Interactive Search Engine

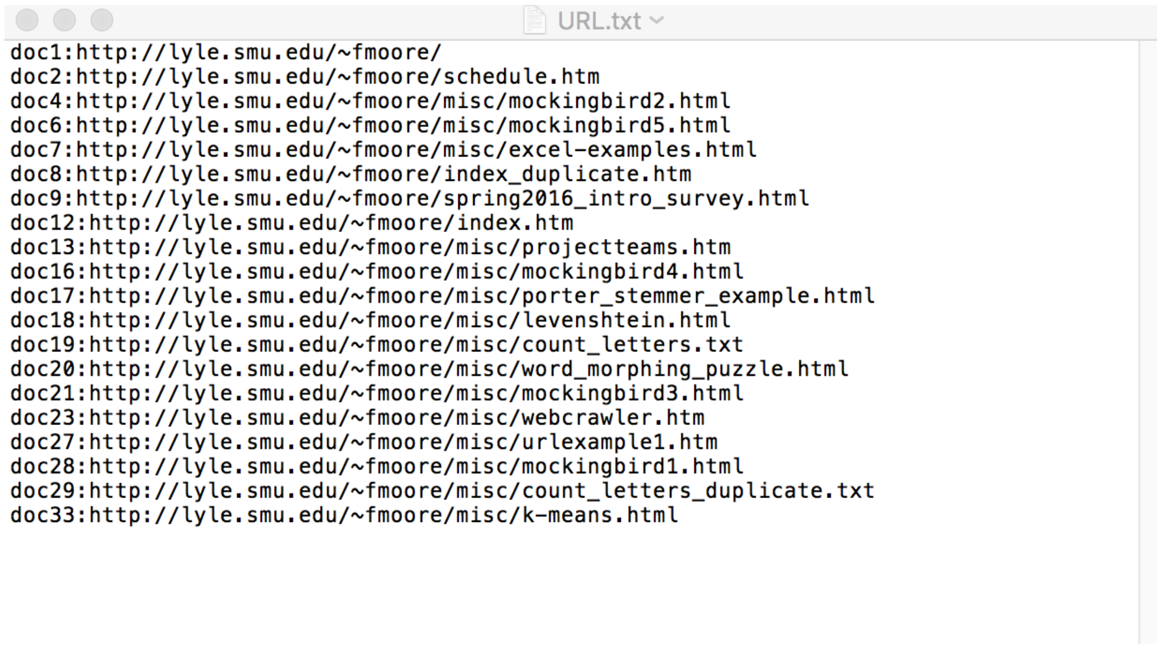
Ruixuan Zhang Xukai Wang

1. Use the web crawler you built in Project 1 that crawled a limited space, looking for text and html files. Describe in detail what aspects you modified to support the second half of the project. [15 points]

We have modified several parts of the crawler to satisfy our needs:

1) **Adding URL for each crawled documents, followed by its ID;**

   As we crawled all the pages and saved them as an txt document, we give each document a specific ID and adding its URL behind the ID. This is for further use of the follow question.



```
doc1:http://lyle.smu.edu/~fmoore/
doc2:http://lyle.smu.edu/~fmoore/schedule.htm
doc4:http://lyle.smu.edu/~fmoore/misc/mockingbird2.html
doc6:http://lyle.smu.edu/~fmoore/misc/mockingbird5.html
doc7:http://lyle.smu.edu/~fmoore/misc/excel-examples.html
doc8:http://lyle.smu.edu/~fmoore/index_duplicate.htm
doc9:http://lyle.smu.edu/~fmoore/spring2016_intro_survey.html
doc12:http://lyle.smu.edu/~fmoore/index.htm
doc13:http://lyle.smu.edu/~fmoore/misc/projectteams.htm
doc16:http://lyle.smu.edu/~fmoore/misc/mockingbird4.html
doc17:http://lyle.smu.edu/~fmoore/misc/porter_stemmer_example.html
doc18:http://lyle.smu.edu/~fmoore/misc/levenshtein.html
doc19:http://lyle.smu.edu/~fmoore/misc/count_letters.txt
doc20:http://lyle.smu.edu/~fmoore/misc/word_morphing_puzzle.html
doc21:http://lyle.smu.edu/~fmoore/misc/mockingbird3.html
doc23:http://lyle.smu.edu/~fmoore/misc/webcrawler.htm
doc27:http://lyle.smu.edu/~fmoore/misc/urlexample1.htm
doc28:http://lyle.smu.edu/~fmoore/misc/mockingbird1.html
doc29:http://lyle.smu.edu/~fmoore/misc/count_letters_duplicate.txt
doc33:http://lyle.smu.edu/~fmoore/misc/k-means.html
```

This file is generated automatically when we run the crawler. As you can see, it contains duplicate documents. We will remove duplicate ones in the following step.

2) **Creating a dictionary, an xml file, for all the words which are selected from the crawled documents;**

This time, our target is to query some words among all the documents, so we need a dictionary to store all the words. We get each word out and saved them as an xml file for further use. Part of the xml file is shown here.

