MSCI 541/720 - HW2
Assigned: Tuesday, June 1, 2021
Due: 11:59PM EDT, Friday, June 11, 2021 (1.5 weeks)

Data
Same data as HW1 plus the file topics.401-450.txt, which is found in Learn.

Problems (3 Problems)

Problem 1. Modify your IndexEngine program from HW1 (program 1) to also perform an in-memory inversion of the LATimes collection. The text of a document should come from the following tags: TEXT, HEADLINE, GRAPHIC. You should strip (remove) any tags found within those tags from the text. You should then tokenize the remaining text by down casing all characters and treat as tokens any contiguous sequence of alphanumeric.

For this HW, do not remove stopwords nor stem words. After tokenizing, you will know the number of "words" in the document, and this is known as the document's length. You need to store the document length along with the other document metadata and be able to retrieve it later.

After tokenizing, the tokens should be converted to their integer id form using your lexicon. Your lexicon consists of two dictionaries: one that maps from a term to its unique integer id, and one maps from a term's integer id back to the term.

The program should create a postings list for each term as explained in class (count of term and doc id stored as a pair in ascending order by doc id). The postings lists should be stored in a dictionary that provides O (1) access to a term's postings list given the term's integer id. You do not need to store position information.

The program should save your inverted index and your lexicon to files in the same directory as the metadata files. You may save the posting lists in plain text or use tools to serialize the data structure for easy saving and loading. You will turn in your code for this problem. A C# program can do this processing in a few minutes using about 800 MB of RAM.

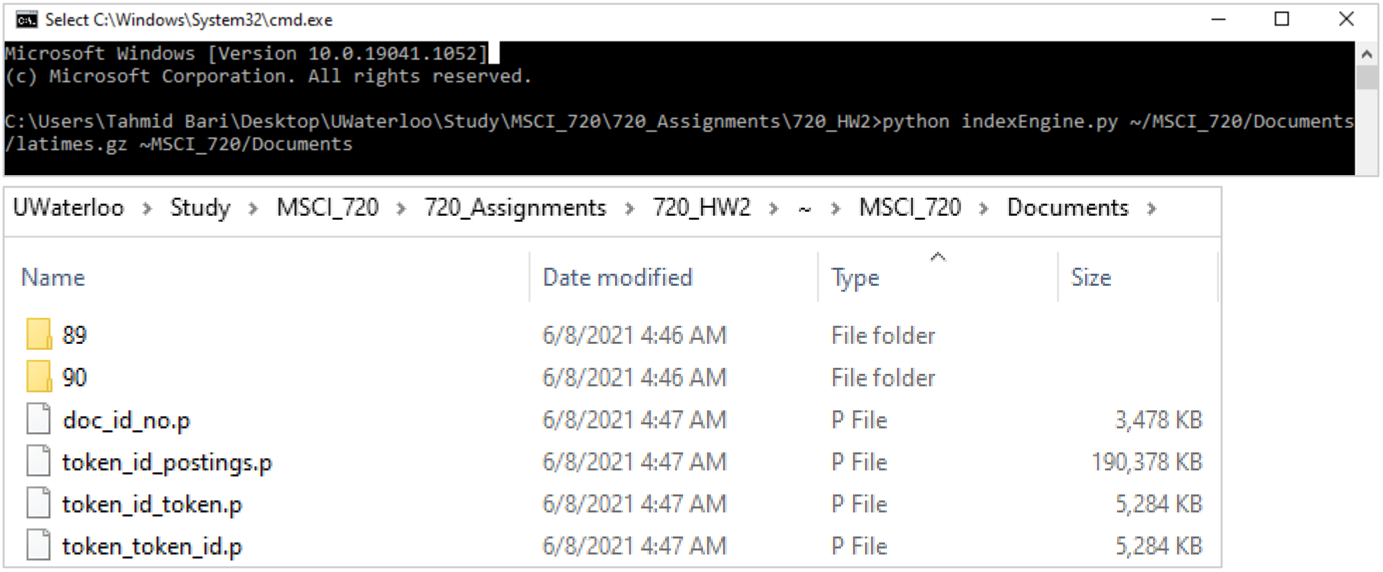
Note: Some students have problems with their java programs running out of memory. Usually, the problem is simply that they need to instruct java to have a larger maximum heap size. From the command line, this is done using the -Xmx option for java. Within Eclipse, or Netbeans, etc. you will need to read instructions on how to do this. A good place to start reading is with a web search for "increase java heap size". All other programming languages that I know about will simply use as much memory as they need.

