CSC 483/583: Programming Project (200 pts)

Building (a part of) Watson

Author: Sixiang Zhang

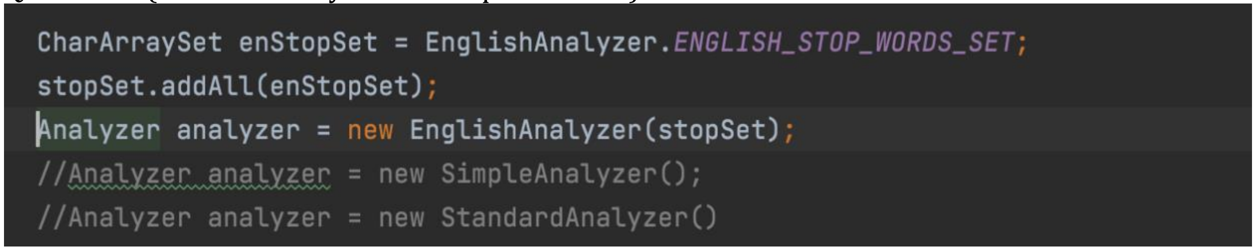
1. Description of the command line to run the python code, with an example :

   Run indexManager for creating indexes

   Run TextSearch for answer questions from the answer.txt

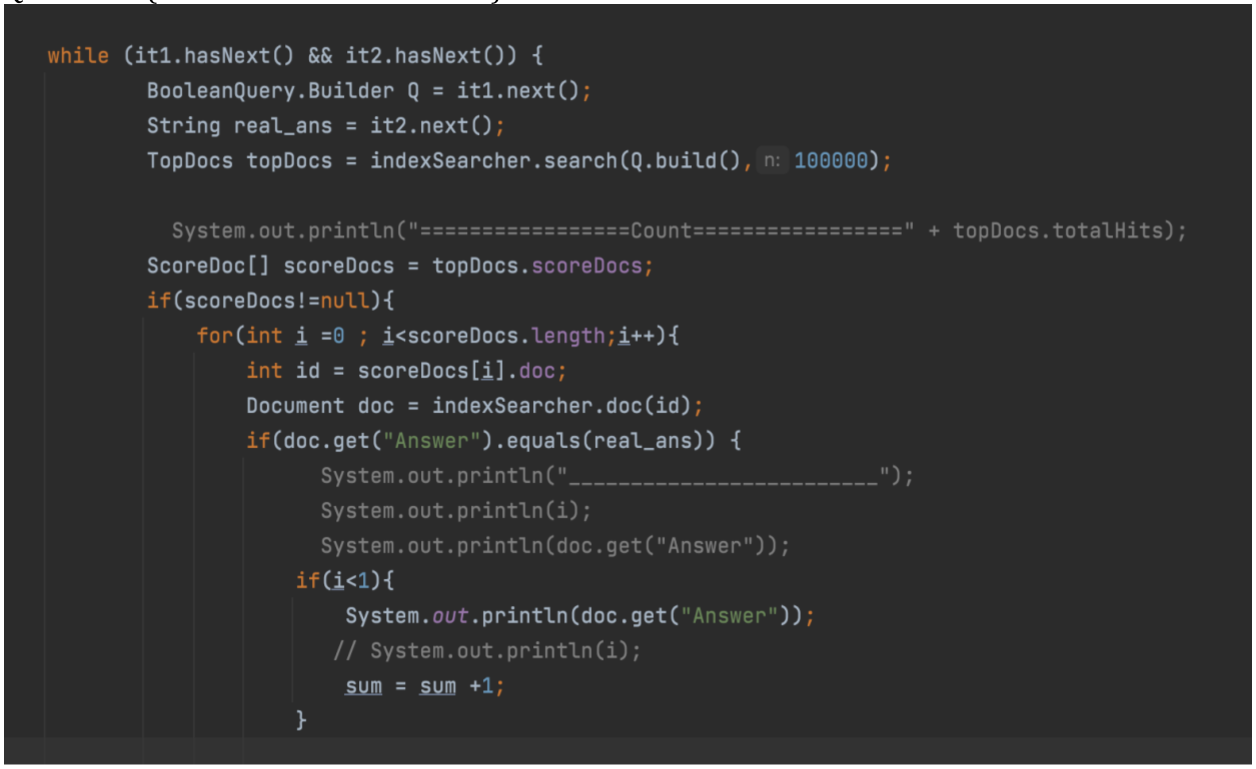
2. Description of the code. You don't have to describe every function implemented. But you should describe the main part of the code and indicate where each question is addressed.

