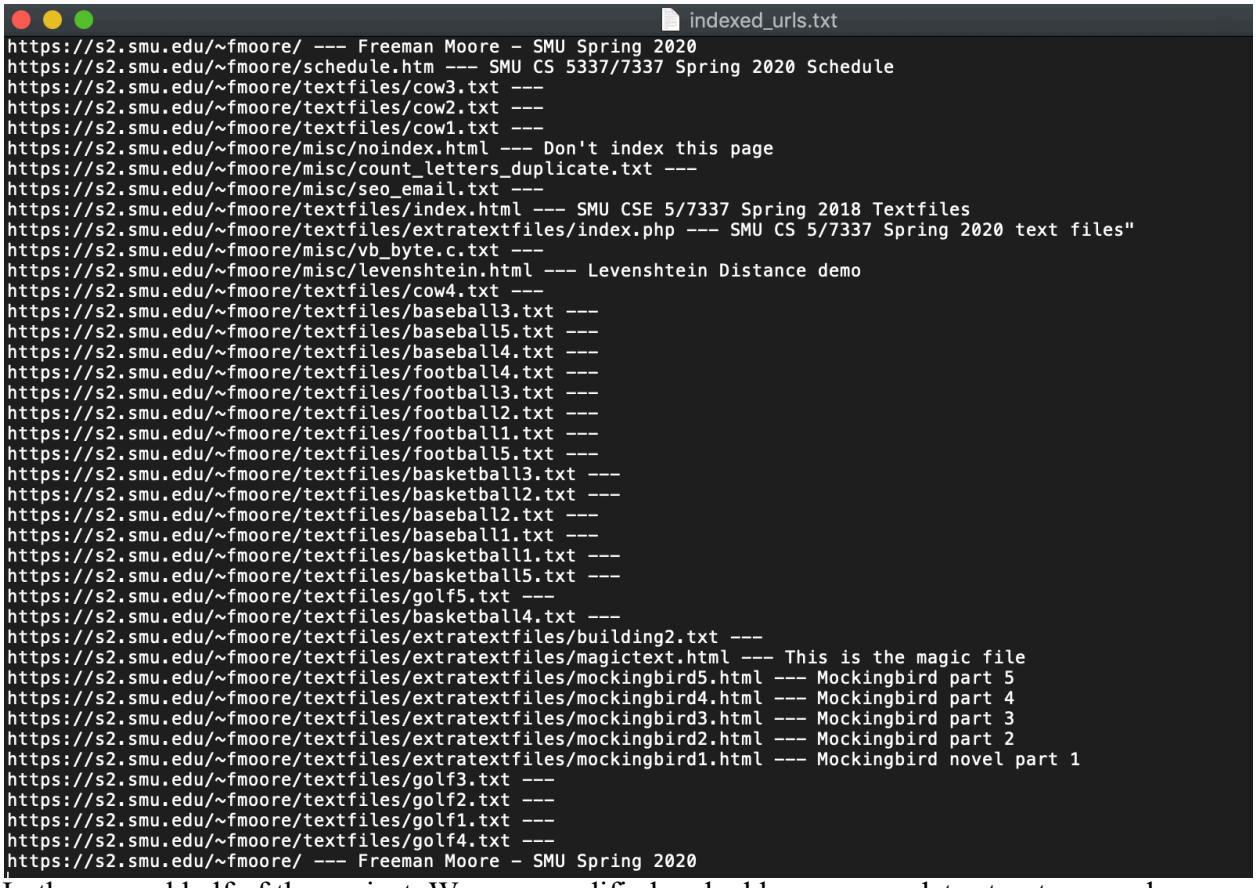
CS 5337/7337: Information Retrieval and Web Search **Name: Shradha Uttarwar**
Spring 2020, Project 2: Query engine implementation (100 points) **SMU_Id: 47800822**
Due: Tuesday May 5, 6:00 pm
Demonstrations are ***encouraged***, but not required.

1. **Use the web crawler you built in Project 1 that crawled a limited space, looking for text, html files and php files. You may need to modify how you saved the information from the documents that you traversed to support the query engine. You may assume a maximum of 60 documents will need to be indexed (the number of files fetched will be higher than this). Describe in detail what you changed to support the second half of the project. [15/15 points]**

   Answer:

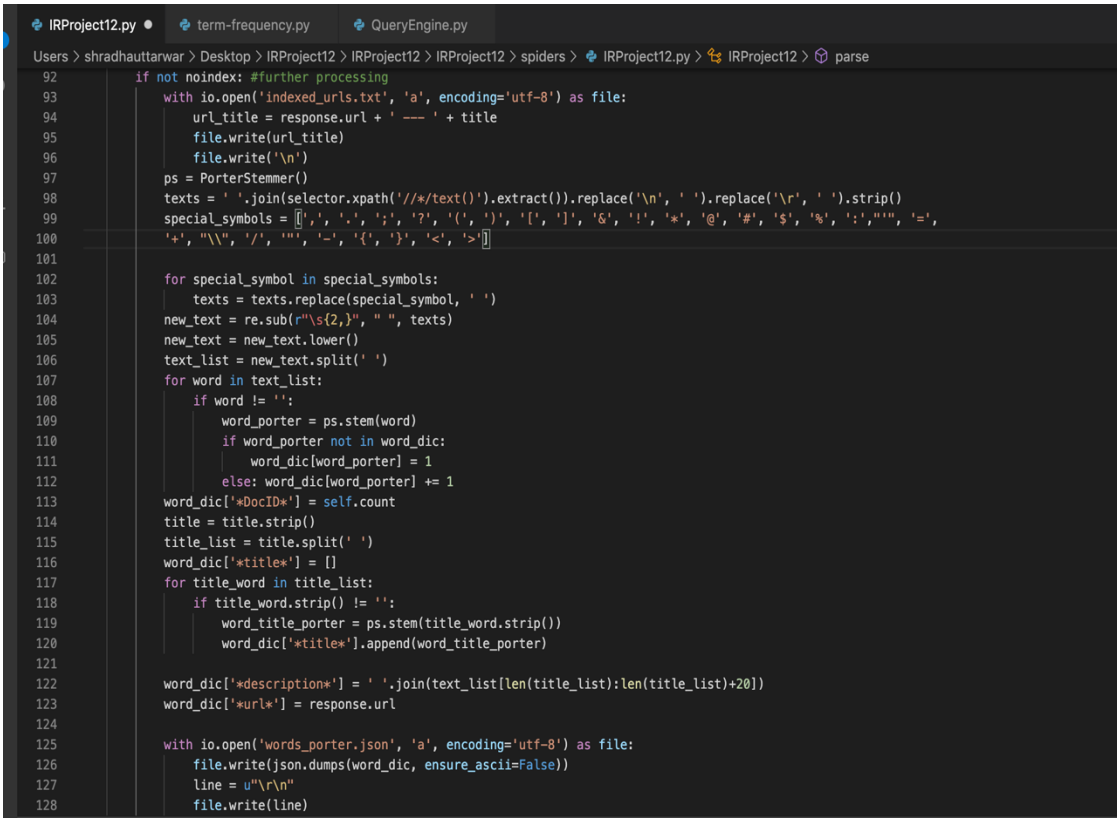
   Using the web crawler in Project 1, I indexed 40 pages this time. The URLs of these pages are following:



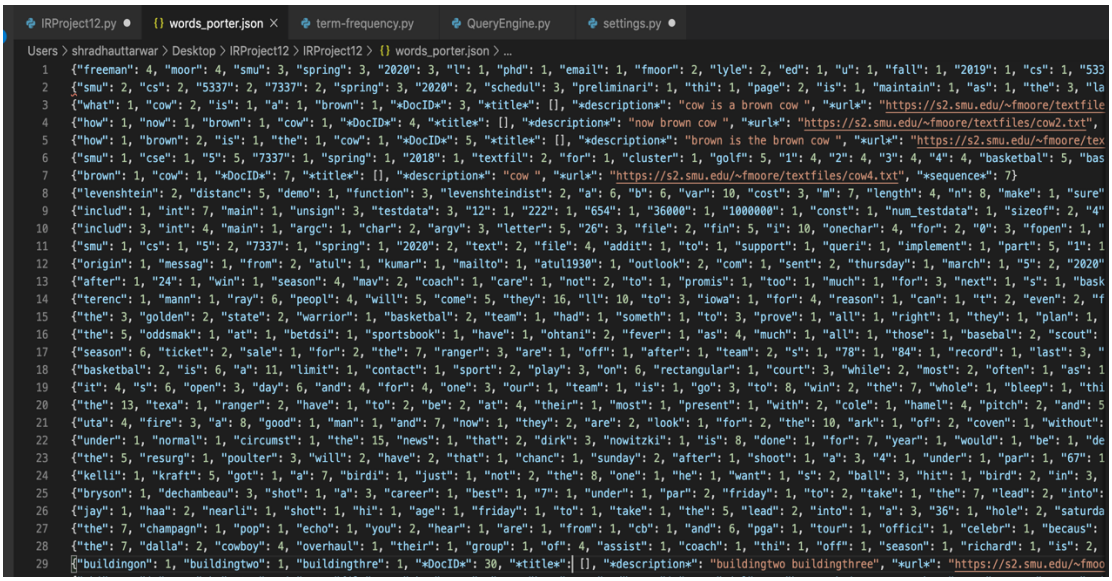
   In the second half of the project, We, can modified and add some new data structures and try to crawl indexed documents.

