Experiment-01

Name: Atharva Paliwal

Roll No: 40

Date: 21/07/2021

Aim: Write a program to implement a small search engine. Consider 10 documents/web pages as the search domain.

A. Create inverted index for the documents.

B. Create a search system for answering the search queries using the index created.

C. Modify the index to add the term frequency information and Rank the output based on term frequency.

Details:

- The documents can be simple text files with contents belonging to different domains or downloaded webpages in HTML format, You can parse the tags using regular expressions to extract the text from the corresponding tags.
- The documnts/web pages may contain non-alphanumeric characters and capital letters. However, we don't want to index apple, Apple, and APPLE differently. They are basically all apple.
- Handle grammer: Considering the grammatical structure of the language, we don't want to index research, researches, and researching separately. They are all about research. When we search for one of these terms we would expect results containing any of those variations.
- Removal of Stop words: We wouldn't like to index words such as 'the', 'a', 'an' because they appear in almost every document and they don't give us very much information about the document or the query.
- Query Types/ Searching- One Word Queries (Queries that consist of a single word), Multi-word queries

Theory:

An inverted index is a data structure that we build while parsing the documents that we are going to answer the search queries on. Given a query, we use the index to return the list of documents relevant for this query. The inverted index contains mappings from terms (words) to the documents that those terms appear in. Each vocabulary term is a key in the index whose value is its postings list. A term's postings list is the list of documents that the term appears in.

To illustrate with an example, if we have the following documents:

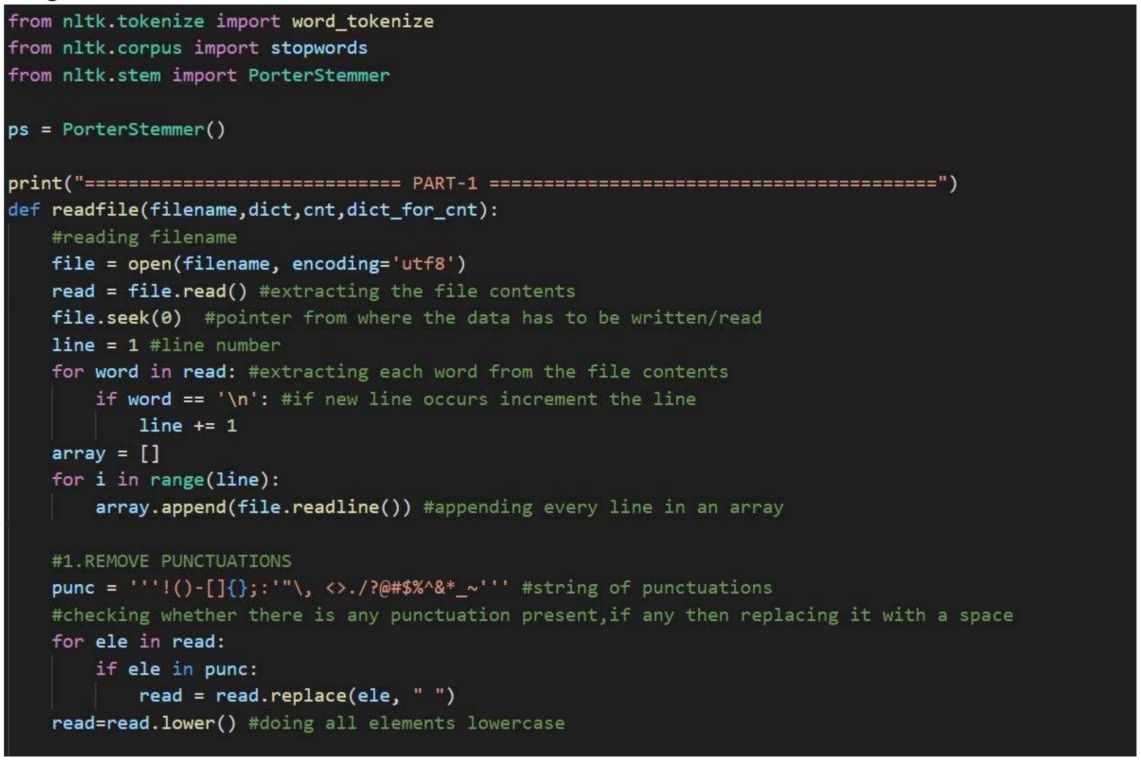
Document 1: Information Retrieval and Web Search

