CWID: A20327606

CS429: Assignment 1

In order to execute the assignment file, below are the steps to follow:

1. Change directory to the folder where the index.py and collection folder are present. (Assuming the collection.zip is unzipped and extracted to a folder “collection”)

2. Run a python command: *python (please use python 3)*

3. Execute the index.py using following command:
   exec(open(“path/index.py”.read())
   For ex: exec(open("/home/sheetal/Information_Retrieval/index.py").read())

4. *a=index('/home/sheetal/Information_Retrieval/collection/')*
   In above command create an object of index class by giving path of collection folder

5. *x=a.and_query(['with', 'without', 'yemen'])*

6. *x=a.print_dict()*

7. *x=a.print_doc_list()*

Algorithm and Analysis:

I have used two different approaches to solve this assignment.

Method 1: and_query
Params:
Query tems list with number of terms to be found in the documents together

1. Create an empty list (*rel_docs*) equivalent to size of the number of documents in collection folder. The index of list denoted the document ID.
2. Get the doc IDs from posting lists for all the terms in query_terms.
3. Iterate once through all the docIDs of all the terms
4. For each document ID found in the above list of lists, increment the value present in corresponding index (equal to document ID) of rel_docs list created in step 1.
5. Iterate through the the rel_docs and find the index with value equal to the length of query_terms (i.e the document Id which is present in posting list of all the query terms)

Method 2: and_query_1

Params:

Query tems list with number of terms to be found in the documents together.

1. Get the doc IDs from posting lists for all the terms in query_terms and append it to a list, thus forming a list of lists.
2. Sort the above list of lists according to the length of sublists.
3. Pick first 2 lists and apply merge algorithm on them.
4. Merge algorithm: using 2 counters one for each list, if the values are same, append it to the result, else increment the value for the list which has smaller value.
5. If the result of above step is empty, break.
6. Else, after getting result from above merge algorithm, call the merge algorithm again by passing the result and third list.
7. Continue above step for all remaining lists
8. Print the document names for the values in above result.

Analysis and Conclusion:

Both the above algorithms run in time O(n+m), where n is the length of query_terms (i.e. number of given queries to AND) and m is the maximum length among the posting lists of all queries.

However, I noticed that first method runs faster than second since it only increments the counter in the array for a given document ID. Notice the time difference below:

```
>>> x=a.and_query_1(['nuclear', 'power', 'country','war','communist'])
Results for the Query:  nuclear AND power AND country AND war AND communist
Total Docs retrieved:  5
Text-47.txt
Text-315.txt
Text-343.txt
Text-290.txt
Text-271.txt
Retrieved in  0.0015759468078613281  seconds
>>> x=a.and_query(['nuclear', 'power', 'country','war','communist'])
Results for the Query:  nuclear AND power AND country AND war AND communist
Total Docs retrieved:  5
Text-47.txt
Text-315.txt
Text-343.txt
Text-290.txt
Text-271.txt
Retrieved in  0.0009610652923583984  seconds
```