Answer 1. (**indexEngine.py**) program has been modified to perform an in-memory inversion of the LATimes collection. TEXT, HEADLINE, GRAPHIC: tags have come out from the text of a document. Any tags found within those tags from the text have been removed. After that, tokenized the remaining text by downcasing all characters and tokens are being considered any contiguous sequence of alphanumeric.

Tokens have been converted to their integer id from using the lexicon after tokenization. And it consisted of two dictionaries: one that maps from a term to its unique integer id, and one maps from a term's integer id back to the term. The program has also created a postings list for each term as explained in class (count of term and doc id stored as a pair in ascending order by doc id). The postings lists have been stored in a dictionary that provides O (1) access to a term's postings list given the term's integer id.

(indexEngine.py) program has saved the inverted index and lexicon to files in the same directory as the metadata files. Here are the files being generated after executing the program:

- doc_id_no.p
- token_id_postings.p
- token_id_token.p
- token_token_id.p



```
Select C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19041.1052]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Tahmid Bari\Desktop\UWaterloo\Study\MSCI_720\720_Assignments\720_HW2>python indexEngine.py ~/MSCI_720/Documents
/latimes.gz ~MSCI_720/Documents
```

UWaterloo › Study › MSCI_720 › 720_Assignments › 720_HW2 › ~ › MSCI_720 › Documents ›

Name	Date modified	Type	Size
89	6/8/2021 4:46 AM	File folder	
90	6/8/2021 4:46 AM	File folder	
doc_id_no.p	6/8/2021 4:47 AM	P File	3,478 KB
token_id_postings.p	6/8/2021 4:47 AM	P File	190,378 KB
token_id_token.p	6/8/2021 4:47 AM	P File	5,284 KB
token_token_id.p	6/8/2021 4:47 AM	P File	5,284 KB

Problem 2. Write a program that will perform Boolean AND retrieval for a list of queries. Details:

The list of queries comes from the search topics in the topics.401-450.txt file. Each search topic has a title. We will treat the titles as our queries. We will use a subset of the topics as our queries. We will use all topics minus topics 416, 423, 437, 444, and 447. Extract the topics' titles and treat each title as the query that a user would enter into a search engine. You may do this by hand. No need to write a program. For example, the query for topic 401 should be:

foreign minorities, Germany

Save these 45 queries to a file in some format that also records the topic number, e.g. 401. I suggest a simple format: place the topic number on one line, and then on the next place the query.

Boolean AND retrieval returns any document that contains all of the query's words. Boolean retrieval does not return the documents in a specific order, but instead simply identifies the set of documents that contain all of the query's words. Boolean retrieval has awful quality and is rarely used by search engines.

Your program, named BooleanAND, should be able to be run from the command prompt1. Your program will take 3 arguments: the directory location of your index, the queries file, and the name of a file to store your output. For example:

BooleanAND /home/smucker/latimes-index queries.txt hw2-results-smucker.txt

The output of your program is the results for all of the given queries. You must make use the stored lexicon, inverted index, and metadata so that your retrieval is fast and efficient. Your result file must have the following internal format:

topicID q0 docno rank score runTag

where the above fields are separated by a single space, and the rows are sorted in ascending order by the topicID (primary key) and the score (secondary key). This is a TREC results file format. The columns are:

topicID: the topic number, an integer value

Q0: the letter Q followed by the number 0, ignored but required

docno: the document's docno, no extra whitespace

rank: the rank of the document from 1 to number of documents retrieved. Rank 1 is the best document. Each document should have a different rank.

score: a numeric value such that if score_i > score_k, then docno_i should be ranked higher (nearer the top of the list, i.e. a smaller rank) than docno_k. For a Boolean retrieval, you may want to assign a score of (number of documents retrieved)-rank, so that you fix the order of the documents even though there is no order to Boolean retrieval.

runTag: a unique string of 12 or fewer alphanumeric characters that uniquely identifies your results (aka a run). For this assignment, you should select a runtag that is the same as your WatIAM/Quest login name plus the string AND, e.g. msmuckerAND (do not use your numeric student id).