As you can see, the xml file record two things for a word: the document ID and its position in the document. This will give us the data for calculating the term frequency of each word. The core code for this part is:

```java
package crawler.queryengine;

import java.io.File;
import java.io.IOException;

public class DictionaryBuilder {

    public static void dictionaryBuilder(String filepath){
        new DuplicateOperater().duplicateOperate(filepath+"/");
        int docTotal = 40;
        for(int i=1; i<=docTotal;i++){
            File docFile = new File(filepath+"/Pages/"+i+".txt");
            File xmlFile = new File(filepath+"/dictionary.xml") ;
            File frequencyFile = new File(filepath+"/TermFrequency.xml");
//          System.out.println(docFile.exists()&&xmlFile.exists()&&frequencyFile.exists());
            if(docFile.exists()&&!xmlFile.exists()&&!frequencyFile.exists()){
                DocToMapBuilder docToMapBuilder = new DocToMapBuilder();
                docToMapBuilder.CreatXMLFile(filepath+"/", i);
            }
            else if(docFile.exists()&&xmlFile.exists()&&frequencyFile.exists()){
                DocToMapBuilder docToMapBuilder = new DocToMapBuilder();
                docToMapBuilder.UpdataXMLFile(filepath+"/", i);
            }
        }

    }
}
```

```java
public class UpdataDictionary {
    public static void UpdateDictionary(Map<String,String> dictionary,String docID,String filepath){
        Element root = null;
        File dictionaryXML = new File(filepath+"/dictionary.xml");
        DocumentBuilder documentBuilder = null;
        DocumentBuilderFactory documentBuilderFactory = null;
        try{
            documentBuilderFactory = DocumentBuilderFactory.newInstance();
            documentBuilder = documentBuilderFactory.newDocumentBuilder();
            Document document = documentBuilder.parse(dictionaryXML);
            root = document.getDocumentElement();
            NodeList words = root.getChildNodes();
            for(int i =0; i<words.getLength();i++){
                Node word = words.item(i);
                if("word".equals(word.getNodeName())){
                    if(dictionary.containsKey(word.getAttributes().getNamedItem("name").getTextContent())){
                        String value = dictionary.get(word.getAttributes().getNamedItem("name").getTextContent());
                        Element newdoc = document.createElement(docID);
                        newdoc.setTextContent(value);
                        word.appendChild(newdoc);
                        dictionary.remove(word.getAttributes().getNamedItem("name").getTextContent());
                    }
                }
            }
            for(Map.Entry<String, String> entry : dictionary.entrySet()){
                Element word = document.createElement("word");
                word.setAttribute("name",entry.getKey());
                Element newdoc = document.createElement(docID);
                newdoc.setTextContent(entry.getValue());
                word.appendChild(newdoc);
                root.appendChild(word);
            }
            TransformerFactory transformerFactory = TransformerFactory.newInstance();
            Transformer transformer = transformerFactory.newTransformer();
            DOMSource source = new DOMSource(document);
            StreamResult result = new StreamResult(new File(filepath+"/dictionary.xml"));
            transformer.transform(source, result);
        }catch(Exception e){
            e.printStackTrace();
        }
    }

}
```

3) **Creating a function to remove duplicate document by comparing a pair of documents;**

   As question 3 part (a) asked, we will talk more specific on this topic in question 3.

4) **Creating a function to remove stop words when crawling;**

   As question 3 part (b) asked, we will talk more specific on this topic in question 3.

5) **Creating a function of counting term frequency which will help calculating the cosine similarity;**

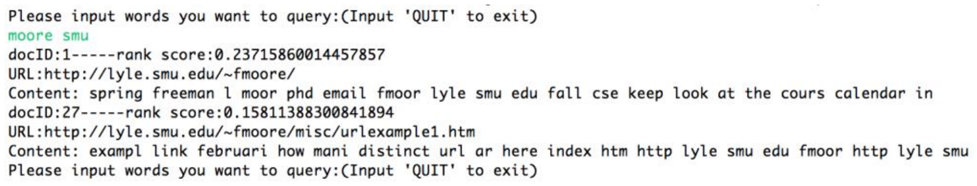
As we are going to rank the documents by its vector space scores, we use tf-idf as the weighting factors to implement the method. So we need to get the term frequency of each word and calculate the score for each word. We also save the data as an xml file, part of it is shown here.

   Again, the xml file record two things for one word: the document ID shows where it comes from; its score for the document. This will be the data set for further calculating.

   The core code for this part is:

6) **Designing a query engine which is used to query certain words among all the document;**

There should be somewhere we could enter our query and search it among all the saved documents. So we design an engine, it looks like this:



```
Please input words you want to query:(Input 'QUIT' to exit)
moore smu
docID:1-----rank score:0.23715860014457857
URL:http://lyle.smu.edu/~fmoore/
Content: spring freeman l moor phd email fmoor lyle smu edu fall cse keep look at the cours calendar in
docID:27-----rank score:0.15811388300841894
URL:http://lyle.smu.edu/~fmoore/misc/urlexample1.htm
Content: exampl link februari how mani distinct url ar here index htm http lyle smu edu fmoor http lyle smu
Please input words you want to query:(Input 'QUIT' to exit)
```

This is followed by the result of crawling. We could input the query after the reminder, and see the ranking result. It contains the document ID, relative URL, its score and the first 20 words of that document after stemmed.

The core code for this part is:

7) **Creating a function to order the document for certain query by its vector space score;**

As mentioned above, we need to ranking the documents. So we design a method to rank them by the vector space score. The result is the same as the picture above.

2. You will need a dictionary of words. [15 points]

   a) What is your definition of “word”?

   My idea of the definition of word is: a single distinct meaningful element of speech or writing, used with others (or sometimes alone) to form a sentence and typically shown with a space on either side when written or printed. This is shown in our work, such as the saved txt files, we use the character of space on either side to recognize the word.

   b) Was the dictionary generated while navigating the pages or as a separate step?

   We generate the dictionary as a separate step after crawling webs in our design.

We try to record the source of the word and its position in the document, this will help us in calculating the cosine similarity for question 5.

c) How many words are in the final dictionary?

If needed, you can define a fixed size for the dictionary based upon the results from Project1, and allowing for a 10% increase.

