CSE 435/535 Information Retrieval (Fall 2016)
Project Two: Boolean Query and Inverted Index

Due Date: October 17th 2016, 23:59 pm

Overview

In project part two you will be given Lucene index generated from the RCV2 multilingual news corpus [1]. Your first task is to get familiar with the Lucene index and learn how to interact with it using Lucene APIs. Then you need to build your own inverted index using the information extracted from the given data. Your index should be stored as Linked List in memory as the examples shown in textbook. After that, you are required to implement two strategies to return boolean query results: term-at-a-time and document-at-a-time. **Java** should be used for this project.

Input Dataset

The given Lucene index is generated by indexing the Reuters Corpus Volume 2 on Solr (which is similar to what you have done in project part one). It contains news stories in twelve languages: Dutch, French, German, Japanese, Russian, Portuguese, Spanish, Latin American Spanish, Italian, Danish, Norwegian, and Swedish. Corresponding field names are: text_nl, text_fr, text_de, text_ja, text_ru, text_pt, text_es, text_es, text_it, text_da, text_no, and text_sv.

Each document contains only ONE of the above text fields and is assigned with a document ID. Some example documents:

```
[
  {
    "text_de": " Hauppauge (Reuter) - Die meisten der 230 Insassen des vor New York explodierten TWA-Jumbos starben
      medizinischen Untersuchungen zufolge unmittelbar nach der Explosion noch in der Luft. Der zuständige Gerichtsmediziner
      Charles Wetli sagte am Montag, er sei sich sicher, daß wahrscheinlich alle Insassen durch die Wucht des Aufpralls ihrer
      Köpfe auf die vor ihnen gelegenen Sitze oder durch den Rückschlag gegen die Kopfstützen starben. Der Pathologe sagte,
      als Privatmann glaube er, daß die Explosion durch eine Bombe verursacht wurde. Pathologische Beweise dafür habe er aber
      nicht.  Der Jumbo war vor einem Monat mit 230 Menschen an Bord vor der Küste New Yorks explodiert und in den Atlantik
      gestürzt. Überlebende gab es nicht. Experten schließen weiter weder einen technischen Defekt noch eine Bombe oder einen
      Raketenangriff als Ursache für die Explosion des Flugzeuges aus.  ker.  (c) Reuters Limited 1996. ",
    "id": "150001"
  },
  {
    "text_ja": "    ［モスクワ　２０日　ロイター］　ロシアのザベリューハ副首相は、９６年の同国穀物生産高が７７００万トンとなり
      、輸出と輸入は同水準になる、との見方を示した。
        同副首相は記者会見で、「国内穀物生産は、国内需要を充足するに十分な水準に達することが可能だ。輸出量と輸入量はほぼ同水準
      になるだろう。穀物生産高が７７００万トンならば、完全自給の可能性も出てくる」と述べた。
        同副首相は、輸出入の数値予想は明らかにしなかった。インタファクス通信が前週伝えたところによると、同副首相は、今年の穀物
      輸出量を４００～５００万トン、同輸入量もこれとほぼ同水準になるとの見通しを明らかにしている。
      原文参照番号［nMY2001713］          (c) Reuters Limited 1996 ",
    "id": "300064"
  },
  {
    "text_fr": " LONDRES, 20 août, Reuter - United Airlines, compagnie aérienne du groupe UAL Corp, doit annoncer sous peu une
      commande de trois milliards de dollars portant sur une cinquantaine d'appareils, rapporte le Financial Times mardi.
      United commanderait 27 gros porteurs à Boeing Co pour $2,5 milliards et une vingtaine d'appareils de capacité inférieure
      . Pour ces derniers, United n'aurait pas encore choisi entre Boeing ou Airbus Industrie, ajoute le quotidien financier.
      /WYE.  (c) Reuters Limited 1996. ",
    "id": "60012"
  }
]
```

Step 1: Interact with Lucene Index and Build Your Own Inverted Index

As the examples in textbook, postings lists should be stored as **Linked Lists**.

You will need to construct your own index in which postings of each term should be ordered by *increasing document IDs*. For example:

$$\begin{tikzcd}
\text{term1} \arrow[r] & \text{doc1} \arrow[r] & \text{doc2} \arrow[r] & \text{doc4} \\
\text{term2} \arrow[r] & \text{doc2} \arrow[r] & \text{doc4} \arrow[r] & \text{doc7}
\end{tikzcd}$$

In order to use the APIs to talk to the given index and get the information you need, import the Lucene module to your project. “lucene-core-6.2.0.jar” can be downloaded from here: https://lucene.apache.org/core/ .