1.1 Web Crawling phase:

Web crawling part for each indexed page, used xpath to extract its title, url, and the 1st of the 20 words of this page and all texts of this page. Therefore, as we can see for every word, we can put it into a dictionary, the key of the dictionary is the word and value of the dictionary if there is a frequency of that word in this document. Then we can add a key called "*title*" and the value of this key "*title*" is the title of this page. Whereas we have also added a key called "*url*" and the value of this key "*url*" is the url of this page. However, we have added a key called "*description*" and the value of this key "*description*" of this page. Alongside assigned a unique document ID for every page, so also added an additional key called "*sequence*" and the value of the key "*sequence*" is the document ID. Then write dictionary in to a intermediate json file "word_porter. json". Each dictionary in this file represents an indexed page.

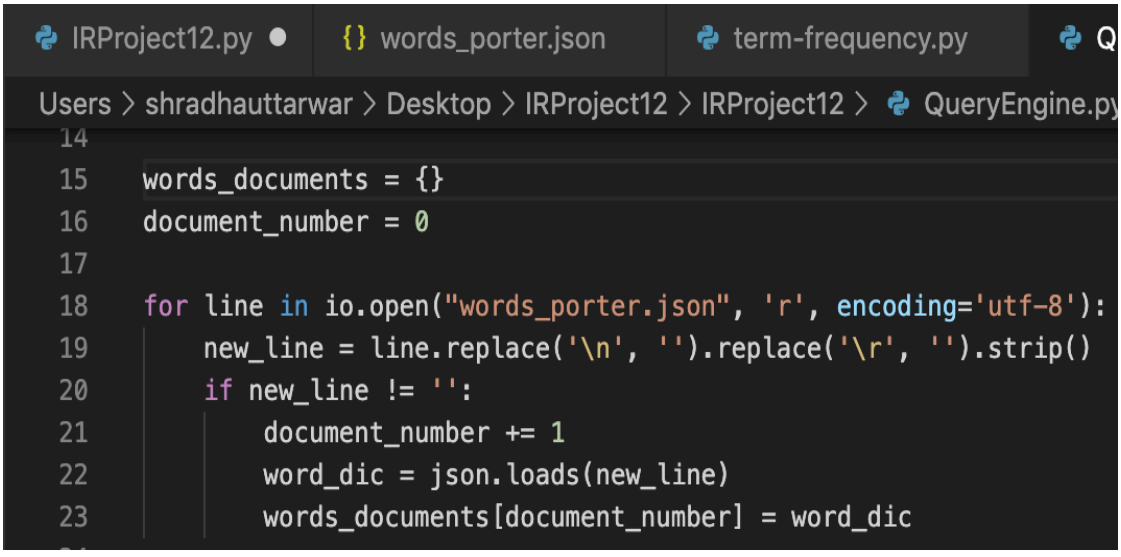


Whereas this file is large so it cannot be shown entirely using screenshot, therefore only put small part of the json file. Below is the result from the "word_porter. Json" file.



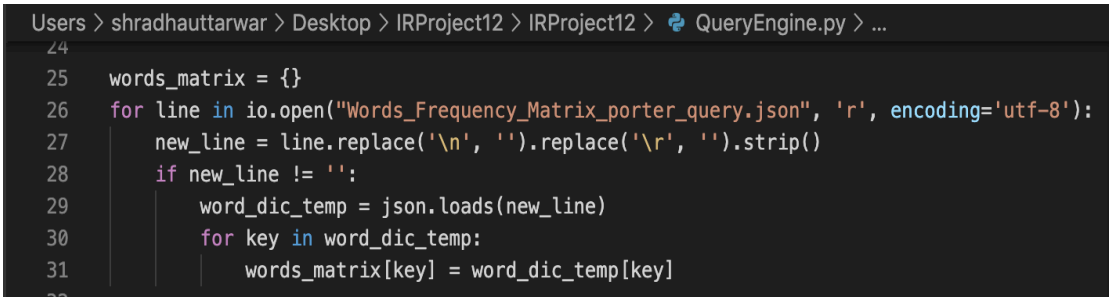
1.2 Created a dictionary for “words_porter. json”:

“Word_porter. json” automatically read file when the query engine is running. So, create a dictionary called words_document= {} to save the file. Therefore, the key if this dictionary is the unique document ID which is the value of “*Sequence*”. The value of this dictionary is the dictionary of each document we saved in file “words_porter. json”. This document used to find the title of url, and the 20 words of the document based on its unique document Id.



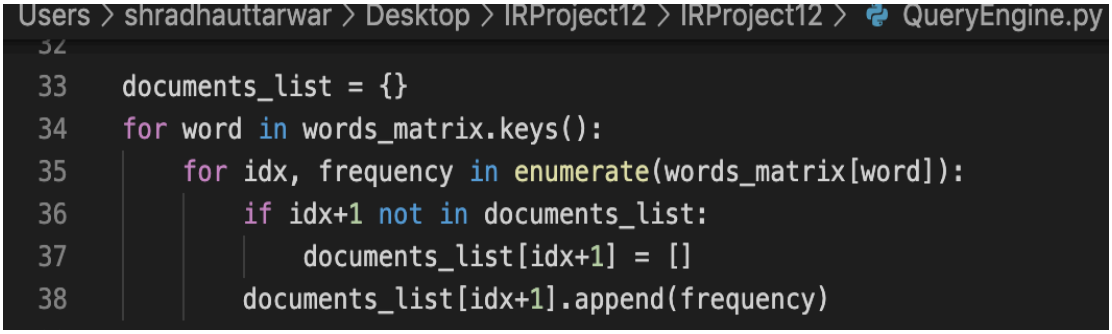
1.3 Created a dictionary for “words_frequency_matrix”:

The word frequency matrix will run the query engine with the automatically for that we have created a dictionary called words_ matrix= {} to save that. The key of this dictionary in each unique word. The value of this dictionary is a list that contains word frequency in each document, so the length of this list is the number of documents.



1.4 Created a dictionary to save term frequency of each term in each document:

Purpose of creating this dictionary to make calculating cosine similarity conveniently. The key of this dictionary is the unique document ID, and the value of this dictionary is term frequency of each term is this document. Therefore, if we see the example of 40 documents and 1000 words, then this dictionary has 40 keys. The value of each key is list that contains term frequency of these 1000 words in this document, so the length of this is 1000.



1.5 Created a dictionary to save the document frequency of each word:

The document frequency dictionary also makes calculating cosine similarly conveniently. During creating this dictionary, we assign a unique word ID to each word. So, the key of this dictionary is the unique word ID and the value of this dictionary is the document frequency of this word.

```
IRProject12.py ●   term-frequency.py   QueryEngine.py ×
Users > shradhauttarwar > Desktop > IRProject12 > IRProject12 > QueryEngine.py > ...
39
40    idf_dict = {}
41    word_number = 0
42    for word in words_matrix.keys():
43        word_number += 1
44        idf_dict[word_number] = document_number - words_matrix[word].count(0) + 1
```

Modification done is the next question.

2. **The user will be able to enter multiple queries, consisting of one or more query words separated by space. The single word query "stop" will cause your program to stop. [15/15 points]**

   a) **Describe what happens if a user enters a word that is not in the dictionary.**

   b) **You may assume the user only enters "words" per your definition.**

   c) **Do not recrawl the website for each query, use your collected data.**

   d) **CS7337: You will need to implement thesaurus expansion on each term using the list provided at the end.**

Answer:

The query engine will run inside a while loop. Therefore, in this while loop, the query engine is waiting for user to enter a query. When the user enters, "stop", the query engine stops running.

```
Users > shradhauttarwar > Desktop > IRProject12 > IRProject12 > QueryEngine.p
55    while True:
56        if not is_expanded:
57            query = input('\nInput query you wish: ')
58
59            if query == 'stop':
60                print('\nClosing the query engine...')
61                break
62            query_list = query.split(' ')
```

```
Input query you wish: stop

Closing the query engine...
(IRProject12) Shradhas-Air:IRProject12 shradhauttarwar$
```

However, we can see the query engine is waiting for user to enter query as long as user doesn't enter "stop". When user enters "stop", the query engine stops running.

Document should not be recrawl for each query. When this query engine starts, we read file ('words_porter. json' & 'Words_Frequency_Matrix_porter_query. json') once user enter query. Then we can find every query during engine running, we will use the file we read to find result for every query until query engines shuts down.

When query word is not in the dictionary some words in a query are in the dictionary and other words in a query are not in the dictionary, we only use the words that is in the dictionary to calculate the cosine similarity. If all of words in a query are not in the dictionary. Then first we print "We did not find article about this query. Now let us try query expansion" and we will expand the query words using thesaurus expansion and use the new query. If the new query is not in the dictionary either, then we print "Sorry, we cannot find any result about this query."