   Question 1 (Different analyzer and stop words list)



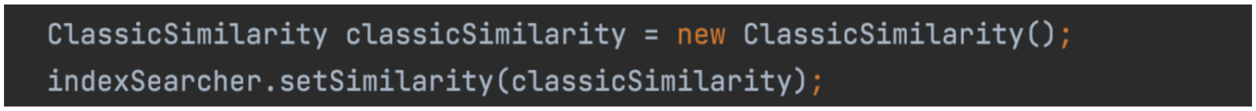
```
CharArraySet enStopSet = EnglishAnalyzer.ENGLISH_STOP_WORDS_SET;
stopSet.addAll(enStopSet);
Analyzer analyzer = new EnglishAnalyzer(stopSet);
//Analyzer analyzer = new SimpleAnalyzer();
//Analyzer analyzer = new StandardAnalyzer()
```

   Question 2 (P@1 function from scratch)



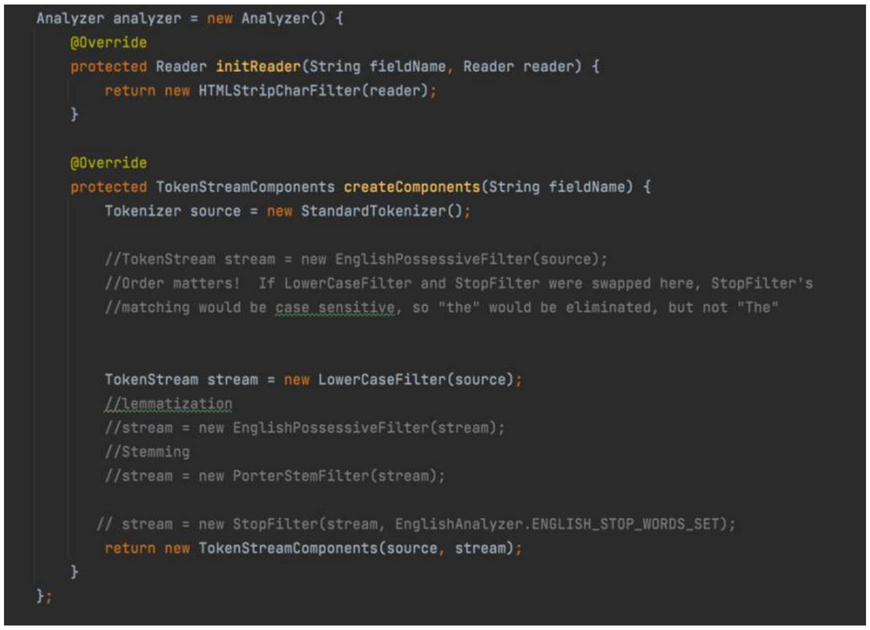
```
while (it1.hasNext() && it2.hasNext()) {
        BooleanQuery.Builder Q = it1.next();
        String real_ans = it2.next();
        TopDocs topDocs = indexSearcher.search(Q.build(), n: 100000);

          System.out.println("================Count================" + topDocs.totalHits);
        ScoreDoc[] scoreDocs = topDocs.scoreDocs;
        if(scoreDocs!=null){
            for(int i =0 ; i<scoreDocs.length;i++){
                int id = scoreDocs[i].doc;
                Document doc = indexSearcher.doc(id);
                if(doc.get("Answer").equals(real_ans)) {
                      System.out.println("________________________");
                      System.out.println(i);
                      System.out.println(doc.get("Answer"));
                    if(i<1){
                        System.out.println(doc.get("Answer"));
                       // System.out.println(i);
                        sum = sum +1;
                    }
```

Question 3 (Score function)