Note:

- You are supposed to ONLY use the methods included in package **org.apache.lucene.index** to get information for your index. DO NOT use other classes which can answer the queries directly, such as IndexSearcher.
- Take a look at the java doc for the class IndexReader (https://lucene.apache.org/core/6_2_1/core/org/apache/lucene/index/IndexReader.html).

Step 2: Boolean Query Processing

You are required to implement the following methods, and provide the results of a set of boolean queries (AND/OR) **on your own index**. Results should be output as a .txt file in the required format.

1. Get postings lists

   This method retrieves the postings lists for each of the given query terms. Input of this method will be a set of terms: term0, term1, … , termN. It should output the postings for each term in the following format:

```
GetPostings
term0
Postings list: 1000 2000 3000 … (by increasing document IDs)
GetPostings
term1
Postings list: 1000 3000 6000 … (by increasing document IDs)
…
GetPostings
termN
Postings list: 2000 7000 8000 … (by increasing document IDs)
```

2. Term-at-a-time AND query

   This method is to implement multi-term boolean AND query on the index using term-at-a-time strategy (TAAT). Input of this function will be a set of terms: term0, term1, …, termN. Output format of this method should be:

   ```
   TaatAnd
   term0 term1 … termN
   Results: 1000 2000 3000 … (by increasing document IDs)
   Number of documents in results: x
   Number of comparisons: y
   ```

   If the result of the query is empty, output:

   ```
   TaatAnd
   term0 term1 … termN
   Results: empty
   Number of documents in results: 0
   Number of comparisons: y
   ```

3. Term-at-a-time OR query

   This function is to implement multi-term boolean OR query on the index using TAAT. Input of this function will be a set of query terms: term0, term1, … , termN. Output format of this method should be:

   ```
   TaatOr
   term0 term1 … termN
   Results: 1000 2000 3000 … (by increasing document IDs)
   Number of documents in results: x
   Number of comparisons: y
   ```

4. Document-at-a-time AND query

   This function is to implement multi-term boolean AND query on the index using document-at-a-time strategy (DAAT). Input of the function will be a set of query terms: term0, term1, …, termN. Output the following to the output file:

   ```
   DaatAnd
   term0 term1 … termN
   Results: 1000 2000 3000 … (by increasing document IDs)
   Number of documents in results: x
   Number of comparisons: y
   ```

   If the result of the query is empty, output:

   ```
   DaatAnd
   term0 term1 … termN
   Results: empty
   Number of documents in results: 0
   Number of comparisons: y
   ```

5. Document-at-a-time OR query

This function is to implement multi-term boolean OR query on the index using DAAT. Input of this function will be a set of query terms: term0, term1, …, termN. Output the following to the output file:

```
DaatOr
term0 term1 … termN
Results: 1000 2000 3000 … (by increasing document IDs)
Number of documents in results: x
Number of comparisons: y
```

There is no requirement for the names of your Java methods. However, you should submit a .jar file **named exactly as "UBITName_project2.jar"**, and your program should start running by executing the following command **on Timberlake**:

```
java -jar UBITName_project2.jar path_of_index output.txt input.txt
```

Here, the first input parameter *path_of_index* should be able to accept a string indicating the path of the given Lucene index. The second input parameter *output.txt* refers to the output file, while the third one, *input.txt* refers to the input file which contains the query terms.

The following assumptions can be made:

- The number of input query terms can be varied.
- All of the input query terms are selected from the vocabulary.
- Query terms should be processed in the order in which they are written in the query. Say, you should process term0 first, then term1, so on and so forth.
- Input query terms will be separated by 'one blank space' in the input file. You can assume there is no blank space within any query term.
- DO NOT use Java build-in methods to do unions and intersections on postings lists directly. DO NOT use build-in functions to check the presence of document IDs directly, such as List.contains(), when you are performing union/intersection on postings. DO NOT use classes such as IndexSearcher to answer queries directly. Create your own pointers and have fun!
- Output should be formatted exactly the same as required. Otherwise you will not be able to get credits because grading will be automated!