Document 2: Search Engine Ranking

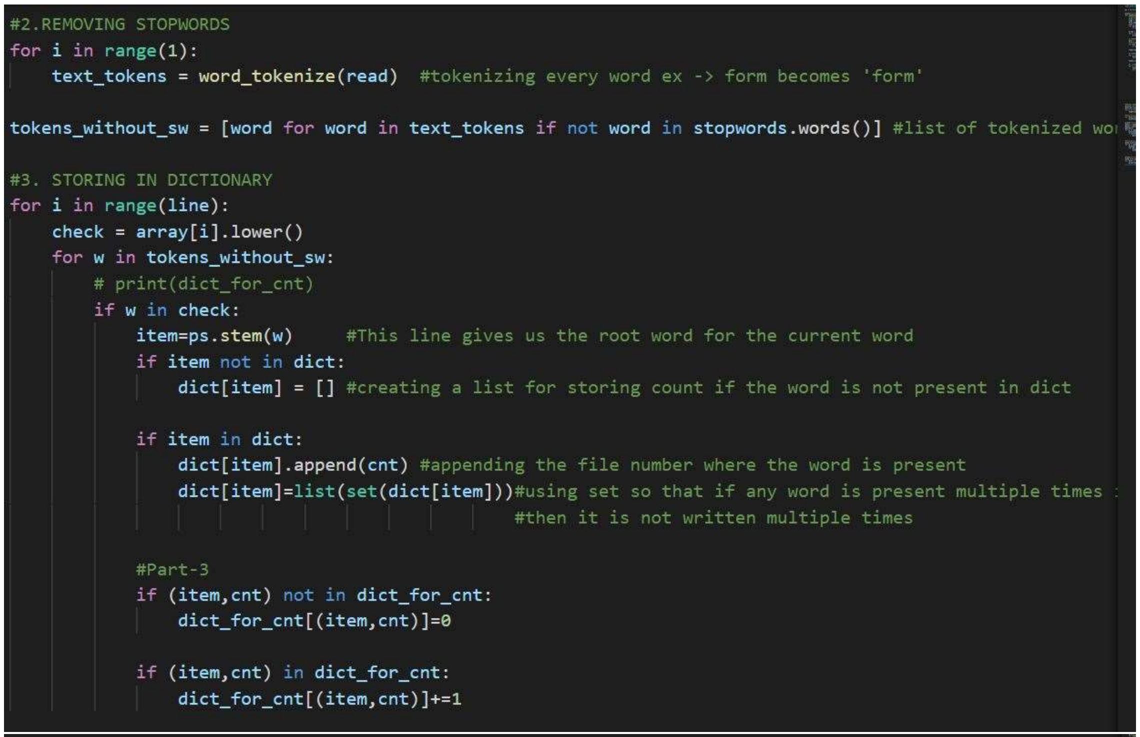
Document 3: Web Search Course

Then the postings list of the term ‘web’ would be the list [1, 3], meaning the term ‘web’ appears in documents with IDs 1 and 3. Similarly the postings list of the term ‘search’ would be [1, 2, 3], and for the term ‘course’ the postings list would be [3]. We may want to keep additional information in the index such as the number of occurrences of the term in the whole collection (its term frequency), or the number of different documents that the term appears in (its document frequency), the positions of the term’s occurrences within a document etc. The amount of information we keep in our index will grow as we add more functionality to our search engine.