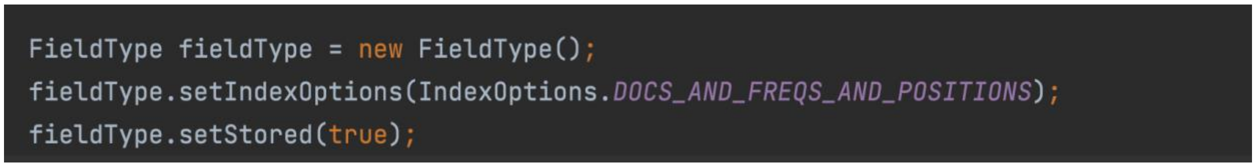
```java
ClassicSimilarity classicSimilarity = new ClassicSimilarity();
indexSearcher.setSimilarity(classicSimilarity);
```

Question 4 (Stem/lemmatization)

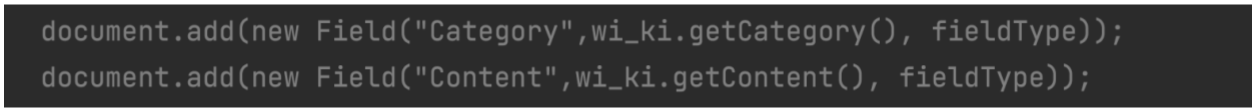


```java
Analyzer analyzer = new Analyzer() {
    @Override
    protected Reader initReader(String fieldName, Reader reader) {
        return new HTMLStripCharFilter(reader);
    }

    @Override
    protected TokenStreamComponents createComponents(String fieldName) {
        Tokenizer source = new StandardTokenizer();

        //TokenStream stream = new EnglishPossessiveFilter(source);
        //Order matters!  If LowerCaseFilter and StopFilter were swapped here, StopFilter's
        //matching would be case sensitive, so "the" would be eliminated, but not "The"


        TokenStream stream = new LowerCaseFilter(source);
        //lemmatization
        //stream = new EnglishPossessiveFilter(stream);
        //Stemming
        //stream = new PorterStemFilter(stream);

       // stream = new StopFilter(stream, EnglishAnalyzer.ENGLISH_STOP_WORDS_SET);
        return new TokenStreamComponents(source, stream);
    }
};
```

Question 5(positional index)

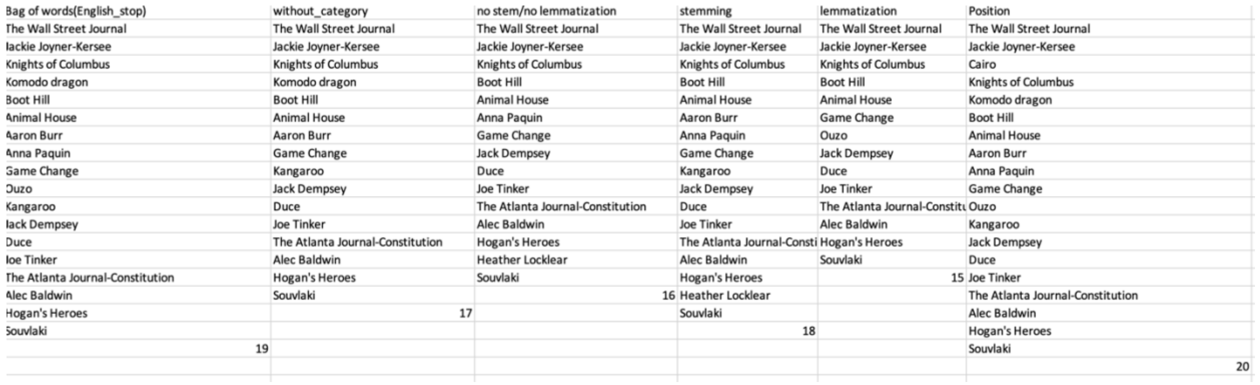


```java
FieldType fieldType = new FieldType();
fieldType.setIndexOptions(IndexOptions.DOCS_AND_FREQS_AND_POSITIONS);
fieldType.setStored(true);
```



```java
document.add(new Field("Category",wi_ki.getCategory(), fieldType));
document.add(new Field("Content",wi_ki.getContent(), fieldType));
```

