CS 429 Information Retrieval

Assignment 1: Boolean Retrieval

Points: 50

Submission deadline: Friday, 02/03/17, 11.59 PM

Note: This is individual work.

In this assignment, you will implement an inverted index in Python that can be used with Boolean queries with AND operators to retrieve relevant documents. You will implement the function in the code stub provided (see attachment). For your implementation, note the following design conditions:

1. Use the code stub provided as an attachment and implement the functions in the stub. You may add additional functions as you see fit.
2. Use only the python modules used in the stub.
3. Use the document collection provided (see attachment) to generate the index.
4. Tokenize the words in each document. There is no need to normalize the tokens. Do the basic tokenization i.e., remove all punctuations, and numerals. Terms in the index should all be in small caps.
5. Each document should be given a unique document id.
6. The posting list, in addition to the document id, should also contain the positions of the term in the document. The index should be in the form ***term: [(ID1,[pos1,pos2,..]), (ID2, [pos1,pos2,…]),….]***
7. Implement a merge algorithm for processing AND Boolean queries. (see additional guidelines below)
8. Print out the time for building the index and processing the query.
9. Provide appropriate comments in code.

Merge Algorithm: (see AND_query() in code stub)

- Implement an efficient algorithm to merge (AND operation) posting lists of multiple query terms.
- The algorithm should take a list of query terms (can be more than 2 terms) as input.
- In class, we looked at an algorithm for merging posting lists of two query terms by moving a pointer over the posting lists simultaneously. Generalize this for n terms. Note that an efficient algorithm looks at each item in the posting lists of the query term only one.

Sample Output:

```
>>> execfile('/home/jkorah/mnt/ir/Assignments/Assignment1/code/index.py')
>>> a=index('/home/jkorah/collection/')
Index built in  103.25177598  seconds.
>>> x=a.and_query(['with', 'without', 'yemen'])
Results for the Query:  with AND without AND yemen
Total Docs retrieved:  6
Text-99.txt
Text-159.txt
Text-121.txt
Text-115.txt
Text-117.txt
Text-86.txt
Retrieved in  0.000526905059814  seconds
>>> x=a.and_query(['with', 'without', 'yemen', 'yemeni'])
Results for the Query:  with AND without AND yemen AND yemeni
Total Docs retrieved:  2
Text-99.txt
Text-121.txt
Retrieved in  0.00115895271301  seconds
>>> x=a.print_dict()
……..
……..
zone [(111, [125]), (122, [82]), (147, [206]), (198, [1739]), (231, [632]), (249, [101]), (293,
[88]), (306, [82, 519]), (329, [288]), (350, [335]), (371, [115]), (393, [246])]
zones [(63, [451]), (261, [522]), (379, [798])]
zoo [(400, [86]), (401, [196, 393])]
zoom [(171, [640])]
zoomed [(410, [830])]
……..
……..
>>> x=a.print_doc_list()
……..
……..
Doc ID:  407  ==>  Text-49.txt
Doc ID:  408  ==>  Text-72.txt
Doc ID:  409  ==>  Text-36.txt
Doc ID:  410  ==>  Text-349.txt
Doc ID:  411  ==>  Text-77.txt
Doc ID:  412  ==>  Text-183.txt
Doc ID:  413  ==>  Text-55.txt
Doc ID:  414  ==>  Text-201.txt
Doc ID:  415  ==>  Text-242.txt
……..
……..
```

Submission:

Submit the following files on blackboard as a .zip file.

1. index.py

2. README file --- briefly describing your merge algorithm and any other comments

3. Output.txt: containing 5 queries queries and the output generated by your code.