IRProject12.py ● | {} words_porter.json | term-frequency.py | QueryEngine.py ● | settings.py ●

Users > shradhauttarwar > Desktop > IRProject12 > IRProject12 > QueryEngine.py > ...

```
143
144     else:
145         if is_expanded:
146             is_expanded = False
147             print('Sorry, we cannot find any result about this query.')
148         else:
149             print('---------------------------')
150             print('No article related this query. Now let us try query expansion:')
151             is_expanded = True
152             last_query_list = [query_word for query_word in query_list]
153
154
```

Let's see some examples:

```
Input query you wish: smu sjhdjdahfuia
--------------------
Score:  0.22278402875996756
URL:  https://s2.smu.edu/~fmoore/
Title:  freeman moor - smu spring 2020
Description:  2020 freeman l moore phd email fmoore lyle smu ed u fall 2019 cs 5330 7330 keep looking at the
--------------------
Score:  0.17362363285124685
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/index.php
Title:  smu CS 5/7337 spring 2020 text files"
Description:  files additional text files to support query implementation part 1 part 2 part 3 part 4 part 5 hocuspocus 3
--------------------
Score:  0.17337378097914474
URL:  https://s2.smu.edu/~fmoore/textfiles/index.html
Title:  smu cse 5/7337 spring 2018 textfil
Description:  textfiles textfiles for clustering golf 1 golf 2 golf 3 golf 4 golf 5 basketball 1 basketball 2 basketball 3
--------------------
Score:  0.15615318273648168
URL:  https://s2.smu.edu/~fmoore/schedule.htm
Title:  smu CS 5337/7337 spring 2020 schedul
Description:  schedule smu cs 5337 7337 spring 2020 preliminary schedule this page is maintained as the latest schedule of content and
```

In the above example for query "*smu sjhdjdahfuia*", word "*smu"* is in the dictionary but word "*sjhdjdahfuia*" is not in the dictionary. So, we will only use word "*smu*" to calculate the similar and returns the result.

```
Input query you wish: nowork
---------------------------
No article related this query. Now let us try query expansion:
Sorry, we cannot find any result about this query.

Input query you wish: no work
---------------------
Score:  0.040508971607146824
URL:  https://s2.smu.edu/~fmoore/textfiles/baseball4.txt
Title:  No title for this article.
Description:  oddsmakers at betdsi sportsbook have ohtani fever as much as all those baseball scouts and writers tripping over themselves as
---------------------
Score:  0.038654316820208194
URL:  https://s2.smu.edu/~fmoore/misc/seo_email.txt
Title:  No title for this article.
Description:  original message from atul kumar mailto atul1930 outlook com sent thursday march 5 2020 12 58 am subject first page
---------------------
Score:  0.03464360745416091
URL:  https://s2.smu.edu/~fmoore/schedule.htm
Title:  smu CS 5337/7337 spring 2020 schedul
Description:  schedule smu cs 5337 7337 spring 2020 preliminary schedule this page is maintained as the latest schedule of content and
---------------------
Score:  0.033494638512305414
URL:  https://s2.smu.edu/~fmoore/textfiles/baseball3.txt
Title:  No title for this article.
Description:  ticket sales for the rangers are off after the team s 78 84 record last season according to a source
---------------------
Score:  0.025695692829510352
URL:  https://s2.smu.edu/~fmoore/textfiles/basketball2.txt
Title:  No title for this article.
Description:  fired a good man and now they are looking for the ark of the covenant without indiana jones – they
---------------------------
No article related this query. Now you can reenter Query:
---------------------
Score:  0.040508971607146824
URL:  https://s2.smu.edu/~fmoore/textfiles/baseball4.txt
Title:  No title for this article.
Description:  oddsmakers at betdsi sportsbook have ohtani fever as much as all those baseball scouts and writers tripping over themselves as
---------------------
Score:  0.038654316820208194
URL:  https://s2.smu.edu/~fmoore/misc/seo_email.txt
Title:  No title for this article.
Description:  original message from atul kumar mailto atul1930 outlook com sent thursday march 5 2020 12 58 am subject first page
---------------------
Score:  0.03464360745416091
URL:  https://s2.smu.edu/~fmoore/schedule.htm
Title:  smu CS 5337/7337 spring 2020 schedul
Description:  schedule smu cs 5337 7337 spring 2020 preliminary schedule this page is maintained as the latest schedule of content and
---------------------
Score:  0.033494638512305414
URL:  https://s2.smu.edu/~fmoore/textfiles/baseball3.txt
Title:  No title for this article.
Description:  ticket sales for the rangers are off after the team s 78 84 record last season according to a source
---------------------
Score:  0.025695692829510352
URL:  https://s2.smu.edu/~fmoore/textfiles/basketball2.txt
Title:  No title for this article.
Description:  fired a good man and now they are looking for the ark of the covenant without indiana jones – they

Input query you wish:
```

For query "*nowork*", word "*nowork*" is not in the dictionary, then we try to expand this word to its alternatives word "*no*" and "*work*" use expanded query "*nowork no work*" find articles.

```
Input query you wish: hoehaiuhbsfanck
---------------------------
No article related this query. Now let us try query expansion:
Sorry, we cannot find any result about this query.

Input query you wish:
```

In the above query we try to find the word "*hoehaiuhbsfanck"* which we could not find the word so we print the statement that the word is "Sorry, we cannot find any result about this query".

The user enters "words" per definition we crawled the text from the documents, we stem each word. So, after we get the query, we can split the query into query_list and then we stem each query word using Porter stemmer. Therefore, after stemming the query word, we the used stemmed query to calculate the cosine similarity.

```
shradhauttarwar > Desktop > IRProject12 > IRProject12 > QueryE
    ps = PorterStemmer()
    query_list_porter = []
    for original_word in query_list:
        word_porter = ps.stem(original_word)
        query_list_porter.append(word_porter)
```

Query document will make query as a document and calculated term frequency for each term in this query document and add this frequency list into dictionary document_list= {} (A dictionary that contain term frequency for each term in each document) for calculating cosine similarity. The key for query in the dictionary is assigned as 0.

```
Users > shradhauttarwar > Desktop > IRProject12 > IRProject12 > QueryEngine.py > ...
74      for matrix_word in words_matrix.keys():
75          words_matrix[matrix_word].append(0)
76
77      word_flag = False
78      for porter_word in query_list_porter:
79          for matrix_word in words_matrix.keys():
80              if matrix_word == porter_word:
81                  word_flag = True
82                  words_matrix[matrix_word][-1] += 1
83
84      if word_flag:
85          documents_list[0] = []
86          for matrix_word in words_matrix.keys():
87              documents_list[0].append(words_matrix[matrix_word][-1])
```

We have implemented thesaurus expansion on each term and then searching term documents. The words_expansion has in place regardless of its being found in document or not to be. Therefore, the words are alternative in the dictionary words expansion as shown in below:

words_expansion = {'beautiful': ['nice', 'fancy'], 'chapter': ['chpt'],

'chpt': ['chapter'], 'responsible': ['owner', 'accountable'], 'freemanmoore':

['freeman', 'moore'], 'dept': ['department'], 'photo': ['photograph', 'image',

'picture'], 'brown': ['beige', 'tan', 'auburn'], 'tues': ['Tuesday'], 'sole':

['owner', 'single', 'shoe', 'boot'], 'homework': ['hmwk', 'home', 'work'],

'novel': ['book', 'unique'], 'computer': ['cse'], 'story': ['novel', 'book'],

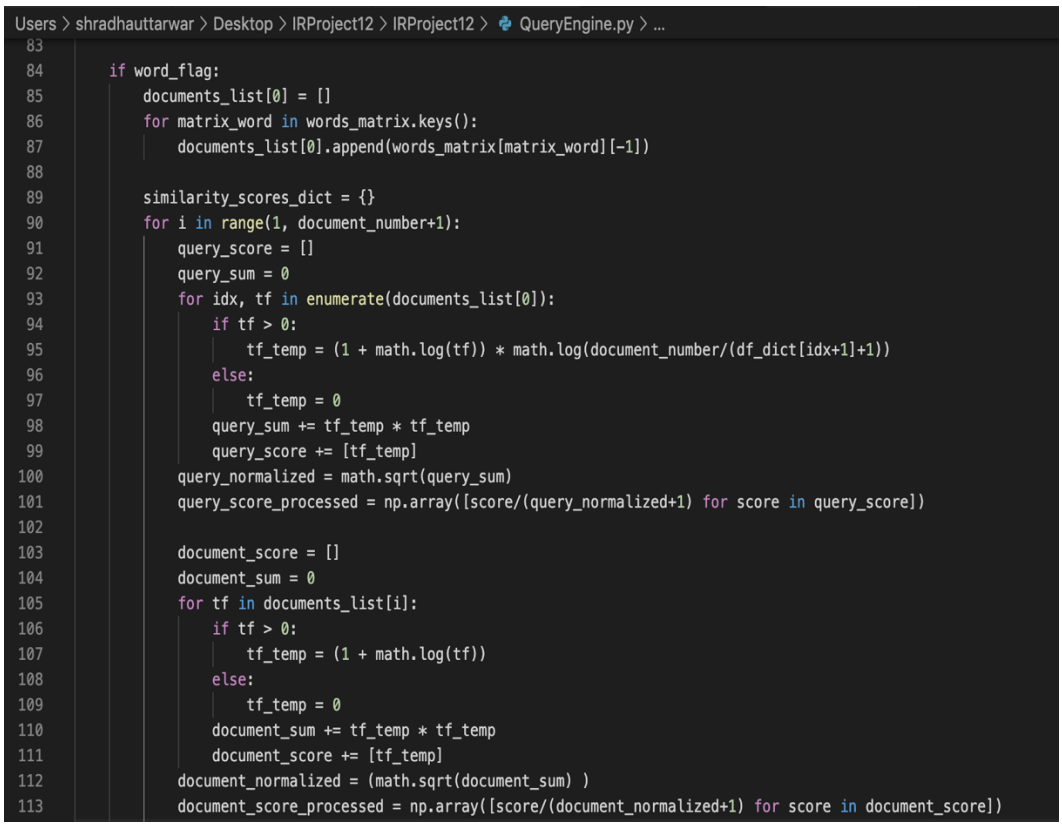
'hocuspocus': ['magic', 'abracadabra'], 'thisworks': ['this', 'work'], }

```
else:
    query_list = [query_word for query_word in last_query_list]
    for query_word in last_query_list:
        if words_expansion.get(query_word):
            for expanded_word in words_expansion[query_word]:
                query_list.append(expanded_word)
```

