Document Search Engine

Mitansh Jain - 160101042
Sujoy Ghosh - 160101073
Akul Agrawal - 160101085
System Programming Lab

March 17, 2018

1 Salient Features

- Traditional indexing and searching algorithm for answering any query.
- Option for both searches phrase query and free text query.
- Printing lines in which query is present and highlighting words from query.
- Document ranking with respect to the query
- Spell check and probable suggestions for the query

2 Indexing

2.1 Data Structures and Algorithm

First, the program goes through all the text files in directory. All the words are stemmed using nltk.stemmer(). Now three dictionaries are maintained:

wordlist : This dictionary stores the number of occurrences of a word in a file.

wordloclist : This dictionary stores all the locations of a word in a file. The location includes the line number and position of the word in that line.

doclist : This dictionary stores the number of words in a document corresponding to its name.

These files are converted into binary files using the pickle library. This is done to prevent time wastage on re-indexing of already indexed files. Thus, on each run time of the program, only those files are indexed which are new and have not been indexed before. Binary files have been used to increase efficiency.

2.2 Time Complexity

All the files are opened using open() function in python. Time taken in indexing can be given in following way:

Time = O((number of files)*(number of lines)*(number of words in lines)*(time taken in stemming one word)*(access time of python list))
= O((total sum of number of words in all files)*(time taken in stemming one word)*(access time))

Time taken in stemming each word using porter stemming algorithm = O(7) = constant

Access time of python dictionary by amortized analysis = O(1)

Net time complexity = O(total number of words in all docs)

3 Searching and Ranking

3.1 Data Structures and Algorithms

For finding phrases' lists wordloclist created earlier is used for each word and intersection is taken to see at what all places all words occur together. Then all the other partial phrases are considered.
Ranking of documents is done using following Okapi BM25 formula:

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

$$\text{IDF}(q_i) = \log \frac{N}{n}$$

where,

D is document to be ranked

Q is query respect to which document is ranked

q_i represents i$^{\text{th}}$ word of query.

k$_1$ is constant = 1.2

b is constant = 0.75

3.2 Time complexity

Calculating time complexity has two parts first for answering query and then ranking documents in which query i present.

Average time complexity for query $=$ O(Ql) + $O(Sd*$ number of times in query) + O((log(average doc length))$*Sd$)+$O(Sd*Ql*$average document length+ O(summation of lines throughout all docs in which query is present))
$\approx O(Sd*$ average document length)

Time complexity for ranking $= O(Sd)$

where,

Sd is sum of number of documents in which query is present

Ql is length of query