In summary, only 4 columns are used: topicID, docno, score, and runTag, but you must include all 6.

The results file does NOT include a header. The results file only consists of the results of the retrieval. Do not write "topicID q0 docno rank score runTag" on the first line of the file.

Do NOT lowercase the docno codes. It is critical that your docno codes be exactly as they are in the document minus any leading or trailing spaces. Once you have your program written, do the following:

A. Create a small, few (4?) document collection and some queries to demonstrate that your program can correctly perform Boolean AND retrieval. Your documents should be very short, too. Show your collection, queries, and the output and explain how this demonstrates that your program works correctly. [It can be a really good idea to make this small collection for development purposes.]

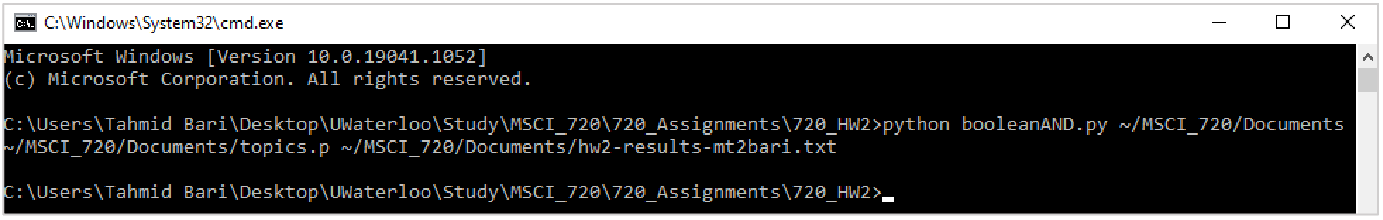
B. Run your program with the latimes and the 45 queries and name the results file WatIAMUserID-hw2-results.txt where WatIAMUserID is your WatIAM user ID (NOT your student number).

Do not forget: You must tokenize the query the same way you tokenize the documents!

Answer 2. (**booleanAND.py**) program has performed the retrieval for a list of queries, which comes from the search topics in the (topics.401-450.txt) file. All of the topics have been used except topics (416, 423, 437, 444, and 447). Another program (**getQueries.py**) has been written to perform this operation. Also, extracted the topics' titles and treated each title as the query that a user would enter into a search engine.

The program has taken 3 arguments: the directory location of your index, the queries file, and the name of a file to store the output.

We can execute the (booleanAND.py) program using the following argument:
Output File Name: **hw2-results-mt2bari.txt**



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19041.1052]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Tahmid Bari\Desktop\UWaterloo\Study\MSCI_720\720_Assignments\720_HW2>python booleanAND.py ~/MSCI_720/Documents
~/MSCI_720/Documents/topics.p ~/MSCI_720/Documents/hw2-results-mt2bari.txt

C:\Users\Tahmid Bari\Desktop\UWaterloo\Study\MSCI_720\720_Assignments\720_HW2>
```

C:\Users\Tahmid Bari\Desktop\UWaterloo\Study\MSCI_720\720_Assignments\720_HW2>python booleanAND.py ~/MSCI_720/Documents ~/MSCI_720/Documents/topics.p ~/MSCI_720/Documents/hw2-results-mt2bari.txt

In order to demonstrate that Boolean AND retrieval is functioning properly, the following test document was created. The test document comprises of 4 individual articles in a format that is similar to the LATimes.gzip file:

```
<DOC>
<DOCNO> LA021890-0100 </DOCNO>
<DOCID> 178084 </DOCID>
<DATE>
<P>
February 18, 1990, Sunday, Home Edition
</P>
</DATE>
<SECTION>
<P>
Opinion; Part M; Page 5; Column 1; Op-Ed Desk
</P>
</SECTION>
<LENGTH>
<P>
753 words
</P>
</LENGTH>
<HEADLINE>
<P>
EVIL LIES IN SYSTEM, NOT IN THE RACE;
</P>
<P>
GERMANY: IT'S NOT THE PREDISPOSITION OF A PEOPLE BUT THE QUEST OF A DICTATOR
FOR TOTAL CONTROL THAT LEADS TO AGGRESSION AND WAR.
</P>
</HEADLINE>
<BYLINE>
<P>
By Jeane Kirkpatrick, Jeane Kirkpatrick is a syndicated columnist in
Washington.
</P>
</BYLINE>
<DATELINE>
```