3. **Implement the cosine similarity of the query against each document. [40/40 points]**

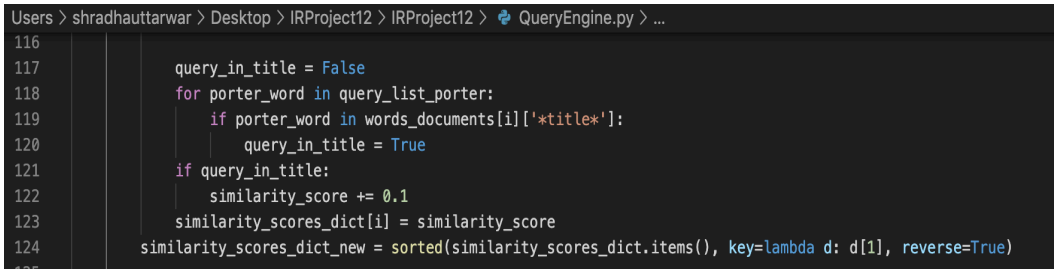
   a) **What document/query weighting scheme did you implement (suggestion: lnc.ltc)?**

   In weighting scheme used lnc.ltc to calculate cosine similarity. For document, using Inc scheme and for query, using ltc scheme. The term frequency of each term in each document and query is stored at dictionary document_list. The key of the dictionary is the unique document ID and special key 0 represents query. Then we use lnc scheme to process document and use ltc to process query. After that, we do product between query and calculate the cosine similarity. Then we store the results into the dictionary similarity_score_dict= {}. The key of this dictionary is unique document ID and the value of this dictionary is the cosine similarity between query and this document.



b) If any of the query words appear in the <title> of a document, add 0.1 to its score.

Adding score for title we have to remember can access the title of each dictionary words_documents= {} using unique document ID. For example, words_documents [1] [‘*title*’] represents the title of document 1. By doing this way we determined the query word appears in the title of each documents. If it does, then we add 0.1 to the cosine similarity score between query and this document.



c) Display the resulting score, document URL, and document title in descending numerical order for the top K=5 results.

```
similarity_scores_dict_new = sorted(similarity_scores_dict.items(), key=lambda d: d[1], reverse=True)
result_count = 0
for number, score in similarity_scores_dict_new[:5]:
    if score > 0:
        result_count += 1
        print('----------------------')
        print('Score: ', score)
        print('URL: ', words_documents[number]['*url*'])
        if not words_documents[number]['*title*']:
            title = 'No title for this article.'
        else:
            title = ' '.join(words_documents[number]['*title*'])
        print('Title: ', title)
        print('Description: ', words_documents[number]['*description*'])
if not is_expanded:
        print('--------------------------')
        print('No article related this query. Now you can reenter Query:')
        is_expanded = True
        last_query_list = [query_word for query_word in query_list]
```

Displaying the top 5 results after we get cosine similarity scores in dictionary similarity_score_dict= {}, we then sort this dictionary in descending order based on its cosine similarity score and print the score, title, url and the first 20 words of the top 5 documents. Remember we can access the title, url and the first 20 words of each document in dictionary words_documents= {} using unique document ID(word_documents[1][‘*title*’],words_documents[1][‘*url*’], words_documents[1][‘*description*’]).

Result shows below:

```
Input query you wish: three year story
--------------------
Score:  0.09698605355852488
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird4.html
Title:  mockingbird part 4
Description:  mockingbird part 4 the story takes place during three years 1933–35 of the great depression in the fictional tired old
--------------------
Score:  0.09447253232678378
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird1.html
Title:  mockingbird novel part 1
Description:  mockingbird part 1 to kill a mockingbird is primarily a novel about growing up under extraordinary circumstances in the 1930s
--------------------
Score:  0.06320967274651282
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird2.html
Title:  mockingbird part 2
Description:  mockingbird part 2 the only neighbor who puzzles them is the mysterious arthur radley nicknamed boo who never comes outside
--------------------
Score:  0.05239093230701331
URL:  https://s2.smu.edu/~fmoore/textfiles/basketball5.txt
Title:  No title for this article.
Description:  is a limited contact sport played on a rectangular court while most often played as a team sport with five
--------------------
Score:  0.04968828394775876
URL:  https://s2.smu.edu/~fmoore/textfiles/basketball1.txt
Title:  No title for this article.
Description:  normal circumstances the news that dirk nowitzki is done for the year would be devastating but thursday s announcement was
```

The query is "*three year story*", and we print the score, url, title and top 5 documents based on their cosine similarity score in descending order.

d) CS7337: Display the first 20 words of the document (this can be the stemmed version).

In order to save the first 20 words of the documents, we saved the first 20 words of the document into file 'words_porter. json' during web crawling phase. This file will then be read into dictionary words_documents= {} during query phase. We can access to the 20 words of the documents using unique document ID such as words_documents [1] [*description*]. Then we print the description when we show the results given the query.

For example, of 1st 5 documents and 20 words:

```
Input query you wish: three year story
--------------------
Score:  0.09698605355852488
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird4.html
Title:  mockingbird part 4
Description:  mockingbird part 4 the story takes place during three years 1933–35 of the great depression in the fictional tired old
--------------------
Score:  0.09447253232678378
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird1.html
Title:  mockingbird novel part 1
Description:  mockingbird part 1 to kill a mockingbird is primarily a novel about growing up under extraordinary circumstances in the 1930s
--------------------
Score:  0.06320967274651282
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird2.html
Title:  mockingbird part 2
Description:  mockingbird part 2 the only neighbor who puzzles them is the mysterious arthur radley nicknamed boo who never comes outside
--------------------
Score:  0.05239093230701331
URL:  https://s2.smu.edu/~fmoore/textfiles/basketball5.txt
Title:  No title for this article.
Description:  is a limited contact sport played on a rectangular court while most often played as a team sport with five
--------------------
Score:  0.04968828394775876
URL:  https://s2.smu.edu/~fmoore/textfiles/basketball1.txt
Title:  No title for this article.
Description:  normal circumstances the news that dirk nowitzki is done for the year would be devastating but thursday s announcement was
```

As we can see the query "*three year story*", we can see it returns 5 results and there is a description (first 20 words) of each results.

4. **Explain why you believe the results are correct. [15/15 points]**

   You must show the results of these queries (tentative):

   a) **<u>moore smu:</u>**

```
Input query you wish: moore smu
--------------------
Score:  0.31503030217522243
URL:  https://s2.smu.edu/~fmoore/
Title:  freeman moor - smu spring 2020
Description:  2020 freeman l moore phd email fmoore lyle smu ed u fall 2019 cs 5330 7330 keep looking at the
--------------------
Score:  0.1487599845963782
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/index.php
Title:  smu CS 5/7337 spring 2020 text files"
Description:  files additional text files to support query implementation part 1 part 2 part 3 part 4 part 5 hocuspocus 3
--------------------
Score:  0.14859451091675568
URL:  https://s2.smu.edu/~fmoore/textfiles/index.html
Title:  smu cse 5/7337 spring 2018 textfil
Description:  textfiles textfiles for clustering golf 1 golf 2 golf 3 golf 4 golf 5 basketball 1 basketball 2 basketball 3
--------------------
Score:  0.13718953030748318
URL:  https://s2.smu.edu/~fmoore/schedule.htm
Title:  smu CS 5337/7337 spring 2020 schedul
Description:  schedule smu cs 5337 7337 spring 2020 preliminary schedule this page is maintained as the latest schedule of content and
--------------------------
No article related this query. Now you can reenter Query:
--------------------
Score:  0.31503030217522243
URL:  https://s2.smu.edu/~fmoore/
Title:  freeman moor - smu spring 2020
Description:  2020 freeman l moore phd email fmoore lyle smu ed u fall 2019 cs 5330 7330 keep looking at the
--------------------
Score:  0.1487599845963782
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/index.php
Title:  smu CS 5/7337 spring 2020 text files"
Description:  files additional text files to support query implementation part 1 part 2 part 3 part 4 part 5 hocuspocus 3
--------------------
Score:  0.14859451091675568
URL:  https://s2.smu.edu/~fmoore/textfiles/index.html
Title:  smu cse 5/7337 spring 2018 textfil
Description:  textfiles textfiles for clustering golf 1 golf 2 golf 3 golf 4 golf 5 basketball 1 basketball 2 basketball 3
--------------------
Score:  0.13718953030748318
URL:  https://s2.smu.edu/~fmoore/schedule.htm
Title:  smu CS 5337/7337 spring 2020 schedul
Description:  schedule smu cs 5337 7337 spring 2020 preliminary schedule this page is maintained as the latest schedule of content and

Input query you wish:
```

Query "*moore smu*" returns 4 results and after 4 results it shows another result which is expanded query. The expanded query returns 4 results too, but which is similar result. Therefore, there are only 4 documents that are relevant to the query "moore smu".