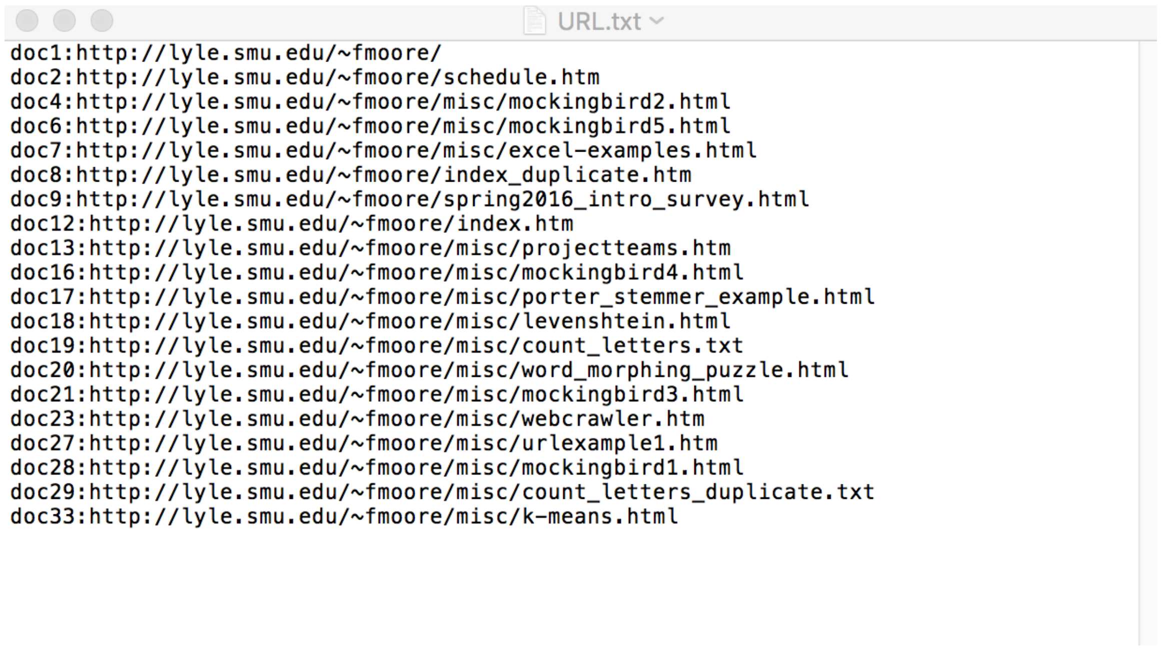
In our dictionary, there are 729 words in total. Dictionary does not contain the stop words.

3. For the purpose of this project, you may assume a maximum of 30 documents. You will need to create a term/document frequency matrix to support item 5. [20 points]

   a) remove duplicate documents from your collection of documents

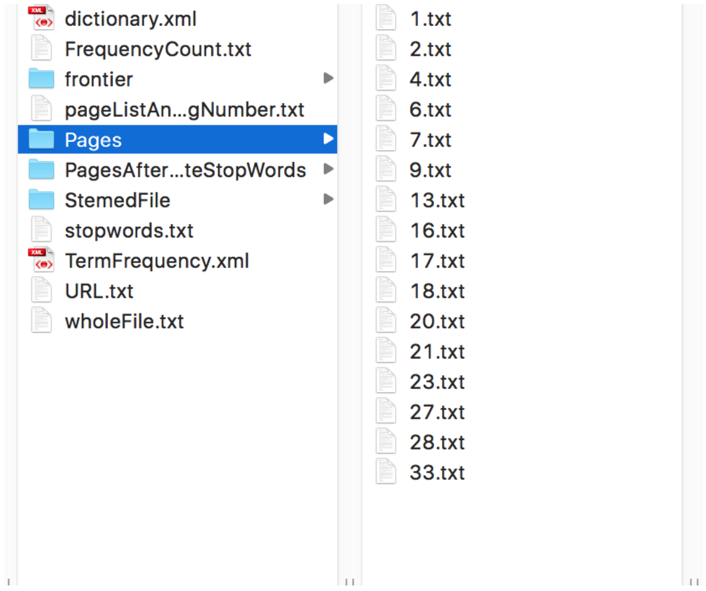
   As we need to remove the duplicate document, so we create a function to remove the totally duplicate document. We compare all the words of a pair documents which are at the same position, if all the words meet each other, then we think they are duplicated and remove one of it.

At first, we get the list like this:



```
doc1:http://lyle.smu.edu/~fmoore/
doc2:http://lyle.smu.edu/~fmoore/schedule.htm
doc4:http://lyle.smu.edu/~fmoore/misc/mockingbird2.html
doc6:http://lyle.smu.edu/~fmoore/misc/mockingbird5.html
doc7:http://lyle.smu.edu/~fmoore/misc/excel-examples.html
doc8:http://lyle.smu.edu/~fmoore/index_duplicate.htm
doc9:http://lyle.smu.edu/~fmoore/spring2016_intro_survey.html
doc12:http://lyle.smu.edu/~fmoore/index.htm
doc13:http://lyle.smu.edu/~fmoore/misc/projectteams.htm
doc16:http://lyle.smu.edu/~fmoore/misc/mockingbird4.html
doc17:http://lyle.smu.edu/~fmoore/misc/porter_stemmer_example.html
doc18:http://lyle.smu.edu/~fmoore/misc/levenshtein.html
doc19:http://lyle.smu.edu/~fmoore/misc/count_letters.txt
doc20:http://lyle.smu.edu/~fmoore/misc/word_morphing_puzzle.html
doc21:http://lyle.smu.edu/~fmoore/misc/mockingbird3.html
doc23:http://lyle.smu.edu/~fmoore/misc/webcrawler.htm
doc27:http://lyle.smu.edu/~fmoore/misc/urlexample1.htm
doc28:http://lyle.smu.edu/~fmoore/misc/mockingbird1.html
doc29:http://lyle.smu.edu/~fmoore/misc/count_letters_duplicate.txt
doc33:http://lyle.smu.edu/~fmoore/misc/k-means.html
```