3. Results



Bag of words(English_stop)	without_category	no stem/no lemmatization	stemming	lemmatization	Position
The Wall Street Journal	The Wall Street Journal	The Wall Street Journal	The Wall Street Journal	The Wall Street Journal	The Wall Street Journal
Jackie Joyner-Kersee	Jackie Joyner-Kersee	Jackie Joyner-Kersee	Jackie Joyner-Kersee	Jackie Joyner-Kersee	Jackie Joyner-Kersee
Knights of Columbus	Knights of Columbus	Knights of Columbus	Knights of Columbus	Knights of Columbus	Cairo
Komodo dragon	Komodo dragon	Boot Hill	Boot Hill	Boot Hill	Knights of Columbus
Boot Hill	Boot Hill	Animal House	Animal House	Animal House	Komodo dragon
Animal House	Animal House	Anna Paquin	Aaron Burr	Game Change	Boot Hill
Aaron Burr	Aaron Burr	Game Change	Anna Paquin	Ouzo	Animal House
Anna Paquin	Game Change	Jack Dempsey	Game Change	Jack Dempsey	Aaron Burr
Game Change	Kangaroo	Duce	Kangaroo	Duce	Anna Paquin
Ouzo	Jack Dempsey	Joe Tinker	Jack Dempsey	Joe Tinker	Game Change
Kangaroo	Duce	The Atlanta Journal-Constitution	Duce	The Atlanta Journal-Constitu	Ouzo
Jack Dempsey	Joe Tinker	Alec Baldwin	Joe Tinker	Alec Baldwin	Kangaroo
Duce	The Atlanta Journal-Constitution	Hogan's Heroes	The Atlanta Journal-Consti	Hogan's Heroes	Jack Dempsey
Joe Tinker	Alec Baldwin	Heather Locklear	Alec Baldwin	Souvlaki	Duce
The Atlanta Journal-Constitution	Hogan's Heroes	Souvlaki	Hogan's Heroes	15	Joe Tinker
Alec Baldwin	Souvlaki	16	Heather Locklear		The Atlanta Journal-Constitution
Hogan's Heroes	17		Souvlaki		Alec Baldwin
Souvlaki			18		Hogan's Heroes
19					Souvlaki
					20

4. Answers for all questions that do not require programming.

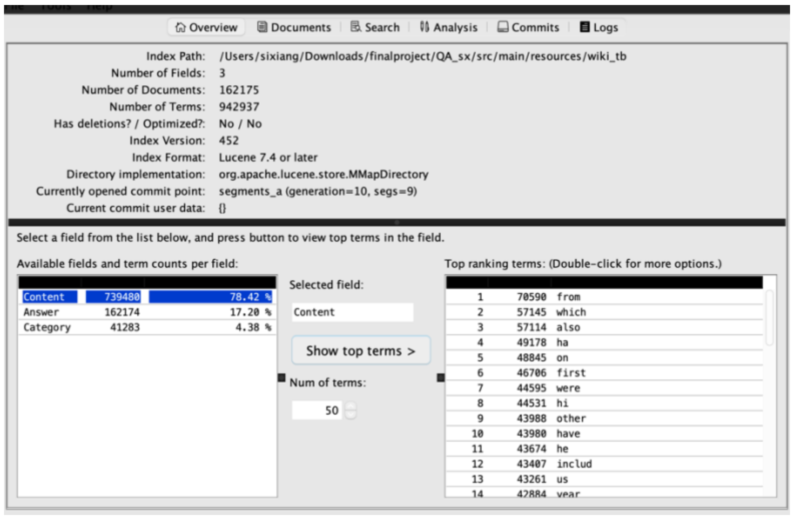
1) (100 pts) Indexing and retrieval: Index the Wikipedia collection with a state of the art Information Retrieval (IR) system such as Lucene (http://lucene.apache. org/) or Terrier (http://terrier.org/). Make sure that each Wikipedia page appears as a separate document in the index (rather than creating a document from each of the 80 files).

a) Describe how you prepared the terms for indexing (stemming, lemmatization, stop words, etc.).

Each document represents each record from the wiki-dataset. And when we make the indexes for the document, we can set different field for different columns. For example, the StringField and Text field both for the String type date, however, the former does not be tokenized. And the propose for the tokenization is for the search. In this project, we do not need search or query by the answers, thus, I decided not to tokenize the answer column.