For 1st document:
https://s2.smu.edu/~fmoore/ , moore and smu appears in both the title and the content of this documents.

For 2nd document:
https://s2.smu.edu/~fmoore/textfiles/extratextfiles/index.php
appears both in the title and the content of this document.

For 3rd and 4th documents:
https://s2.smu.edu/~fmoore/textfiles/index.html
https://s2.smu.edu/~fmoore/schedule.htm
smu appears in the title of the document

b) Bob Ewell where Scout

Result for "Bob Ewell Scout":

```
Input query you wish: Bob Ewell where Scout
----------
Score:  0.172887913202098
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird5.html
Title:  mockingbird part 5
Description:  mockingbird part 5 atticus does not want jem and scout to be present at tom robinson s trial no seat
----------
Score:  0.064930801283103
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird4.html
Title:  mockingbird part 4
Description:  mockingbird part 4 the story takes place during three years 1933–35 of the great depression in the fictional tired old
----------
Score:  0.0645829961452741 7
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird1.html
Title:  mockingbird novel part 1
Description:  mockingbird part 1 to kill a mockingbird is primarily a novel about growing up under extraordinary circumstances in the 1930s
----------
Score:  0.051609001028916655
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird3.html
Title:  mockingbird part 3
Description:  mockingbird part 3 suddenly scout and jem have to tolerate a barrage of racial slurs and insults because of atticus
----------
Score:  0.04352345183907334
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird2.html
Title:  mockingbird part 2
Description:  mockingbird part 2 the only neighbor who puzzles them is the mysterious arthur radley nicknamed boo who never comes outside
```

For this Query 5 results are returned:

For 1st document:
https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird5.html
Bob appears 3 times whereas Ewell and Scout appears 6 times in the content and where appears once in the content.

For 2nd document:
https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird4.html
where appears once and Scout appears 2 times.

For 3rd document:
https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird1.html
Ewell and Scout appears in the content

For 4th document:
https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird3.html
After visiting this link, we can see Scout appears 4 times in the content.

For 5th document:
https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird2.html
After visiting this link, we can see Scout appears 3 times in the content.

c) <u>three year story</u>

```
Input query you wish: three year story
--------------------
Score:  0.09698605355852488
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird4.html
Title:  mockingbird part 4
Description:  mockingbird part 4 the story takes place during three years 1933–35 of the great depression in the fictional tired old
--------------------
Score:  0.09447253232678378
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird1.html
Title:  mockingbird novel part 1
Description:  mockingbird part 1 to kill a mockingbird is primarily a novel about growing up under extraordinary circumstances in the 1930s
--------------------
Score:  0.06320967274651282
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird2.html
Title:  mockingbird part 2
Description:  mockingbird part 2 the only neighbor who puzzles them is the mysterious arthur radley nicknamed boo who never comes outside
--------------------
Score:  0.05239093230701331
URL:  https://s2.smu.edu/~fmoore/textfiles/basketball5.txt
Title:  No title for this article.
Description:  is a limited contact sport played on a rectangular court while most often played as a team sport with five
--------------------
Score:  0.04968828394775876
URL:  https://s2.smu.edu/~fmoore/textfiles/basketball1.txt
Title:  No title for this article.
Description:  normal circumstances the news that dirk nowitzki is done for the year would be devastating but thursday s announcement was
```

Result shows 5 document for this query.

For 1st document:
https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird4.html
Query word Appears in this content.

For 2nd document:
https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird1.html
Query word Appears in this content.

For 3rd document:
https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird2.html
Query word Appears in this content.

For 4th document:
https://s2.smu.edu/~fmoore/textfiles/basketball5.txt
Query word Appears in this content.

For 5th document:
https://s2.smu.edu/~fmoore/textfiles/basketball1.txt
Query word Appears in this content.

d) Atticus to defend Maycomb

```
Input query you wish: Atticus to defend Maycomb
--------------------
Score:  0.17256190070635355
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird4.html
Title:  mockingbird part 4
Description:  mockingbird part 4 the story takes place during three years 1933–35 of the great depression in the fictional tired old
--------------------
Score:  0.13205000158378427
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird1.html
Title:  mockingbird novel part 1
Description:  mockingbird part 1 to kill a mockingbird is primarily a novel about growing up under extraordinary circumstances in the 1930s
--------------------
Score:  0.072555913975665555
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird5.html
Title:  mockingbird part 5
Description:  mockingbird part 5 atticus does not want jem and scout to be present at tom robinson s trial no seat
--------------------
Score:  0.044322212893428255
URL:  https://s2.smu.edu/~fmoore/textfiles/baseball2.txt
Title:  No title for this article.
Description:  s opening day and for one day our team is going to win the whole bleeping thing it s opening
--------------------
Score:  0.041069353485050086
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird3.html
Title:  mockingbird part 3
Description:  mockingbird part 3 suddenly scout and jem have to tolerate a barrage of racial slurs and insults because of atticus
```

Result shows 5 document for the query:
For 1st document:
https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird4.html
Query word Appears in this content.

For 2nd document:
https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird1.html
Query word Appears in this content.

For 3rd document:
https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird5.html
Query word Appears in this content.

For 4th document:
https://s2.smu.edu/~fmoore/textfiles/baseball2.txt
Query word Appears in this content.

For 5th document:
https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird3.html
Query word Appears in this content.

e) <u>hocuspocus thisworks</u>

```
Input query you wish: hocuspocus thiswork
---------------------
Score:  0.08189520556775087
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/index.php
Title:  smu CS 5/7337 spring 2020 text files"
Description:  files additional text files to support query implementation part 1 part 2 part 3 part 4 part 5 hocuspocus 3
---------------------------
No article related this query. Now you can reenter Query:
```

Query “hocuspocus thisworks” returns 1 result which is less then 5 results. So, we try to expand the query “hocuspocus thisworks”

The expanded query “hocuspocus thisworks magic abracadabra this work” returns 3 results this time.

For 1st document:
https://s2.smu.edu/~fmoore/textfiles/extratextfiles/magictext.html
Query word Appears in this content.

For 2nd document:
https://s2.smu.edu/~fmoore/textfiles/extratextfiles/index.php
Query word Appears in this content.

For 3rd document:
https://s2.smu.edu/~fmoore/textfiles/baseball5.txt
Query word Appears in this content.

5. **Instructor testing with additional queries. [15/15 points]**

   Answer: You can also check, just follow the *"Readme file"*.

   1. **beautiful:** This query result shows below also shows the expanded result.

```
Input query you wish: beautiful
----------------------
Score:  0.05266360012571011
URL:  https://s2.smu.edu/~fmoore/textfiles/baseball1.txt
Title:  No title for this article.
Description:  texas rangers have to be at their most presentable with cole hamels pitching and if thursday is the best version
---------------------------
No article related this query. Now you can reenter Query:
----------------------
Score:  0.05176960701656745
URL:  https://s2.smu.edu/~fmoore/textfiles/golf5.txt
Title:  No title for this article.
Description:  resurgent poulter will have that chance sunday after shooting a 4 under par 67 to take a one shot lead
----------------------
Score:  0.04020464212013578
URL:  https://s2.smu.edu/~fmoore/textfiles/baseball1.txt
Title:  No title for this article.
Description:  texas rangers have to be at their most presentable with cole hamels pitching and if thursday is the best version

Input query you wish:
```

      URL: https://s2.smu.edu/~fmoore/textfiles/baseball1.txt

      This URL content word beautiful.

   2. **chapter:** There is not result for chapter therefore the expanded query '*chpt*' document is shown below,

```
Input query you wish: chapter
--------------------------
No article related this query. Now let us try query expansion:
----------------------
Score:  0.1404520810435768
URL:  https://s2.smu.edu/~fmoore/schedule.htm
Title:  smu CS 5337/7337 spring 2020 schedul
Description:  schedule smu cs 5337 7337 spring 2020 preliminary schedule this page is maintained as the latest schedule of content and
```

      URL: https://s2.smu.edu/~fmoore/schedule.htm

3. **chpt:** This query result shows below only 1 document have shown result.