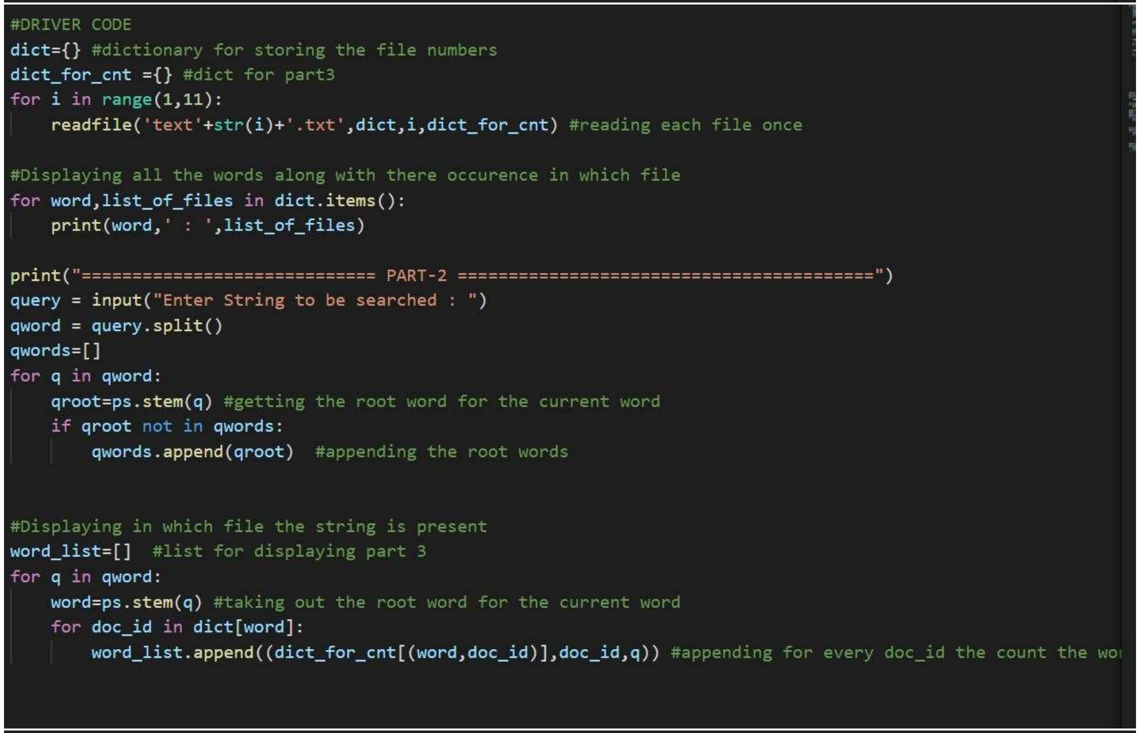
Program:



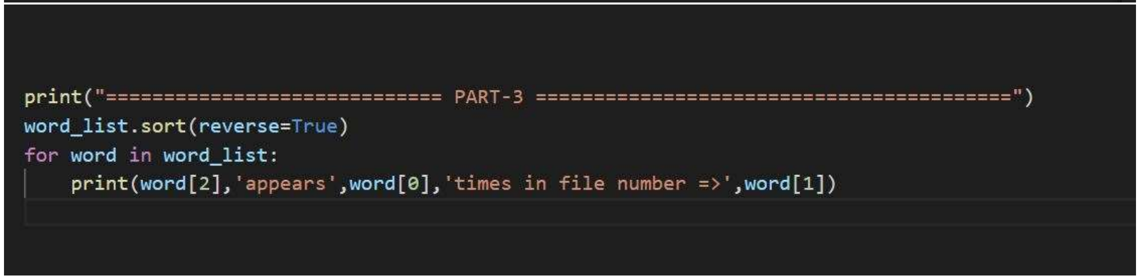
```python
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

ps = PorterStemmer()

print("============================ PART-1 ========================================")
def readfile(filename,dict,cnt,dict_for_cnt):
    #reading filename
    file = open(filename, encoding='utf8')
    read = file.read() #extracting the file contents
    file.seek(0)  #pointer from where the data has to be written/read
    line = 1 #line number
    for word in read: #extracting each word from the file contents
        if word == '\n': #if new line occurs increment the line
            line += 1
    array = []
    for i in range(line):
        array.append(file.readline()) #appending every line in an array

    #1.REMOVE PUNCTUATIONS
    punc = '''!()-[]{};:'"\, <>./?@#$%^&*_~''' #string of punctuations
    #checking whether there is any punctuation present,if any then replacing it with a space
    for ele in read:
        if ele in punc:
            read = read.replace(ele, " ")
    read=read.lower() #doing all elements lowercase
```



```
#2.REMOVING STOPWORDS
for i in range(1):
    text_tokens = word_tokenize(read)  #tokenizing every word ex -> form becomes 'form'

tokens_without_sw = [word for word in text_tokens if not word in stopwords.words()] #list of tokenized wor

#3. STORING IN DICTIONARY
for i in range(line):
    check = array[i].lower()
    for w in tokens_without_sw:
        # print(dict_for_cnt)
        if w in check:
            item=ps.stem(w)     #This line gives us the root word for the current word
            if item not in dict:
                dict[item] = [] #creating a list for storing count if the word is not present in dict

            if item in dict:
                dict[item].append(cnt) #appending the file number where the word is present
                dict[item]=list(set(dict[item]))#using set so that if any word is present multiple times
                                                #then it is not written multiple times

            #Part-3
            if (item,cnt) not in dict_for_cnt:
                dict_for_cnt[(item,cnt)]=0

            if (item,cnt) in dict_for_cnt:
                dict_for_cnt[(item,cnt)]+=1
```

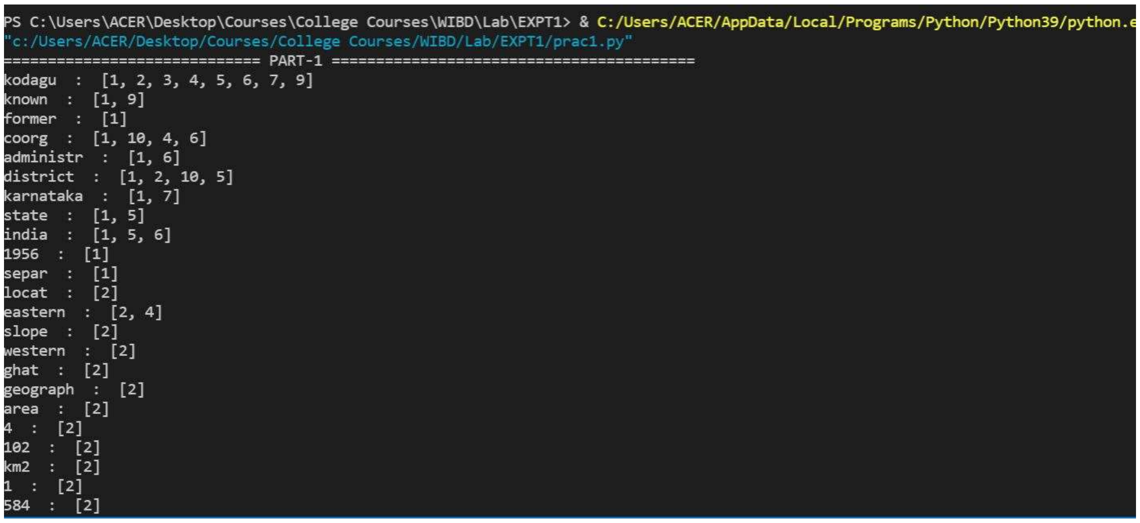


```
#DRIVER CODE
dict={} #dictionary for storing the file numbers
dict_for_cnt ={} #dict for part3
for i in range(1,11):
    readfile('text'+str(i)+'.txt',dict,i,dict_for_cnt) #reading each file once

#Displaying all the words along with there occurence in which file
for word,list_of_files in dict.items():
    print(word,' : ',list_of_files)

print("============================ PART-2 ========================================")
query = input("Enter String to be searched : ")
qword = query.split()
qwords=[]
for q in qword:
    qroot=ps.stem(q) #getting the root word for the current word
    if qroot not in qwords:
        qwords.append(qroot)  #appending the root words


#Displaying in which file the string is present
word_list=[]  #list for displaying part 3
for q in qword:
    word=ps.stem(q) #taking out the root word for the current word
    for doc_id in dict[word]:
        word_list.append((dict_for_cnt[(word,doc_id)],doc_id,q)) #appending for every doc_id the count the wor
```