However, when we run the program to remove the duplicate ones, we get the result like this:



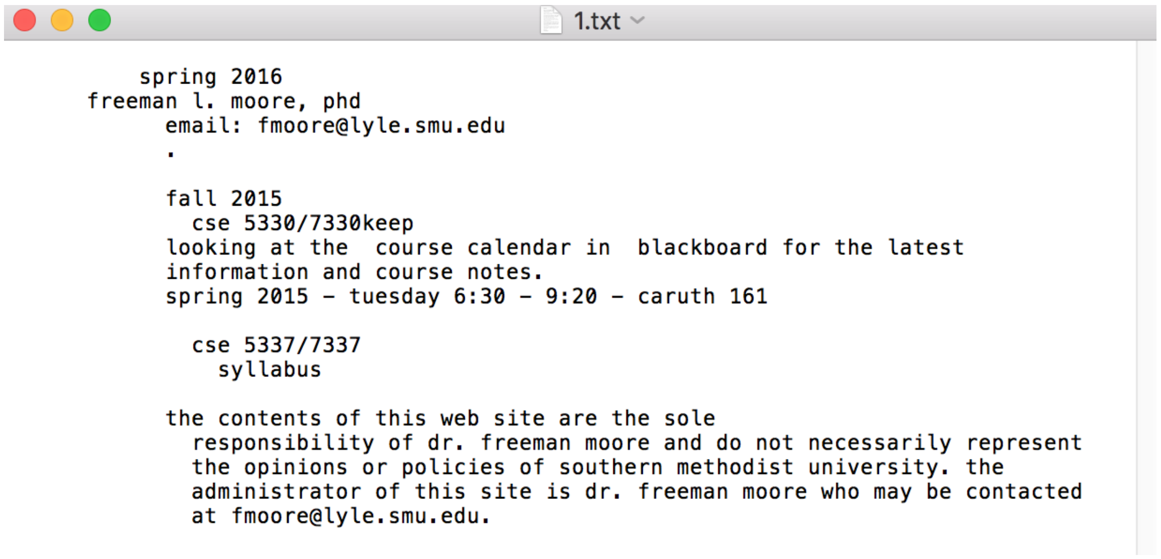
As you can see, we have removed doc8, doc12, doc19 and doc29, which are duplicate documents.

The core code for this part is:

b) remove stop words from documents.   A list will be provided.

As we are going to give a score for each document for the certain query, we should get rid of the stop words. So, once we saved the page as a txt file, then we remove the stop words. We change the word type into char, and set rules to manage them. Then we import the stop words list and compare each word with the stop words. Once found the same word, we delete it from the source file.

One example here, original txt:



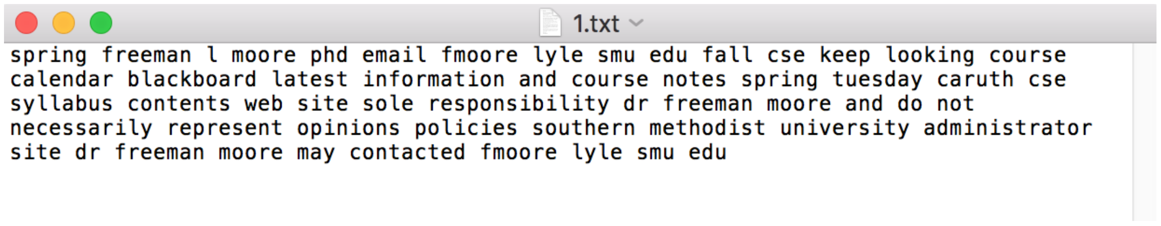
```
        spring 2016
    freeman l. moore, phd
          email: fmoore@lyle.smu.edu
          .

          fall 2015
            cse 5330/7330keep
          looking at the  course calendar in  blackboard for the latest
          information and course notes.
          spring 2015 - tuesday 6:30 - 9:20 - caruth 161

            cse 5337/7337
              syllabus

          the contents of this web site are the sole
            responsibility of dr. freeman moore and do not necessarily represent
            the opinions or policies of southern methodist university. the
            administrator of this site is dr. freeman moore who may be contacted
            at fmoore@lyle.smu.edu.
```

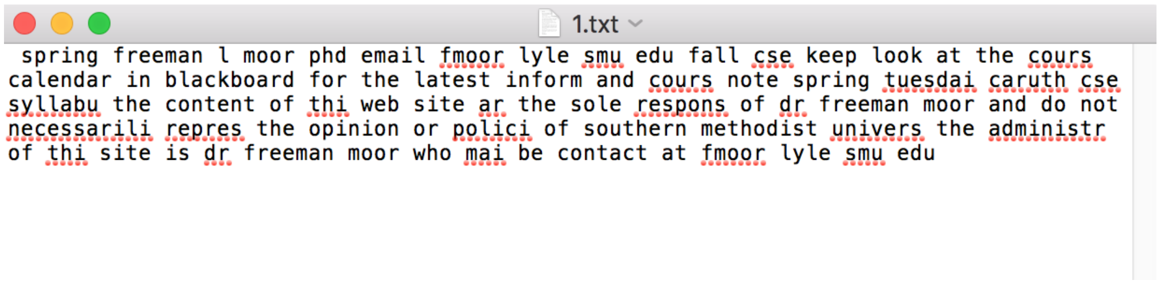
After removing stop words:



```
spring freeman l moore phd email fmoore lyle smu edu fall cse keep looking course
calendar blackboard latest information and course notes spring tuesday caruth cse
syllabus contents web site sole responsibility dr freeman moore and do not
necessarily represent opinions policies southern methodist university administrator
site dr freeman moore may contacted fmoore lyle smu edu
```