   URL： https://s2.smu.edu/~fmoore/schedule.htm

```
Input query you wish: chpt
----------------------
Score:  0.1342486096984444
URL:  https://s2.smu.edu/~fmoore/schedule.htm
Title:  smu CS 5337/7337 spring 2020 schedul
Description:  schedule smu cs 5337 7337 spring 2020 preliminary schedule this pa
ge is maintained as the latest schedule of content and
--------------------------
No article related this query. Now you can reenter Query:
----------------------
Score:  0.1342486096984444
URL:  https://s2.smu.edu/~fmoore/schedule.htm
Title:  smu CS 5337/7337 spring 2020 schedul
Description:  schedule smu cs 5337 7337 spring 2020 preliminary schedule this pa
ge is maintained as the latest schedule of content and

Input query you wish:
```

   Extended Url content the same result so only one document is relevant.

4. **responsible:** This query result shows below also shows the expanded result.

```
Input query you wish: responsible
---------------------
Score:  0.0604465946854374
URL:  https://s2.smu.edu/~fmoore/
Title:  freeman moor - smu spring 2020
Description:  2020 freeman l moore phd email fmoore lyle smu ed u fall 2019 cs 5330 7330 keep looking at the
---------------------
Score:  0.0577845944345048
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird5.html
Title:  mockingbird part 5
Description:  mockingbird part 5 atticus does not want jem and scout to be present at tom robinson s trial no seat
---------------------
Score:  0.04079005947397189
URL:  https://s2.smu.edu/~fmoore/misc/seo_email.txt
Title:  No title for this article.
Description:  original message from atul kumar mailto atul1930 outlook com sent thursday march 5 2020 12 58 am subject first page
--------------------------
No article related this query. Now you can reenter Query:
---------------------
Score:  0.04344653371691458
URL:  https://s2.smu.edu/~fmoore/textfiles/baseball2.txt
Title:  No title for this article.
Description:  s opening day and for one day our team is going to win the whole bleeping thing it s opening
---------------------
Score:  0.041767611115076105
URL:  https://s2.smu.edu/~fmoore/
Title:  freeman moor - smu spring 2020
Description:  2020 freeman l moore phd email fmoore lyle smu ed u fall 2019 cs 5330 7330 keep looking at the
---------------------
Score:  0.03992821235917886
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird5.html
Title:  mockingbird part 5
Description:  mockingbird part 5 atticus does not want jem and scout to be present at tom robinson s trial no seat
---------------------
Score:  0.028185265858824873
URL:  https://s2.smu.edu/~fmoore/misc/seo_email.txt
Title:  No title for this article.
Description:  original message from atul kumar mailto atul1930 outlook com sent thursday march 5 2020 12 58 am subject first page

Input query you wish:
```

5. **freemanmoore:** There is not result for query “freemanmoore” therefore the expanded query ‘*freeman moore*’ document is shown below:

```
Input query you wish: freemanmoore
--------------------------
No article related this query. Now let us try query expansion:
--------------------
Score:  0.345059952060554
URL:  https://s2.smu.edu/~fmoore/
Title:  freeman moor - smu spring 2020
Description:  2020 freeman l moore phd email fmoore lyle smu ed u fall 2019 cs 5330 7330 keep looking at the
```

6. **dept:** There is no result shown, not even expanded query.

```
Input query you wish: dept
--------------------------
No article related this query. Now let us try query expansion:
Sorry, we cannot find any result about this query.
```

7. **photo:** There is no result shown, not even expanded query.

```
Input query you wish: photo
--------------------------
No article related this query. Now let us try query expansion:
Sorry, we cannot find any result about this query.
```

8. **brown:** This query result shows below, there are 5 documents resulted related to brown.

```
Input query you wish: brown
----------------------
Score:  0.3048160515790603
URL:  https://s2.smu.edu/~fmoore/textfiles/cow1.txt
Title:  No title for this article.
Description:  brown is the brown cow
----------------------
Score:  0.2699788495141498
URL:  https://s2.smu.edu/~fmoore/textfiles/cow4.txt
Title:  No title for this article.
Description:  cow
----------------------
Score:  0.21726220001698174
URL:  https://s2.smu.edu/~fmoore/textfiles/cow2.txt
Title:  No title for this article.
Description:  now brown cow
----------------------
Score:  0.18002927038998093
URL:  https://s2.smu.edu/~fmoore/textfiles/cow3.txt
Title:  No title for this article.
Description:  cow is a brown cow
----------------------
Score:  0.03213994433135377
URL:  https://s2.smu.edu/~fmoore/schedule.htm
Title:  smu CS 5337/7337 spring 2020 schedul
Description:  schedule smu cs 5337 7337 spring 2020 preliminary schedule this page is maintained as the latest schedule of content and
---------------------------
No article related this query. Now you can reenter Query:
----------------------
Score:  0.3048160515790603
URL:  https://s2.smu.edu/~fmoore/textfiles/cow1.txt
Title:  No title for this article.
Description:  brown is the brown cow
----------------------
Score:  0.2699788495141498
URL:  https://s2.smu.edu/~fmoore/textfiles/cow4.txt
Title:  No title for this article.
Description:  cow
----------------------
Score:  0.21726220001698174
URL:  https://s2.smu.edu/~fmoore/textfiles/cow2.txt
Title:  No title for this article.
Description:  now brown cow
----------------------
Score:  0.18002927038998093
URL:  https://s2.smu.edu/~fmoore/textfiles/cow3.txt
Title:  No title for this article.
Description:  cow is a brown cow
----------------------
Score:  0.03213994433135377
URL:  https://s2.smu.edu/~fmoore/schedule.htm
Title:  smu CS 5337/7337 spring 2020 schedul
Description:  schedule smu cs 5337 7337 spring 2020 preliminary schedule this page is maintained as the latest schedule of content and

Input query you wish:
```

9. **tues:** There is not result for query "*tues*" therefore the expanded query '*Tuesday*' document is shown below:

```
Input query you wish: tues
---------------------------
No article related this query. Now let us try query expansion:
----------------------
Score:  0.05935465695968675
URL:  https://s2.smu.edu/~fmoore/textfiles/football4.txt
Title:  No title for this article.
Description:  cowboys vice president stephen jones commented on the release of
eceiver dez bryant for the first time tuesday jones said

Input query you wish:
```

10. **sole:** This query result shows below also shows the expanded result.

```
Input query you wish: sole
----------------------
Score:  0.06508050313047531
URL:  https://s2.smu.edu/~fmoore/
Title:  freeman moor - smu spring 2020
Description:  2020 freeman l moore phd email fmoore lyle smu ed u fall 2019 cs 5330 7330 keep looking at the
----------------------------
No article related this query. Now you can reenter Query:
----------------------
Score:  0.04968400054522204
URL:  https://s2.smu.edu/~fmoore/
Title:  freeman moor - smu spring 2020
Description:  2020 freeman l moore phd email fmoore lyle smu ed u fall 2019 cs 5330 7330 keep looking at the
----------------------
Score:  0.03962132638954409
URL:  https://s2.smu.edu/~fmoore/textfiles/baseball2.txt
Title:  No title for this article.
Description:  s opening day and for one day our team is going to win the whole bleeping thing it s opening

Input query you wish:
```

11. **<u>homework:</u>** There is not result for query "*homework*" therefore the expanded query '*hmwk home work*' document is shown below:

```
Input query you wish: homework
--------------------------
No article related this query. Now let us try query expansion:
----------------------
Score:  0.10537658283054217
URL:  https://s2.smu.edu/~fmoore/schedule.htm
Title:  smu CS 5337/7337 spring 2020 schedul
Description:  schedule smu cs 5337 7337 spring 2020 preliminary schedule this page is maintained as the latest schedule of content and
----------------------
Score:  0.03445883261389325
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird5.html
Title:  mockingbird part 5
Description:  mockingbird part 5 atticus does not want jem and scout to be present at tom robinson s trial no seat
----------------------
Score:  0.031143142489181604
URL:  https://s2.smu.edu/~fmoore/misc/seo_email.txt
Title:  No title for this article.
Description:  original message from atul kumar mailto atul1930 outlook com sent thursday march 5 2020 12 58 am subject first page
----------------------
Score:  0.023533280609195992
URL:  https://s2.smu.edu/~fmoore/textfiles/baseball1.txt
Title:  No title for this article.
Description:  texas rangers have to be at their most presentable with cole hamels pitching and if thursday is the best version
----------------------
Score:  0.022722136742997865
URL:  https://s2.smu.edu/~fmoore/textfiles/basketball1.txt
Title:  No title for this article.
Description:  normal circumstances the news that dirk nowitzki is done for the year would be devastating but thursday s announcement was

Input query you wish:
```

12. **<u>novel:</u>** This query result shows below and also shows the expanded result.

```
Input query you wish: novel
----------------------
Score:  0.1948090458782001
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird1.html
Title:  mockingbird novel part 1
Description:  mockingbird part 1 to kill a mockingbird is primarily a novel about growing up under extraordinary circumstances in the 1930s
--------------------------
No article related this query. Now you can reenter Query:
----------------------
Score:  0.1948090458782001
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird1.html
Title:  mockingbird novel part 1
Description:  mockingbird part 1 to kill a mockingbird is primarily a novel about growing up under extraordinary circumstances in the 1930s
```