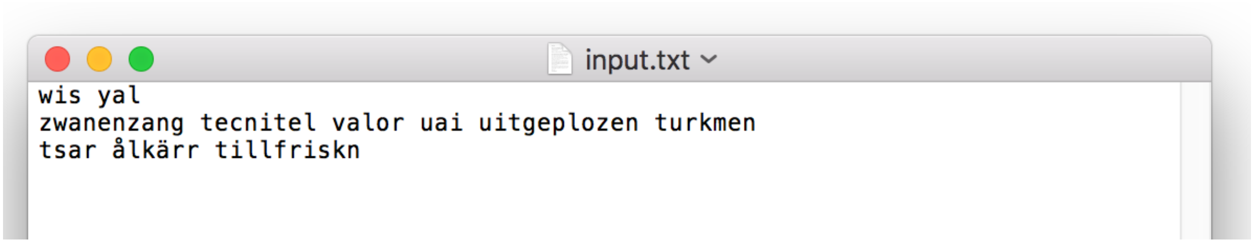
Sample Input Output

The input file will be a .txt file containing multiple lines. Each line refers to a set of query terms which are separated by one blank space. For example:

```
term0 term1
term2 term3 term4 term5 term6 term7
term8 term9 term10
```

In this case, the output should be in the following order:

GetPostings term0 term1
TaatAnd term0 term1
TaatOr term0 term1
DaatAnd term0 term1
DaatOr term0 term1
GetPostings term2 term3 term4 term5 term6 term7
TaatAnd term2 term3 term4 term5 term6 term7
TaatOr term2 term3 term4 term5 term6 term7
DaatAnd term2 term3 term4 term5 term6 term7
DaatOr term2 term3 term4 term5 term6 term7
GetPostings term8 term9 term10
TaatAnd term8 term9 term10
TaatOr term8 term9 term10
DaatAnd term8 term9 term10
DaatOr term8 term9 term10



```
wis yal
zwanenzang tecnitel valor uai uitgeplozen turkmen
tsar ålkärr tillfriskn
```

The corresponding output file for the input mentioned before will be:

GetPostings
term0
Postings list: 1000 2000 3000 …
GetPostings
term1
Postings list: 1000 2000 3000 …
TaatAnd
term0 term1
Results: 1000 2000 3000 …
Number of documents in results: x
Number of comparisons: y
TaatOr
term0 term1
Results: 1000 2000 3000 …
Number of documents in results: x
Number of comparisons: y
DaatAnd
term0 term1

```
Results: 1000 2000 3000 …
Number of documents in results: x
Number of comparisons: y
DaatOr
term0 term1
Results: 1000 2000 3000 …
Number of documents in results: x
Number of comparisons: y
GetPostings
term2
Postings list: 1000 2000 3000 …
GetPostings
term3
Postings list: 1000 2000 3000 …
GetPostings
term4
Postings list: 1000 2000 3000 …
GetPostings
term5
Postings list: 1000 2000 3000 …
GetPostings
term6
Postings list: 1000 2000 3000 …
GetPostings
term7
Postings list: 1000 2000 3000 …
TaatAnd
term2 term3 term4 term5 term6 term7
Results: 1000 2000 3000 …
Number of documents in results: x
Number of comparisons: y
TaatOr
term2 term3 term4 term5 term6 term7
Results: 1000 2000 3000 …
Number of documents in results: x
Number of comparisons: y
DaatAnd
term2 term3 term4 term5 term6 term7
Results: 1000 2000 3000 …
Number of documents in results: x
Number of comparisons: y
DaatOr
term2 term3 term4 term5 term6 term7
Results: 1000 2000 3000 …
Number of documents in results: x
Number of comparisons: y
GetPostings
term8
Postings list: 1000 2000 3000 …
GetPostings
term9
Postings list: 1000 2000 3000 …
GetPostings
term10
Postings list: 1000 2000 3000 …
TaatAnd
term8 term9 term10
Results: 1000 2000 3000 …
Number of documents in results: x
Number of comparisons: y
TaatOr
term8 term9 term10
Results: 1000 2000 3000 …
Number of documents in results: x
Number of comparisons: y
DaatAnd
term8 term9 term10
Results: 1000 2000 3000 …
Number of documents in results: x
Number of comparisons: y
DaatOr
term8 term9 term10
Results: 1000 2000 3000 …
Number of documents in results: x
Number of comparisons: y
```

Grading and Evaluation

A successful implementation should be able to support all the functions mentioned before. It should also generate correct results in the required format. Failure in following the instructions and requirements will result in 0.

Rubrics:
Total marks for this project: 15
Successfully rebuild index and correct output for GetPostings: 5
Correct strategy for TAAT: 3
Correct strategy for DAAT: 3
Correct results for TaatAnd / TaatOr / DaatAnd / DaatOr: 1 + 1 + 1 + 1

What to Submit

You should use cse-submit script to submit a .jar file named exactly as "**UBITName_project2.jar**". Double check and make sure your submission can be executed successfully **on Timberlake**.
Late submission will NOT be accepted.

References

[1] http://trec.nist.gov/data/reuters/reuters.html