The core code for this part is:

Furthermore, we also stemmed the words:


1.txt

```
 spring freeman l moor phd email fmoor lyle smu edu fall cse keep look at the cours
calendar in blackboard for the latest inform and cours note spring tuesdai caruth cse
syllabu the content of thi web site ar the sole respons of dr freeman moor and do not
necessarili repres the opinion or polici of southern methodist univers the administr
of thi site is dr freeman moor who mai be contact at fmoor lyle smu edu
```

4. The user will be able to enter multiple queries, consisting of one or more query words separated by space on a line. Interaction ends when the query consists of the single word Quit. [10 points]

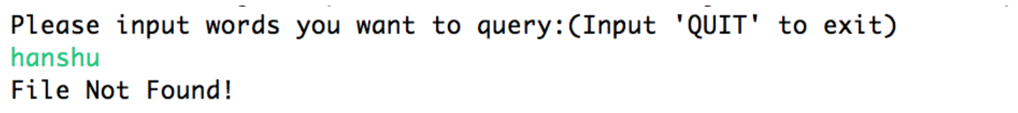
   a) What happens if a query contains a word that is not in the dictionary? Case sensitive?

   At first, I want to show the function of ending the interaction by input word QUIT in the query engine:

```
Please input words you want to query:(Input 'QUIT' to exit)
QUIT
Finished!
```

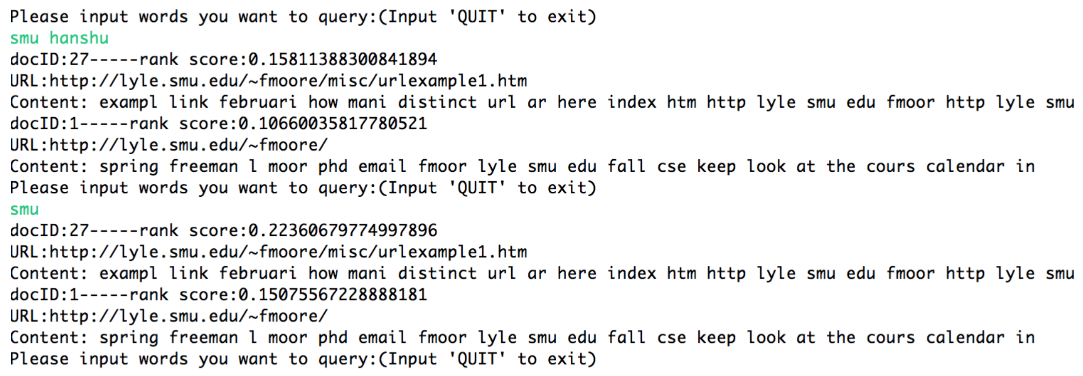
   As we changed the query words into lowercase, so no matter you input the “QUIT” or “quit” or any type, the engine will recognize the word and end the interaction. If you want to run it again, you should run the whole system again.

If the query only contains a word not include within the dictionary, then the engine will remind that no file was found for that query. One example is shown here:



```
Please input words you want to query:(Input 'QUIT' to exit)
hanshu
File Not Found!
```

However, if the query just contains some words not in the dictionary, it will show the result of ranking. But the score will change, one example is shown here:

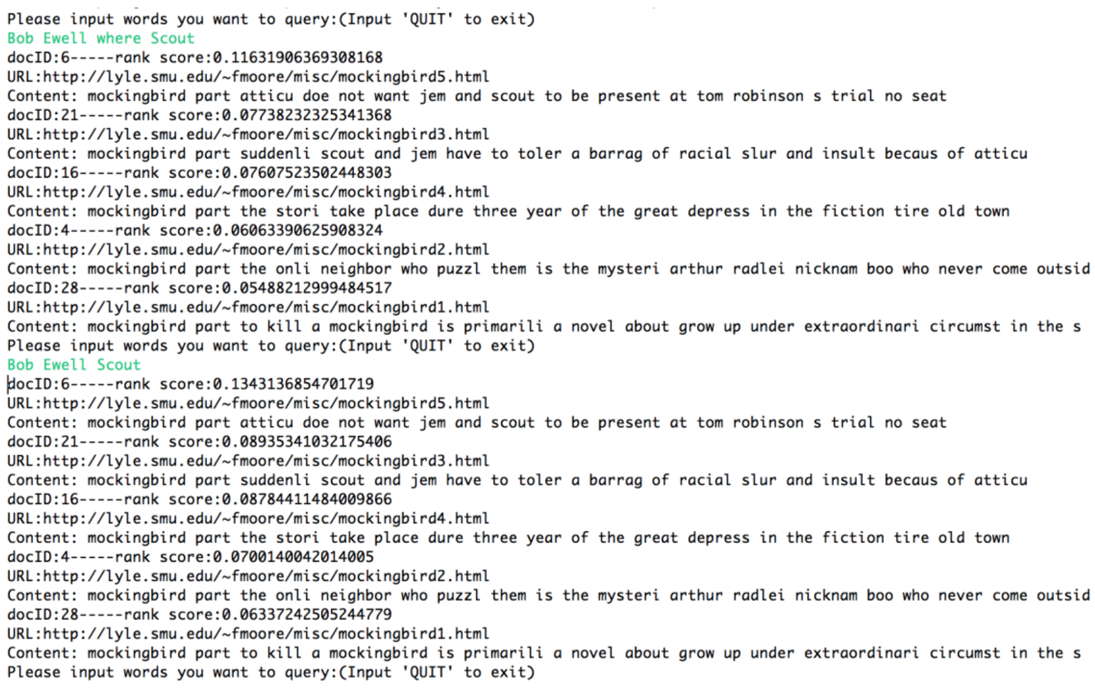


```
Please input words you want to query:(Input 'QUIT' to exit)
smu hanshu
docID:27-----rank score:0.15811388300841894
URL:http://lyle.smu.edu/~fmoore/misc/urlexample1.htm
Content: exampl link februari how mani distinct url ar here index htm http lyle smu edu fmoor http lyle smu
docID:1-----rank score:0.10660035817780521
URL:http://lyle.smu.edu/~fmoore/
Content: spring freeman l moor phd email fmoor lyle smu edu fall cse keep look at the cours calendar in
Please input words you want to query:(Input 'QUIT' to exit)
smu
docID:27-----rank score:0.22360679774997896
URL:http://lyle.smu.edu/~fmoore/misc/urlexample1.htm
Content: exampl link februari how mani distinct url ar here index htm http lyle smu edu fmoor http lyle smu
docID:1-----rank score:0.15075567228888181
URL:http://lyle.smu.edu/~fmoore/
Content: spring freeman l moor phd email fmoor lyle smu edu fall cse keep look at the cours calendar in
Please input words you want to query:(Input 'QUIT' to exit)
```