13. **<u>computer:</u>** There is not result for query "*computer*" therefore the expanded query '*cse*' document is shown below:

```
Input query you wish: computer
---------------------------
No article related this query. Now let us try query expansion:
----------------------
Score:  0.18161728298182483
URL:  https://s2.smu.edu/~fmoore/textfiles/index.html
Title:  smu cse 5/7337 spring 2018 textfil
Description:  textfiles textfiles for clustering golf 1 golf 2 golf 3 golf 4 golf 5 basketball 1 basketball 2 basketball 3

Input query you wish:
```

14. **story:** This query result shows below and also shows the expanded result.

```
Input query you wish: story
----------------------
Score:  0.05034007174288458
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird1.html
Title:  mockingbird novel part 1
Description:  mockingbird part 1 to kill a mockingbird is primarily a novel about growing up under extraordinary circumstances in the 1930s
----------------------
Score:  0.04670248852855358
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird2.html
Title:  mockingbird part 2
Description:  mockingbird part 2 the only neighbor who puzzles them is the mysterious arthur radley nicknamed boo who never comes outside
----------------------
Score:  0.03412274563928915
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird4.html
Title:  mockingbird part 4
Description:  mockingbird part 4 the story takes place during three years 1933–35 of the great depression in the fictional tired old
----------------------
Score:  0.033033545254433264
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird5.html
Title:  mockingbird part 5
Description:  mockingbird part 5 atticus does not want jem and scout to be present at tom robinson s trial no seat
---------------------------
No article related this query. Now you can reenter Query:
----------------------
Score:  0.21496873824673213
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird1.html
Title:  mockingbird novel part 1
Description:  mockingbird part 1 to kill a mockingbird is primarily a novel about growing up under extraordinary circumstances in the 1930s
----------------------
Score:  0.030930457151792632
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird2.html
Title:  mockingbird part 2
Description:  mockingbird part 2 the only neighbor who puzzles them is the mysterious arthur radley nicknamed boo who never comes outside
----------------------
Score:  0.02259905532126555
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird4.html
Title:  mockingbird part 4
Description:  mockingbird part 4 the story takes place during three years 1933–35 of the great depression in the fictional tired old
----------------------
Score:  0.02187769192297675
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/mockingbird5.html
Title:  mockingbird part 5
Description:  mockingbird part 5 atticus does not want jem and scout to be present at tom robinson s trial no seat

Input query you wish:
```

15. **hocuspocus:** This query result shows below and also shows the expanded result.

```
Input query you wish: hocuspocus
----------------------
Score:  0.08189520556775087
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/index.php
Title:  smu CS 5/7337 spring 2020 text files"
Description:  files additional text files to support query implementation part 1 part 2 part 3 part 4 part 5 hocuspocus 3
----------------------------
No article related this query. Now you can reenter Query:
----------------------
Score:  0.27756189592705854
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/magictext.html
Title:  thi is the magic file
Description:  magic shows up here and in the title brown        beige   tan    auburn
----------------------
Score:  0.06602754269175717
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/index.php
Title:  smu CS 5/7337 spring 2020 text files"
Description:  files additional text files to support query implementation part 1 part 2 part 3 part 4 part 5 hocuspocus 3
----------------------
Score:  0.02942547213772617
URL:  https://s2.smu.edu/~fmoore/textfiles/baseball5.txt
Title:  No title for this article.
Description:  mann ray people will come ray they ll come to iowa for reasons they can t even fathom they ll

Input query you wish:
```

16. **thisworks:** There is not result for query "*thiswork*" therefore the expanded query '*this work'* document is shown below:

```
Input query you wish: thiswork
----------------------------
No article related this query. Now let us try query expansion:
Sorry, we cannot find any result about this query.

Input query you wish:
```

```
Input query you wish: this work
----------------------
Score:  0.14669291132566536
URL:  https://s2.smu.edu/~fmoore/textfiles/extratextfiles/magictext.html
Title:  thi is the magic file
Description:  magic shows up here and in the title brown        beige   tan    auburn
----------------------
Score:  0.06536835341303449
URL:  https://s2.smu.edu/~fmoore/misc/seo_email.txt
Title:  No title for this article.
Description:  original message from atul kumar mailto atul1930 outlook com sent thursday march 5 2020 12 58 am subject first page
----------------------
Score:  0.044771061983734387
URL:  https://s2.smu.edu/~fmoore/textfiles/basketball3.txt
Title:  No title for this article.
Description:  24 win season mavs coach careful not to promise too much for next season s basketball team our fans deserve
----------------------
Score:  0.037300292874449144
URL:  https://s2.smu.edu/~fmoore/textfiles/football1.txt
Title:  No title for this article.
Description:  the first time since dez bryant was released last friday cowboy head coach jason garrett and team captions spoke publicly
----------------------
Score:  0.034156228153657014
URL:  https://s2.smu.edu/~fmoore/
Title:  freeman moor – smu spring 2020
Description:  2020 freeman l moore phd email fmoore lyle smu ed u fall 2019 cs 5330 7330 keep looking at the
```

Source Code:

IRProject12 for crawl:

```
#ShradhaUttarwar

import scrapy
from scrapy.http import Request
from scrapy.selector import Selector
import re
import json
import io
import hashlib
from nltk.stem import PorterStemmer

N = int(input("Enter value of N=40. 40 document to be indexed :"))
class IRProject12(scrapy.Spider):
    name = 'IRProject12'
    start_urls = ['https://s2.smu.edu/~fmoore/', ]
    dup_dic = {}
    count = 0


    def parse(self, response): #parse function is defined
        selector = Selector(response)
        md = hashlib.md5()
        md.update(response.body)
        hashed_data = md.hexdigest()
        outgoing_urls = []
        ingoing_urls = []
        full_url = []
        broken_urls = []
        nontxt_urls = []
        in_going_pdf_xlsx_urls = []
        duplicated_urls = []
        meta_noindex = []
        URL_Frontier = []
        graphic_urls = []

        if hashed_data in self.dup_dic:
            duplicated_urls.append(response.url)
        else:
            self.dup_dic[hashed_data] = 1

            if response.status == 404:
                broken_urls.append(response.url)
```

```
            if response.status == 200:
                if self.count < int(N): #Limiting the number of pages to N
                    self.count += 1
                    word_dic = {}
                    title = selector.xpath('//title') #extracting titles
                    title = ''.join(title.xpath('string(.)').extract())
                    title = title.replace('\n', '').replace('\r', '').strip()
                    title = re.sub(r"\s{2,}", " ", title)
                    urls = selector.xpath('//a/@href | //img/@src').extract()
                    for url_part in urls: #filtering out URLs outside the domain of
http and fmoore
                        if re.match(r'^http', url_part):
                            url = url_part
                        else:
                            if not re.search(r'~fmoore/$', response.url):
                                response_list = response.url.split('/')
                                del response_list[-1]
                                url = '/'.join(response_list) + '/' + url_part
                            else: url = response.url + url_part

                        if re.match(r'^https://s2.smu.edu/~fmoore/', url) or
re.match(r'^http://s2.smu.edu/~fmoore/', url):
                            ingoing_urls.append(url)
                            # appending non-text URLs
                            if not (re.search(r'txt$', url_part) or re.search(r'htm$',
url_part) or re.search(r'html$', url_part) or re.search(r'php$', url_part)):
                                nontxt_urls.append(url)
                            #filer out non-txt URLs (pdf, xlsx, jpg, jpeg, png, gif,
pptx, dontgohere)
                            if re.search(r'gif$', url_part) or re.search(r'jpg$',
url_part) or re.search(r'jpeg$', url_part) or re.search(r'png$', url_part) or
re.search(r'pptx$',url_part):
                                graphic_urls.append(url)
                            elif re.search(r'pdf$', url) or re.search(r'xlsx$', url):
                                in_going_pdf_xlsx_urls.append(url)
                            elif 'dontgohere/' in url:
                                broken_urls.append(url)
                            elif '@lyle' in url:
                                broken_urls.append(url)
                            else: URL_Frontier.append(url)
                        else:
                            outgoing_urls.append(url)
                            full_url.append(url)


                    noindex = False
                    meta_tag = selector.xpath('//head/meta/@content').extract()
                    if meta_tag:
```

```
                        if re.match(r'^noindex', ''.join(meta_tag).lower()):
                            self.count -= 1
                            meta_noindex.append(response.url)
                            noindex = True
                            exit

                    if not noindex: #further processing
                        with io.open('indexed_urls.txt', 'a', encoding='utf-8') as
file:
                            url_title = response.url + ' --- ' + title
                            file.write(url_title)
                            file.write('\n')
                        ps = PorterStemmer()
                        texts = '
'.join(selector.xpath('//*/text()').extract()).replace('\n', ' ').replace('\r', '
').strip()
                        special_symbols = [',', '.', ';', '?', '(', ')', '[', ']',
'&', '!', '*', '@', '#', '$', '%', ':',"'", '=',
                        '+', "\\", '/', '"', '-', '{', '}', '<', '>']

                        for special_symbol in special_symbols:
                            texts = texts.replace(special_symbol, ' ')
                        new_text = re.sub(r"\s{2,}", " ", texts)
                        new_text = new_text.lower()
                        text_list = new_text.split(' ')
                        for word in text_list:
                            if word != '':
                                word_porter = ps.stem(word)
                                if word_porter not in word_dic:
                                    word_dic[word_porter] = 1
                                else: word_dic[word_porter] += 1
                        word_dic['*DocID*'] = self.count
                        title = title.strip()
                        title_list = title.split(' ')
                        word_dic['*title*'] = []
                        for title_word in title_list:
                            if title_word.strip() != '':
                                word_title_porter = ps.stem(title_word.strip())
                                word_dic['*title*'].append(word_title_porter)

                        word_dic['*description*'] = '
'.join(text_list[len(title_list):len(title_list)+20])
                        word_dic['*url*'] = response.url

                        with io.open('words_porter.json', 'a', encoding='utf-8') as
file:
                            file.write(json.dumps(word_dic, ensure_ascii=False))
                            line = u"\r\n"
```

```
                            file.write(line)


                else:
                  print("Too high value of N")
                  exit
        if URL_Frontier:
            for in_url in URL_Frontier:
                yield Request(in_url, callback=self.parse, dont_filter=True)

        if outgoing_urls:
            for outgoing_url in outgoing_urls:
                with io.open('outgoing_urls.txt', 'a', encoding='utf-8') as file:
                    file.write(outgoing_url)
                    file.write('\n')

        if broken_urls:
            for broken_url in broken_urls:
                with io.open('broken_urls.txt', 'a', encoding='utf-8') as file1:
                    file1.write(broken_url)
                    file1.write('\n')

        if nontxt_urls:
            for nontxt_url in nontxt_urls:
                with io.open('nontext_urls.txt', 'a', encoding='utf-8') as file2:
                    file2.write(nontxt_url)
                    file2.write('\n')

        if graphic_urls:
            for graphic_url in graphic_urls:
                with io.open('graphic_urls.txt', 'a', encoding='utf-8') as file2:
                    file2.write(graphic_url)
                    file2.write('\n')

        if in_going_pdf_xlsx_urls:
            for in_going_pdf_xlsx_url in in_going_pdf_xlsx_urls:
                with io.open('in_going_pdf_xlsx_urls.txt', 'a', encoding='utf-8') as
file3:
                    file3.write(in_going_pdf_xlsx_url)
                    file3.write('\n')

        if duplicated_urls:
            for duplicated_url in duplicated_urls:
                with io.open('duplicated_urls.txt', 'a', encoding='utf-8') as file4:
                    file4.write(duplicated_url)
                    file4.write('\n')

        if meta_noindex:
```

```
            for meta_noindex_url in meta_noindex:
                with io.open('meta_noindex.txt', 'a', encoding='utf-8') as file5:
                    file5.write(meta_noindex_url)
                    file5.write('\n')
```