Stemming, lemmatization, stop words, etc.:

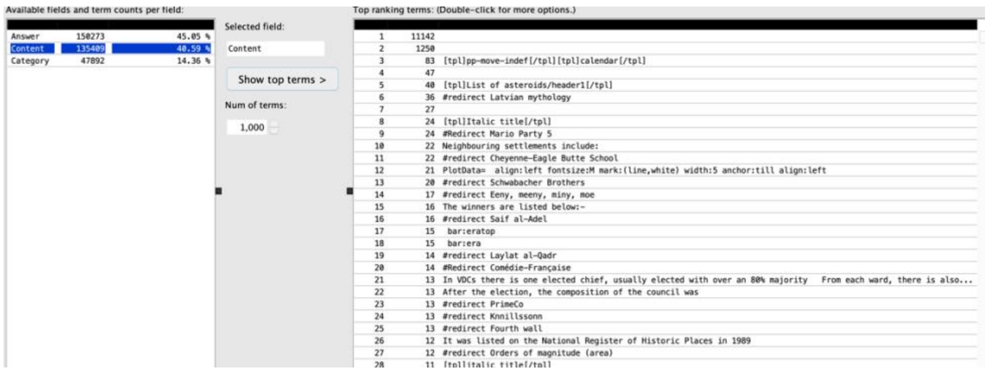
Then, there are many built-in Analyzers from Lucene. What is more, Lucene's analyzers are consist of tokenizers and filters. Different analyzers have different combinations of tokenizers and filters. I tried three different analyzers (StandardAnalyzer, SimpleAnalyzer, and EngligshAnalyzer) and one customer analyzer. The difference between the SimpleAnalyzer and StandardAnalyzer is that StandardAnalyzer has a stopwords filter, SimpleAnalyzer does not. Moreover, EnglishAnalyzer has stemming and lemmatization, but the other two do not. My customer analyzer is for the later adjusting the stemming and lemmatization effect. One more thing is that, the default stopwords is that: "a", "an", "and", "are", "as", "at", "be", "but", "by", "for", "if", "in", "into", "is", "it", "no", "not", "of", "on", "or", "such", "that", "the", "their", "then", "there", "these", "they", "this", "to", "was", "will", "with". However, after this stopwords list, I saw that the indexes have more stopwords. I added "from","which","also","ha","on","he","it" into the stop words list.



```
                 Index Path:  /Users/sixiang/Downloads/finalproject/QA_sx/src/main/resources/wiki_tb
           Number of Fields:  3
        Number of Documents:  162175
            Number of Terms:  942937
  Has deletions? / Optimized?:  No / No
              Index Version:  452
               Index Format:  Lucene 7.4 or later
   Directory implementation:  org.apache.lucene.store.MMapDirectory
Currently opened commit point:  segments_a (generation=10, segs=9)
   Current commit user data:  {}

Select a field from the list below, and press button to view top terms in the field.

Available fields and term counts per field:
Content    739480   78.42 %
Answer     162174   17.20 %
Category    41283    4.38 %

Selected field: Content
Show top terms >
Num of terms: 50

Top ranking terms: (Double-click for more options.)
 1   70590  from
 2   57145  which
 3   57114  also
 4   49178  ha
 5   48845  on
 6   46706  first
 7   44595  were
 8   44531  hi
 9   43988  other
10   43980  have
11   43674  he
12   43407  includ
13   43261  us
14   42884  year
```

b) What issues specific to Wikipedia content did you discover, and how did you address them?

There are many reference links, file paths, or picture paths that we need to remove from Wikipedia content. Most of them have similar format, for example, they start with </ref> or [ref] or [/ref]. Moreover, They are some strange characters, which are not English; I remove them from the content. Furthermore, we need to remove" _" from the words, which sometimes creates a huge "term" that we may not search. Moreover, there is a GUI called luke, which shows how Lucene stores the indexes. For example, as the picture shows (from luke) below. I need to remove the [#redicrect] or [#Redicrect] or #REDICRECT from content. Last but not least, always check the blank line. The Wikipedia content contains a lot of blank lines needed to be removed.