As you can see, the first query contains the word “hanshu” which is not a word in the dictionary. But the engine also feedbacks the ranking for the whole query. Once we delete the word “hanshu”, a new result is shown. The difference between two results is that the score gets changed. That means the unknown word in a query will reduce the score for the same dictionary. Whereas, as long as there is one word which is included within the dictionary, there will be a result to feedback.

b) What happens if a query contains a stop word? A set of a queries will be provided.

Let’s see the result of an example at first.



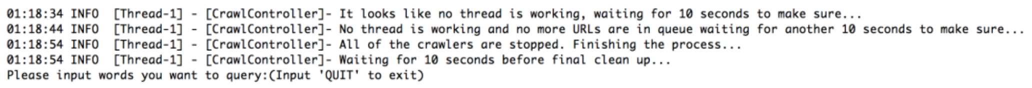
```
Please input words you want to query:(Input 'QUIT' to exit)
Bob Ewell where Scout
docID:6-----rank score:0.11631906369308168
URL:http://lyle.smu.edu/~fmoore/misc/mockingbird5.html
Content: mockingbird part atticu doe not want jem and scout to be present at tom robinson s trial no seat
docID:21-----rank score:0.07738232325341368
URL:http://lyle.smu.edu/~fmoore/misc/mockingbird3.html
Content: mockingbird part suddenli scout and jem have to toler a barrag of racial slur and insult becaus of atticu
docID:16-----rank score:0.07607523502448303
URL:http://lyle.smu.edu/~fmoore/misc/mockingbird4.html
Content: mockingbird part the stori take place dure three year of the great depress in the fiction tire old town
docID:4-----rank score:0.06063390625908324
URL:http://lyle.smu.edu/~fmoore/misc/mockingbird2.html
Content: mockingbird part the onli neighbor who puzzl them is the mysteri arthur radlei nicknam boo who never come outsid
docID:28-----rank score:0.05488212999484517
URL:http://lyle.smu.edu/~fmoore/misc/mockingbird1.html
Content: mockingbird part to kill a mockingbird is primarili a novel about grow up under extraordinari circumst in the s
Please input words you want to query:(Input 'QUIT' to exit)
Bob Ewell Scout
docID:6-----rank score:0.1343136854701719
URL:http://lyle.smu.edu/~fmoore/misc/mockingbird5.html
Content: mockingbird part atticu doe not want jem and scout to be present at tom robinson s trial no seat
docID:21-----rank score:0.08935341032175406
URL:http://lyle.smu.edu/~fmoore/misc/mockingbird3.html
Content: mockingbird part suddenli scout and jem have to toler a barrag of racial slur and insult becaus of atticu
docID:16-----rank score:0.08784411484009866
URL:http://lyle.smu.edu/~fmoore/misc/mockingbird4.html
Content: mockingbird part the stori take place dure three year of the great depress in the fiction tire old town
docID:4-----rank score:0.0700140042014005
URL:http://lyle.smu.edu/~fmoore/misc/mockingbird2.html
Content: mockingbird part the onli neighbor who puzzl them is the mysteri arthur radlei nicknam boo who never come outsid
docID:28-----rank score:0.06337242505244779
URL:http://lyle.smu.edu/~fmoore/misc/mockingbird1.html
Content: mockingbird part to kill a mockingbird is primarili a novel about grow up under extraordinari circumst in the s
Please input words you want to query:(Input 'QUIT' to exit)
```

Here, we use two queries “Bob Ewell where Scout” and “Bob Ewell Scout”. The former one contains the stop word “where”, the second one gets rid of it. The difference between two results also lie in the score. The reason for this phenomenon is the algorithm. Although the stop words are deleted from the txt file, but when they come into the query, they will occupy a place in the query vector. This will influence the value for non-stopwords words when calculating the unit value. And it will of course reduce the score which could be seen above. So, the stop words will not influence the order of the ranking, but the score will be lower if a query contains stop words.

