**INFORMATION RETRIVEL AND WEB SEARCH**

**INFS 7410**

ROSHAN BAGLA

S4447418

Master of Computer Science (Management)

**THE UNIVERSITY OF QUEENSLAND**

1. Brief description for your source code and any integrated libraries.

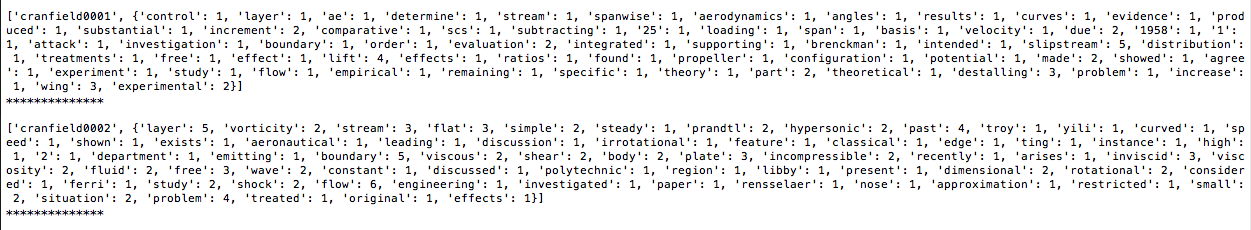
Firstly, my code is written in python language. I’ve used this language because I am very comfortable in it and I have good command over it. The libraries that I have used are elucidated below:

* Beautiful Soup: This library is used to extract the data from the HTML or other tags.
* NLTK (Natural language toolkit): This library helps in tokenization of the sentences.
* Counter: Counter library provides the frequency of each word in a document or find the count of how many times a particular word is appearing in a document.

My source code is divided into 2 parts. In the first part, I have worked on creating an inverted index and in second part; I have worked on building search engine. Later part of the code contains around 120 lines of code. Moreover, I have used **Document at a time approach** as discussed in the lecture slides.

**Code explanation:**

Firstly, after tokenization and removing the special or stop words, I focused on doing the mapping of each document with the words in it. Using the Counter function I have calculated the frequency of each word in each document and mapped it with the document number.

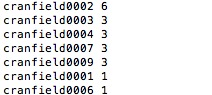


For example 1, [‘cranfield0001’, {‘control’: 1, ‘layer’: 1, ‘ae’ :1, ………}]

For example 2, [‘cransfield0002’, {‘layer’ : 5, ‘vorticity’: 2, ‘stream’: 3 ………….}]

* After mapping, I have worked on the search engine part i.e. finding the top ranked documents that have the high number

After giving input (could be one keyword or more keywords) I have calculated the frequencies and stored them in a list as shown below.



1. Plotting of term frequency distribution histogram sorted by frequency, and lists the top 10 most frequent and 10 least frequent terms with their frequencies.

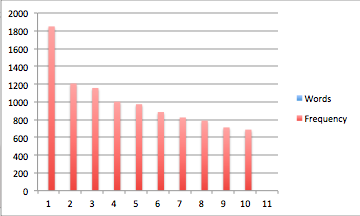


Figure 1 Top 10 frequent words

**The list of top 10 frequent words:**

|  |  |
| --- | --- |
| Words | Frequency |
| flow | 1849 |
| pressure | 1207 |
| boundary | 1156 |
| layer | 1002 |
| number | 973 |
| results | 885 |
| mach | 824 |
| theory | 789 |
| shock | 712 |
| method | 687 |
|  |  |

**Histogram for top 10 least frequent words**

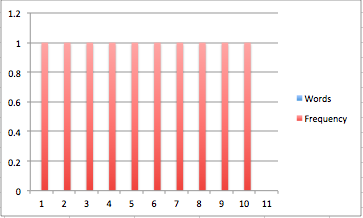
****

Figure 2 Top 10 least frequent words

|  |  |
| --- | --- |
| Words | Frequency |
| expands | 1 |
| understand | 1 |
| xenon | 1 |
| intentionally | 1 |
| balsa | 1 |
| adaptable | 1 |
| concur | 1 |
| sustains | 1 |
| lance | 1 |
| contacting | 1 |
|  |  |
|  |  |

1. Reporting the query, query results (top 10 documents with sum of frequencies>0), and average query execution time over 10 executions of the same query, for 3 one-keyword queries, 3 two-keyword queries and 3 three-keyword queries of your choice such that these three queries return no results, a few results and many results.

The results of executing the **3 one-keyword** queries are as follows:

* First query is **‘flow’** and the average execution time for this is 0.00531101226807 seconds. The results are:

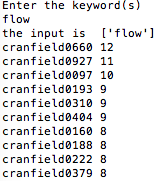
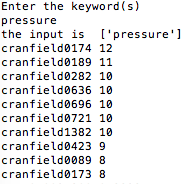


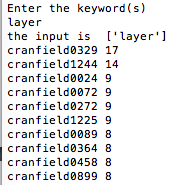
Figure 3 The entered keyword was flow

* Second query is **‘pressure’** and the average execution time for this is 0.00319004058838. The results of the query are



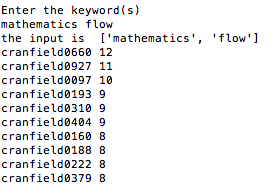
* Third query is **‘layer’** and the average execution time is 0.00407886505127 sec.

The results are displayed below

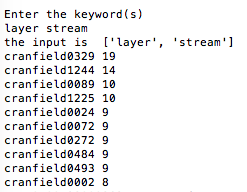


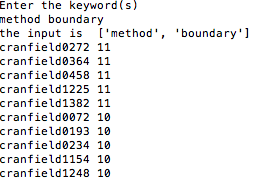
The results of executing the **3 two-keyword** queries are as follows:

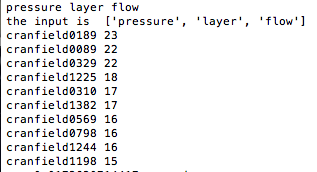
* The first two-keyword query is **‘ mathematics flow’** and the average running time is 0.00733017921448 seconds. The results are shown below



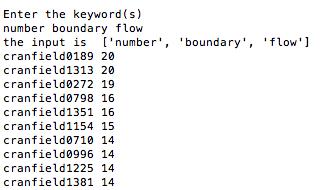
* The second two-keywords query is **‘ layer stream’** and the average execution time is 0.00738000869751 sec.



* The third two-keyword query is **‘method boundary’** and the average execution time is 0.00765800476074 seconds. 
* The first three-keyword query is ‘**pressure layer flow’** and the average execution time is 0.0173630714417 seconds.



* The second three-keyword query is **‘number boundary flow’** and the average execution time is 0.0090639591217 seconds.



* The third three-keyword query is ‘**number xenon lance’** and the average execution time is 0.0222690105438 seconds.