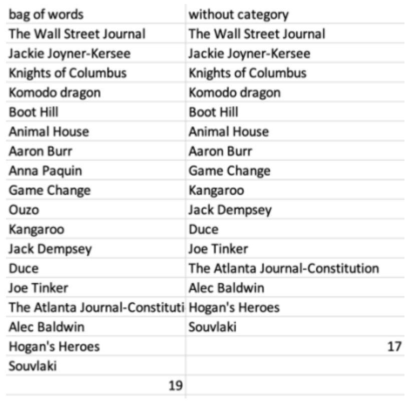
```
Available fields and term counts per field:
Answer     150273   45.05 %
Content    135409   40.59 %
Category    47892   14.36 %

Selected field: Content
Show top terms >
Num of terms: 1,000

Top ranking terms: (Double-click for more options.)
 1   11142
 2    1250
 3      83  [tpl]pp-move-indef[/tpl][tpl]calendar[/tpl]
 4      47
 5      40  [tpl]List of asteroids/header1[/tpl]
 6      36  #redirect Latvian mythology
 7      27
 8      24  [tpl]Italic title[/tpl]
 9      24  #Redirect Mario Party 5
10      22  Neighbouring settlements include:
11      22  #redirect Cheyenne-Eagle Butte School
12      21  PlotData=  align:left fontsize:M mark:(line,white) width:5 anchor:till align:left
13      20  #redirect Schwabacher Brothers
14      17  #redirect Eeny, meeny, miny, moe
15      16  The winners are listed below:-
16      16  #redirect Saif al-Adel
17      15   bar:eratop
18      15   bar:era
19      14  #redirect Laylat al-Qadr
20      14  #Redirect Comédie-Française
21      13  In VDCs there is one elected chief, usually elected with over an 80% majority   From each ward, there is also...
22      13  After the election, the composition of the council was
23      13  #redirect PrimeCo
24      13  #redirect Knnillssonn
25      13  #redirect Fourth wall
26      12  It was listed on the National Register of Historic Places in 1989
27      12  #redirect Orders of magnitude (area)
28      11  [tpl]italic title[/tpl]
```

c) Describe how you built the query from the clue. For example, are you using all the words in the clue or a subset? If the latter, what is the best algorithm for selecting the subset of words from the clue?

For the query part, I use the BooleanQuery from Lucene. BooleanQuery can compare multiple queries to search from the index. It has settings should, must, and must not. Here, I give the two queries with should. And I used all the words in the clue. The reason is that most of the documents are very large. For example, Michal Jackson has more than 6000 words. And all the clues are relatively short, and I have to use all the words to have better hits.

d) Are you using the category of the question?

Yes. If I am not using the category, the total hits at rank 1 are worse. I put the results as follows:



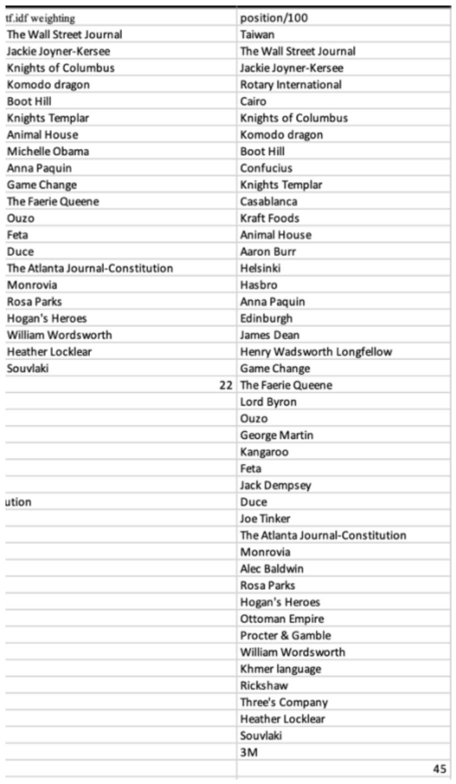
bag of words	without category
The Wall Street Journal	The Wall Street Journal
Jackie Joyner-Kersee	Jackie Joyner-Kersee
Knights of Columbus	Knights of Columbus
Komodo dragon	Komodo dragon
Boot Hill	Boot Hill
Animal House	Animal House
Aaron Burr	Aaron Burr
Anna Paquin	Game Change
Game Change	Kangaroo
Ouzo	Jack Dempsey
Kangaroo	Duce
Jack Dempsey	Joe Tinker
Duce	The Atlanta Journal-Constitution
Joe Tinker	Alec Baldwin
The Atlanta Journal-Constituti	Hogan's Heroes
Alec Baldwin	Souvlaki
Hogan's Heroes	17
Souvlaki	
19	