```
<P>
WASHINGTON
</P>
</DOC>
<DOC>
<DOCNO> LA090490-0093 </DOCNO>
<DOCID> 271251 </DOCID>
<DATE>
<P>
September 4, 1990, Tuesday, Home Edition
</P>
</DATE>
<SECTION>
<P>
World Report; Part H; Page 1; Column 2; World Report
</P>
</SECTION>
<LENGTH>
<P>
2490 words
</P>
</LENGTH>
<HEADLINE>
<P>
EUROPEANS HAVE MUCH TO LOSE IN THE GULF PUZZLE;
</P>
<P>
FRANCE, WHICH HAS PINNED ITS MIDEAST POLICY ON BAGHDAD FOR TWO DECADES,
PROBABLY HAS THE MOST AT STAKE IN THE REGION.
</P>
</HEADLINE>
<BYLINE>
<P>
By RONE TEMPEST, TIMES STAFF WRITER
</P>
</BYLINE>
<DATELINE>
<P>
PARIS
</P>
</DATELINE>
<TEXT>
<P>
On the ground in Saudi Arabia and on the waters of the Persian Gulf, the United
States carries the biggest stick in the showdown with the forces of Iraqi
leader Saddam Hussein.
</P>
<P>
</DOC>
```

```
<DOC>
<DOCNO> LA050789-0068 </DOCNO>
<DOCID> 53734 </DOCID>
<DATE>
<P>
May 7, 1989, Sunday, Home Edition
</P>
</DATE>
<SECTION>
<P>
Part 1; Page 1; Column 1; Foreign Desk
</P>
</SECTION>
<LENGTH>
<P>
2602 words
</P>
</LENGTH>
<HEADLINE>
<P>
NO MELTING POT;
</P>
<P>
EUROPE BUSY CLOSING DOOR TO FOREIGNERS
</P>
</HEADLINE>
<BYLINE>
<P>
By TYLER MARSHALL, Times Staff Writer
</P>
</BYLINE>
<DATELINE>
<P>
ALGECIRAS, Spain
</P>
</DATELINE>
<TEXT>
<P>
They come at night, braving the treacherous strait in leaky 20-foot fishing
boats called pateras.
</P>
<P>
</DOC>
<DOC>
<DOCNO> LA122389-0060 </DOCNO>
<DOCID> 152613 </DOCID>
<DATE>
<P>
December 23, 1989, Saturday, Home Edition
```

```
</P>
</DATE>
<SECTION>
<P>
Part A; Page 1; Column 1; Foreign Desk
</P>
</SECTION>
<LENGTH>
<P>
2378 words
</P>
</LENGTH>
<HEADLINE>
<P>
ROMANIANS TOPPLE CEAUSESCU REGIME;
</P>
<P>
EAST EUROPE: THE DICTATOR AND HIS WIFE FLEE BUCHAREST AS HIS 24-YEAR TYRANNY
COMES TO A VIOLENT HALT.
</P>
</HEADLINE>
<BYLINE>
<P>
By DAN FISHER, TIMES STAFF WRITER
</P>
</BYLINE>
<DATELINE>
<P>
LONDON
</P>
</DATELINE>
<TEXT>
<P>
The Warsaw Pact's last Stalinist domino, the tyrannical regime of Romanian
President Nicolae Ceausescu, toppled with a violent crash Friday after several
days of Europe's bloodiest fighting since the end of World War II.
</P>
<P>
</DOC>
<DOC>
```

The queries below were then run using the Boolean AND retrieval script against the test document collection.

topics = {1: 'race', 2: 'puzzle', 3: 'foreigners', 4: 'regime', 5: 'writers'}

After performing the Boolean AND retrieval script, the following was outputted:

```
1 q0 LA021890-0100 1 0 mt2bariAND
2 q0 LA021890-0100 1 1 mt2bariAND
2 q0 LA090490-0093 2 0 mt2bariAND
3 q0 LA122389-0060 1 0 mt2bariAND
```

```
4 q0 LA122389-0060 1 0 mt2bariAND
5 q0 LA021890-0100 1 2 mt2bariAND
5 q0 LA090490-0093 2 1 mt2bariAND
5 q0 LA050789-0068 3 0 mt2bariAND
```

Topic 1: 'race'
Only the first document contains 'race'. Thus, the Boolean AND script outputted only the first article for the query.

Topic 2: 'puzzle'
Both the first and second articles contain the words 'puzzle', while the other two articles don't contain either word.

Topic 3: 'foreigners'
Similar to the first topic, only the fourth article contains the word 'foreigners', thus it is the only result in the output.

Topic 4: 'regime'
Only the fourth article has the term 'regime', and thus it is the only one shown in the output result.

Topic 5: 'writers'
The first, second, and fourth articles all contain the word 'writers', and thus they are shown in the output result.

Problem 3. For topics 401 and 403, take the rank 1 through 10 documents that you returned in the results file generated in problem 2 and judge the documents for relevance to the topic. (Note: different students are likely to have different documents here because there is no defined sort order for a Boolean retrieval.) To guide your decision about what is and is not relevant, read the original topic description including the narrative section for these topics. Report in a table the rank, the docno, your judgment (relevant or non-relevant), and a brief reason why the document is or is not relevant. To judge the relevance, you need to actually read the documents and also the topic description/narrative. Also report for topics 401 and 403, the precision of the 10 documents that you have judged.

Answer 3.

Topic 401: 'foreign minorities, Germany'

Rank	Docno	Relevance	Explanation
1	LA021890-0100	Non-relevant	Minorities refers to minorities of adults, not foreign minorities
2	LA090490-0093	Non-relevant	No match for 'foreign minorities'
3	LA050789-0068	Non-relevant	Document discusses countries other than Germany
4	LA122389-0060	Non-relevant	Article focuses on Romania
5	LA111289-0073	Non-relevant	Articles just mentions plight of minorities
6	LA121890-0117	Non-relevant	Examines many countries, not just Germany
7	LA100889-0019	Non-relevant	Article is about Latvian politics
8	LA040490-0003	Non-relevant	Focuses on Lithuanian politics
9	LA051390-0170	Non-relevant	Article concerns the USSR
10	LA052190-0065	Non-relevant	Article pertains Romania

Precision: 0.0

Topic 403: 'osteoporosis'

Rank	Docno	Relevance	Explanation
1	LA111589-0004	Non-relevant	Article is on a study concerning menopause
2	LA051490-0120	Non-relevant	Document discusses taking an oral fluoride supplement to improve bone density
3	LA110490-0091	Non-relevant	Briefly mentions osteoporosis in passing; mostly about HGH
4	LA101890-0267	Non-relevant	Article discusses Didronel, an oral supplement for bone density loss
5	LA032489-0093	Non-relevant	Mentions osteoporosis in passing
6	LA071290-0133	Non-relevant	Etidronate, which is a drug used to abate osteoporosis
7	LA092890-0067	Non-relevant	Article discusses calcium supplements for slowing bone loss
8	LA120390-0005	Non-relevant	Article briefly mentions osteoporosis
9	LA030689-0082	Non-relevant	Talks about osteoporosis, but does not discuss dietary intakes
10	LA011389-0029	Non-relevant	Relates to estrogen cream's effects on osteoperosis

Precision: 0.4

Appendix:

Croft, W. B. Metzler, D. Strohman, T. (2015) *Search Engines Information Retrieval in Practice* Retrieved from https://ciir.cs.umass.edu/downloads/SEIRiP.pdf

Thunderstone February 12, 2019 *What is the Difference Between a Database and a Search Engine?* Retrieved from https://www.thunderstone.com/blog/archive/what-is-the-difference-between-a-database-and-a-search-engine/

Lucidworks July 26, 2019 *Full Text Search Engines vs. DBMS* Retrieved from https://lucidworks.com/post/full-text-search-engines-vs-dbms/

GitHub repository https://github.com/rfarmaha