5. Implement the cosine similarity of the query against all documents. [40 points]

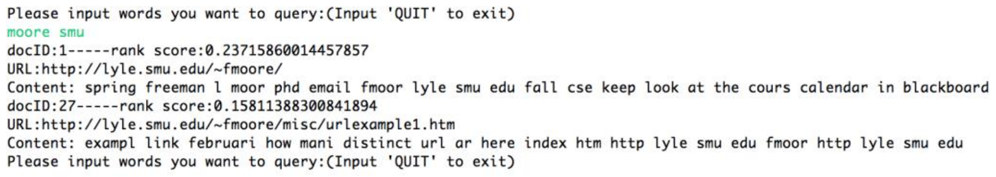
   a) Display the similarity measure and document URL in descending numerical order for the top N results.

   After the program crawled the webs, it will stop and kill the no more use thread to save memory. Then, there will be a reminder to tell we could start querying.

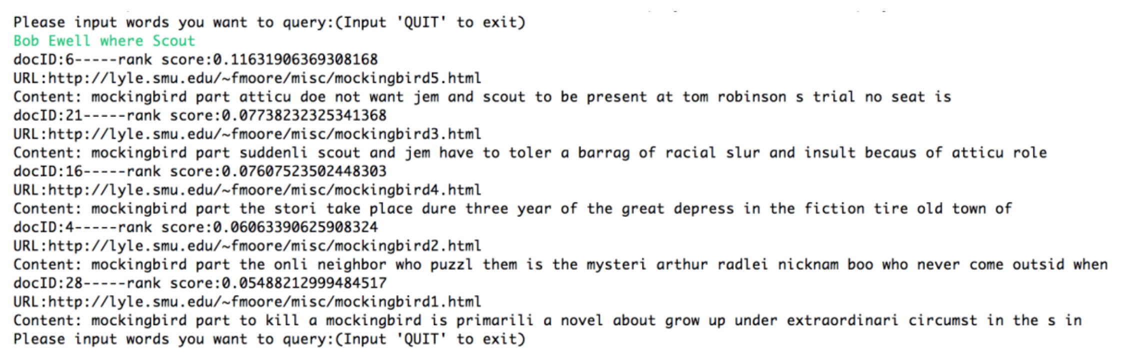


```
01:18:34 INFO  [Thread-1] - [CrawlController]- It looks like no thread is working, waiting for 10 seconds to make sure...
01:18:44 INFO  [Thread-1] - [CrawlController]- No thread is working and no more URLs are in queue waiting for another 10 seconds to make sure...
01:18:54 INFO  [Thread-1] - [CrawlController]- All of the crawlers are stopped. Finishing the process...
01:18:54 INFO  [Thread-1] - [CrawlController]- Waiting for 10 seconds before final clean up...
Please input words you want to query:(Input 'QUIT' to exit)
```

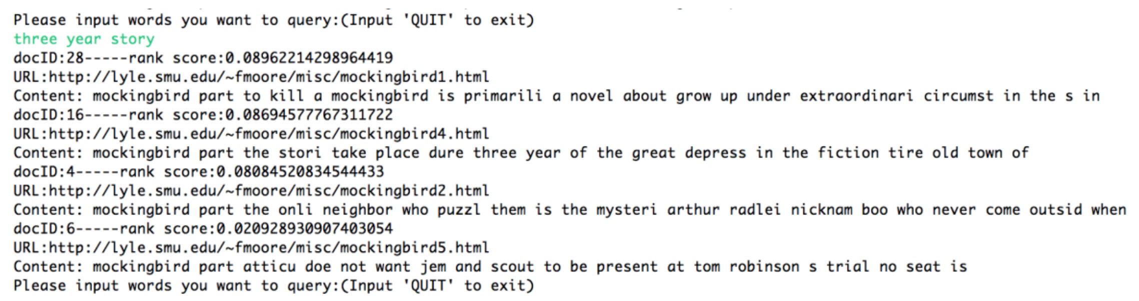
   The whole results for the provided 5 queries are:



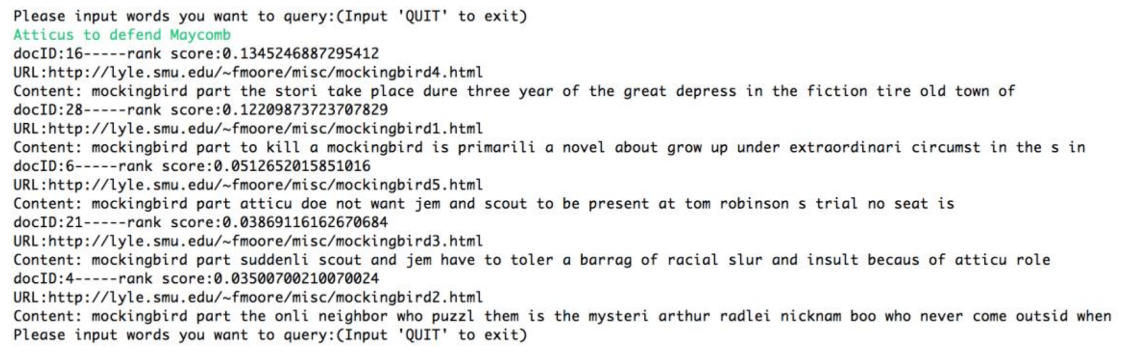
```
Please input words you want to query:(Input 'QUIT' to exit)
moore smu
docID:1-----rank score:0.23715860014457857
URL:http://lyle.smu.edu/~fmoore/
Content: spring freeman l moor phd email fmoor lyle smu edu fall cse keep look at the cours calendar in blackboard
docID:27-----rank score:0.15811388300841894
URL:http://lyle.smu.edu/~fmoore/misc/urlexample1.htm
Content: exampl link februari how mani distinct url ar here index htm http lyle smu edu fmoor http lyle smu edu
Please input words you want to query:(Input 'QUIT' to exit)
```



