as Sunak faces jibes', 'MP fired for making up policy on the hoof - Starmer', 'Ex-ambassador Sir Christopher Meyer dies at 78', "Dorries defends Claire's Accessories Sunak attack", 'Sunak says yes to return of grammar schools', 'Chris Mason: Truss courts Johnson loyalists as Sunak faces jibes', 'MP fired for making up policy on the hoof - Starmer', 'Ex-ambassador Sir Christopher Meyer dies at 78', "Dorries defends Claire's Accessories Sunak attack", 'Bryant makes charity donation after claims disproved', "Have wages been 'stuck for a decade'? And other claims", 'Cuts leave military at risk as threats rise - MPs', 'Government rules out bank holiday for Euro win', 'Truss vows to criminalise street harassment', 'How UK Greens are learning from overseas allies', 'Five takeaways from a heated Truss-Sunak clash', 'Who are the Tories that will choose the next PM?', 'What taxes would the two Tory leadership rivals cut?', "Lessons will be learned about the UK's handling of Covid, before another pandemic strikes, Baroness Hallett, chairing the public inquiry, has said.", 'Opening the inquiry, she promised to be "fair and robust".', 'The former High Court judge said she would conduct the inquiry as quickly as possible, without giving a timeframe for its completion.', 'Those who had suffered the most deserved to know if more could have been done, she said.', 'Lives had been lost, education harmed, businesses folded and mental and physical health had suffered.', '"Every person has had their life changed to some extent," Lady Hallett said.', '"Those who have suffered the most will want to know if any more could have been done to reduce their suffering."', 'The inquiry can compel witnesses to give evidence and release documents, but cannot prosecute or fine anyone.', 'It was a substantial task that would take time and have a significant cost, Lady Hallett said.', 'But she added: "I am determined to undertake the inquiry as speedily as possible so lessons can be learned before another pandemic strikes."', 'Public hearings will begin in the spring.', 'Before then, Lady Hallettt said, the key topics for the inquiry would include:', 'The inquiry will begin taking evidence from experts in September.', 'There will be many involved in the care sector who will have a great deal to say. They will remember the early warnings from other countries about the vulnerability of care homes. ', 'Care providers will recount their struggle to get protective equipment; what they saw as the slowness of government guidance; the rapid discharge of hospital patients into care homes, some taking with them the virus and above all, their sense of being forgotten. ', 'Families who were unable to see or in too many cases, say goodbye to their loved ones, are likely to have their say later in the inquiry. ', 'But in weighing up our preparedness for the pandemic, there is one number that perhaps tells the most powerful story. In the first wave between March and June 2020, nearly 20,000 care home residents died of Covid in England and Wales. At that time, it represented more than a third of the total number of deaths.', 'Jo Goodman, co-founder of Covid-19 Bereaved Families for Justice campaign, said: "Today was an emotional day for those of us who have lost loved ones and it meant a lot to hear Baroness Hallet recognise the devastating nature of bereavement and the pain we\'ve been through. ', '"Ultimately, all bereaved families want the same thing, which is to make sure that lessons are learnt from our devastating losses to protect others in the future."', 'Charles Persinger, who is part of the campaign and lost his mother and wife to Covid, added: "We\'ve waited a long time to get to this point - we would have liked the inquiry to start sooner. But what is important now is to really get to the bottom of the mistakes that were made."', "Several reports have already put the UK government's handling of the pandemic under the spotlight.", 'The Covid pandemic had a devastating impact on ethnic minority communities in the UK. ', 'In the first wave of the pandemic, black people were almost four times as likely to die of Covid than white people, while Asians were twice as likely to die. About 95% of doctors who died of Covid were ethnic minorities. ', 'Saleyha Ahsan, a doctor and documentary maker, lost her dad to Covid-19. She tells me it\'s good inequalities are being considered, but she\'s "keen to see how [Baroness Hallett] does it in real terms".', '"I\'m not convinced it\'s going to cover it all, so I am wary." ', 'Dr Ahsan is involved with the Covid-19 Bereaved Families for Justice, which led the campaign for an inquiry to happen.', '"The main thing is is that we have now crossed the start line," she says. "We\'ve been calling for this for a whileâ\x80¦ and today is a definite firing of the starting gun." ', 'From intricate mosaics to amazing hieroglyphics - what else hides under the water?', 'Matthew Syed is calling for a nuclear awakening...']