Term frequency matrix:

```
#ShradhaUttarwar

import io
import json
word_dic_full = {}
for line in io.open("words_porter.json", 'r', encoding='utf-8'):
    new_line = line.replace('\n', '').replace('\r', '').strip()
    if new_line != '':
        word_dic = json.loads(new_line)
        for key in word_dic.keys():
            if key != '*DocID*' and key != '*description*' and key != '*url*' and key
!= '*title*':
                if key not in word_dic_full:
                    word_dic_full[key] = [0 for _ in range(40)]
                    word_dic_full[key][word_dic['*DocID*']-1] += word_dic[key]
                else:
                    word_dic_full[key][word_dic['*DocID*']-1] += word_dic[key]
with io.open('Words_Frequency_Matrix_porter_query.json', 'a', encoding='utf-8') as
file:
    for item in word_dic_full.keys():
        word_dic_temp = {}
        word_dic_temp[item] = word_dic_full[item]
        file.write(json.dumps(word_dic_temp, ensure_ascii=False))
        line = u"\r\n"
        file.write(line)
```

Query Engine:

```
# -*- coding: utf-8 -*-
"""
Created on Sat May  2 16:49:18 2020

@author: Shradha
"""
import json
import io
import math
import numpy as np
```

```
from nltk.stem import PorterStemmer

number_punctuations = [',', '.', ':', ';', '?', '(', ')', '[', ']', '&', '!', '*',
'@', '#', '$', '%', ': ', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9']

words_documents = {}
document_number = 0

for line in io.open("words_porter.json", 'r', encoding='utf-8'):
    new_line = line.replace('\n', '').replace('\r', '').strip()
    if new_line != '':
        document_number += 1
        word_dic = json.loads(new_line)
        words_documents[document_number] = word_dic

words_matrix = {}
for line in io.open("Words_Frequency_Matrix_porter_query.json", 'r', encoding='utf-
8'):
    new_line = line.replace('\n', '').replace('\r', '').strip()
    if new_line != '':
        word_dic_temp = json.loads(new_line)
        for key in word_dic_temp:
            words_matrix[key] = word_dic_temp[key]

documents_list = {}
for word in words_matrix.keys():
    for idx, frequency in enumerate(words_matrix[word]):
        if idx+1 not in documents_list:
            documents_list[idx+1] = []
        documents_list[idx+1].append(frequency)

idf_dict = {}
word_number = 0
for word in words_matrix.keys():
    word_number += 1
    idf_dict[word_number] = document_number - words_matrix[word].count(0) + 1

words_expansion = {'beautiful': ['nice', 'fancy'], 'chapter': ['chpt'],
'chpt': ['chapter'], 'responsible': ['owner', 'accountable'], 'freemanmoore':
['freeman', 'moore'], 'dept': ['department'], 'photo': ['photograph', 'image',
'picture'], 'brown': ['beige', 'tan', 'auburn'], 'tues': ['Tuesday'], 'sole':
['owner', 'single', 'shoe', 'boot'], 'homework': ['hmwk', 'home', 'work'],
'novel': ['book', 'unique'], 'computer': ['cse'], 'story': ['novel', 'book'],
'hocuspocus': ['magic', 'abracadabra'], 'thisworks': ['this', 'work'], }
is_expanded = False
last_query_list = []
while True:
    if not is_expanded:
```

```
        query = input('\nInput query you wish: ')

        if query == 'stop':
            print('\nClosing the query engine...')
            break
        query_list = query.split(' ')
    else:
        query_list = [query_word for query_word in last_query_list]
        for query_word in last_query_list:
            if words_expansion.get(query_word):
                for expanded_word in words_expansion[query_word]:
                    query_list.append(expanded_word)

    ps = PorterStemmer()
    query_list_porter = []
    for original_word in query_list:
        word_porter = ps.stem(original_word)
        query_list_porter.append(word_porter)

    for matrix_word in words_matrix.keys():
        words_matrix[matrix_word].append(0)


    word_flag = False
    for porter_word in query_list_porter:
        for matrix_word in words_matrix.keys():
            if matrix_word == porter_word:
                word_flag = True
                words_matrix[matrix_word][-1] += 1

    if word_flag:
        documents_list[0] = []
        for matrix_word in words_matrix.keys():
            documents_list[0].append(words_matrix[matrix_word][-1])

        similarity_scores_dict = {}
        for i in range(1, document_number+1):
            query_score = []
            query_sum = 0
            for idx, tf in enumerate(documents_list[0]):
                if tf > 0:
                    tf_temp = (1 + math.log(tf)) *
math.log(document_number/(idf_dict[idx+1]+1))
                else:
                    tf_temp = 0
                query_sum += tf_temp * tf_temp
                query_score += [tf_temp]
            query_normalized = math.sqrt(query_sum)
```

```
            query_score_processed = np.array([score/(query_normalized+1) for score in
query_score])

            document_score = []
            document_sum = 0
            for tf in documents_list[i]:
                if tf > 0:
                    tf_temp = (1 + math.log(tf))
                else:
                    tf_temp = 0
                document_sum += tf_temp * tf_temp
                document_score += [tf_temp]
            document_normalized = (math.sqrt(document_sum) )
            document_score_processed = np.array([score/(document_normalized+1) for
score in document_score])

            similarity_score = np.dot(query_score_processed, document_score_processed)

            query_in_title = False
            for porter_word in query_list_porter:
                if porter_word in words_documents[i]['*title*']:
                    query_in_title = True
            if query_in_title:
                similarity_score += 0.1
            similarity_scores_dict[i] = similarity_score
        similarity_scores_dict_new = sorted(similarity_scores_dict.items(), key=lambda
d: d[1], reverse=True)
        result_count = 0
        for number, score in similarity_scores_dict_new[:5]:
            if score > 0:
                result_count += 1
                print('----------------------')
                print('Score: ', score)
                print('URL: ', words_documents[number]['*url*'])
                if not words_documents[number]['*title*']:
                    title = 'No title for this article.'
                else:
                    title = ' '.join(words_documents[number]['*title*'])
                print('Title: ', title)
                print('Description: ', words_documents[number]['*description*'])
        if not is_expanded:
                print('----------------------------')
                print('No article related this query. Now you can reenter Query:')
                is_expanded = True
                last_query_list = [query_word for query_word in query_list]
        else:
           if is_expanded:
            is_expanded = False
```

```
            last_query = []

    else:
        if is_expanded:
            is_expanded = False
            print('Sorry, we cannot find any result about this query.')
        else:
            print('----------------------------')
            print('No article related this query. Now let us try query expansion:')
            is_expanded = True
            last_query_list = [query_word for query_word in query_list]
```

----------End----------