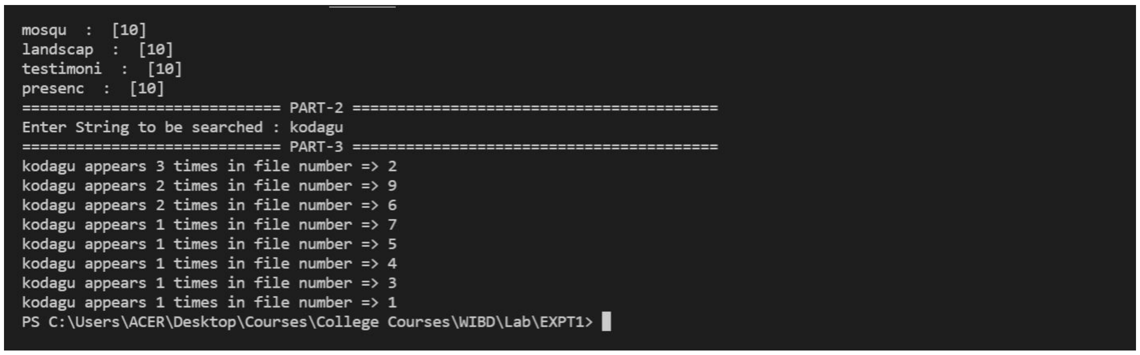


```
print("============================= PART-3 =========================================")
word_list.sort(reverse=True)
for word in word_list:
    print(word[2],'appears',word[0],'times in file number =>',word[1])
```

Output:



```
PS C:\Users\ACER\Desktop\Courses\College Courses\WIBD\Lab\EXPT1> & C:/Users/ACER/AppData/Local/Programs/Python/Python39/python.e
"c:/Users/ACER/Desktop/Courses/College Courses/WIBD/Lab/EXPT1/prac1.py"
============================ PART-1 ========================================
kodagu  :  [1, 2, 3, 4, 5, 6, 7, 9]
known  :  [1, 9]
former  :  [1]
coorg  :  [1, 10, 4, 6]
administr  :  [1, 6]
district  :  [1, 2, 10, 5]
karnataka  :  [1, 7]
state  :  [1, 5]
india  :  [1, 5, 6]
1956  :  [1]
separ  :  [1]
locat  :  [2]
eastern  :  [2, 4]
slope  :  [2]
western  :  [2]
ghat  :  [2]
geograph  :  [2]
area  :  [2]
4  :  [2]
102  :  [2]
km2  :  [2]
1  :  [2]
584  :  [2]
```



```
mosqu  :  [10]
landscap  :  [10]
testimoni  :  [10]
presenc  :  [10]
============================ PART-2 ========================================
Enter String to be searched : kodagu
============================ PART-3 ========================================
kodagu appears 3 times in file number => 2
kodagu appears 2 times in file number => 9
kodagu appears 2 times in file number => 6
kodagu appears 1 times in file number => 7
kodagu appears 1 times in file number => 5
kodagu appears 1 times in file number => 4
kodagu appears 1 times in file number => 3
kodagu appears 1 times in file number => 1
PS C:\Users\ACER\Desktop\Courses\College Courses\WIBD\Lab\EXPT1>
```

END