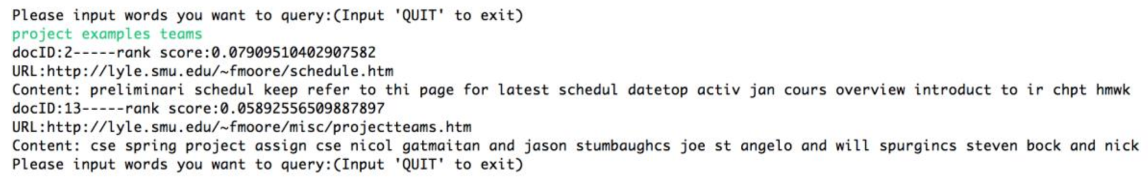
```
Please input words you want to query:(Input 'QUIT' to exit)
Bob Ewell where Scout
docID:6-----rank score:0.11631906369308168
URL:http://lyle.smu.edu/~fmoore/misc/mockingbird5.html
Content: mockingbird part atticu doe not want jem and scout to be present at tom robinson s trial no seat is
docID:21-----rank score:0.07738232325341368
URL:http://lyle.smu.edu/~fmoore/misc/mockingbird3.html
Content: mockingbird part suddenli scout and jem have to toler a barrag of racial slur and insult becaus of atticu role
docID:16-----rank score:0.07607523502448303
URL:http://lyle.smu.edu/~fmoore/misc/mockingbird4.html
Content: mockingbird part the stori take place dure three year of the great depress in the fiction tire old town of
docID:4-----rank score:0.06063390625908324
URL:http://lyle.smu.edu/~fmoore/misc/mockingbird2.html
Content: mockingbird part the onli neighbor who puzzl them is the mysteri arthur radlei nicknam boo who never come outsid when
docID:28-----rank score:0.05488212999484517
URL:http://lyle.smu.edu/~fmoore/misc/mockingbird1.html
Content: mockingbird part to kill a mockingbird is primarili a novel about grow up under extraordinari circumst in the s in
Please input words you want to query:(Input 'QUIT' to exit)
```



```
Please input words you want to query:(Input 'QUIT' to exit)
three year story
docID:28-----rank score:0.08962214298964419
URL:http://lyle.smu.edu/~fmoore/misc/mockingbird1.html
Content: mockingbird part to kill a mockingbird is primarili a novel about grow up under extraordinari circumst in the s in
docID:16-----rank score:0.08694577767311722
URL:http://lyle.smu.edu/~fmoore/misc/mockingbird4.html
Content: mockingbird part the stori take place dure three year of the great depress in the fiction tire old town of
docID:4-----rank score:0.08084520834544433
URL:http://lyle.smu.edu/~fmoore/misc/mockingbird2.html
Content: mockingbird part the onli neighbor who puzzl them is the mysteri arthur radlei nicknam boo who never come outsid when
docID:6-----rank score:0.020928930907403054
URL:http://lyle.smu.edu/~fmoore/misc/mockingbird5.html
Content: mockingbird part atticu doe not want jem and scout to be present at tom robinson s trial no seat is
Please input words you want to query:(Input 'QUIT' to exit)
```



```
Please input words you want to query:(Input 'QUIT' to exit)
Atticus to defend Maycomb
docID:16-----rank score:0.1345246887295412
URL:http://lyle.smu.edu/~fmoore/misc/mockingbird4.html
Content: mockingbird part the stori take place dure three year of the great depress in the fiction tire old town of
docID:28-----rank score:0.12209873723707829
URL:http://lyle.smu.edu/~fmoore/misc/mockingbird1.html
Content: mockingbird part to kill a mockingbird is primarili a novel about grow up under extraordinari circumst in the s in
docID:6-----rank score:0.0512652015851016
URL:http://lyle.smu.edu/~fmoore/misc/mockingbird5.html
Content: mockingbird part atticu doe not want jem and scout to be present at tom robinson s trial no seat is
docID:21-----rank score:0.03869116162670684
URL:http://lyle.smu.edu/~fmoore/misc/mockingbird3.html
Content: mockingbird part suddenli scout and jem have to toler a barrag of racial slur and insult becaus of atticu role
docID:4-----rank score:0.03500700210070024
URL:http://lyle.smu.edu/~fmoore/misc/mockingbird2.html
Content: mockingbird part the onli neighbor who puzzl them is the mysteri arthur radlei nicknam boo who never come outsid when
Please input words you want to query:(Input 'QUIT' to exit)
```



```
Please input words you want to query:(Input 'QUIT' to exit)
project examples teams
docID:2-----rank score:0.07909510402907582
URL:http://lyle.smu.edu/~fmoore/schedule.htm
Content: preliminari schedul keep refer to thi page for latest schedul datetop activ jan cours overview introduct to ir chpt hmwk
docID:13-----rank score:0.05892556509887897
URL:http://lyle.smu.edu/~fmoore/misc/projectteams.htm
Content: cse spring project assign cse nicol gatmaitan and jason stumbaughcs joe st angelo and will spurgincs steven bock and nick
Please input words you want to query:(Input 'QUIT' to exit)
```

Here, we change all the query words into lowercase which could ignore the difference between the capital and lower-case in the query. The result ranking all the relative documents by its vector space score.

For each result, it shows the document ID, URL, score and the first 20 stemmed words.

b) What weighting factors did you implement?

We use the tf-idf as the weighting factor when implementing the cosine similarity algorithm.

c) What value of N did you pick? Why? Obviously, N << 30.

Actually, we should stop if score relatively unchanged. However, in this project, there are too few documents. If we really want to choose a N for ranking in this project, we think the number of 2 may be a good choice. We have two reasons: to some queries, the score seems to be unchanged from the second one; another reason is there are only two results for some queries. So, we did not set a N for the ranking, but we think the number of 2 may be best adjusted to this project.