2) (25 pts) Measuring performance: Measure the performance of your Jeopardy system, using one of the metrics discussed in class, e.g., precision at 1 (P@1), normalized discounted cumulative gain (NDCG), or mean reciprocal rank (MRR). Note: not all the above metrics are relevant here! Justify your choice, and then report performance using the metric of your choice.

I choose the precision at 1(P@1) in this project as the measuring performance method. The reason is that the system I wrote is a QA system; people who use the QA system are looking for the first place answer. No one cared about the later answers.
I implement this method from scratch. Since Lucene has the functions topDocs, we can use code,topDocs.totalHits, to check the total number of hits from the index. And we can store all the results. Here, I just used the rank 0 results, the first place, and compared it with the correct answer from the question.txt.

3) (25 pts) Changing the scoring function: Replace the scoring function in your system with another. For example, by default Lucene uses a probabilistic scoring function (BM25). You can replace this default choice with cosine similarity based on tf.idf weighting. How does this change impact the performance of your system?

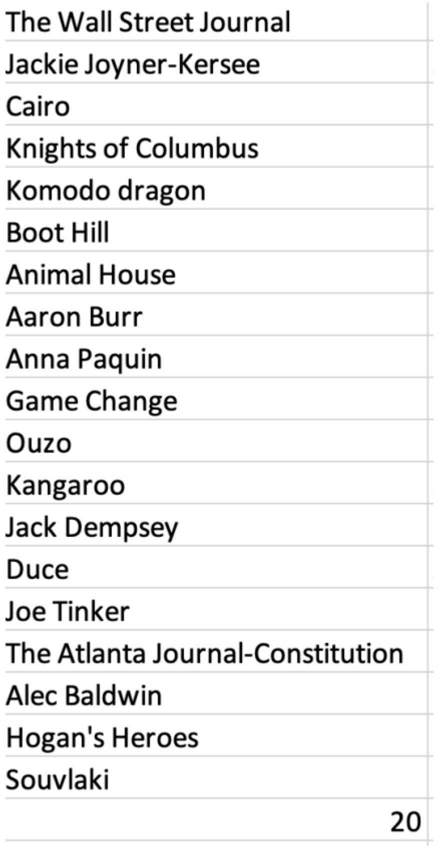
Changing the scoring function from BM25 to tf.idf weighting worsens my system. First, It can not predict anything correctly in the first place. And I rose to the rank 100; it only hit 22, which is much worse than before.



tf.idf weighting	position/100
The Wall Street Journal	Taiwan
Jackie Joyner-Kersee	The Wall Street Journal
Knights of Columbus	Jackie Joyner-Kersee
Komodo dragon	Rotary International
Boot Hill	Cairo
Knights Templar	Knights of Columbus
Animal House	Komodo dragon
Michelle Obama	Boot Hill
Anna Paquin	Confucius
Game Change	Knights Templar
The Faerie Queene	Casablanca
Ouzo	Kraft Foods
Feta	Animal House
Duce	Aaron Burr
The Atlanta Journal-Constitution	Helsinki
Monrovia	Hasbro
Rosa Parks	Anna Paquin
Hogan's Heroes	Edinburgh
William Wordsworth	James Dean
Heather Locklear	Henry Wadsworth Longfellow
Souvlaki	Game Change
22	The Faerie Queene
	Lord Byron
	Ouzo
	George Martin
	Kangaroo
	Feta
	Jack Dempsey
ution	Duce
	Joe Tinker
	The Atlanta Journal-Constitution
	Monrovia
	Alec Baldwin
	Rosa Parks
	Hogan's Heroes
	Ottoman Empire
	Procter & Gamble
	William Wordsworth
	Khmer language
	Rickshaw
	Three's Company
	Heather Locklear
	Souvlaki
	3M
	45

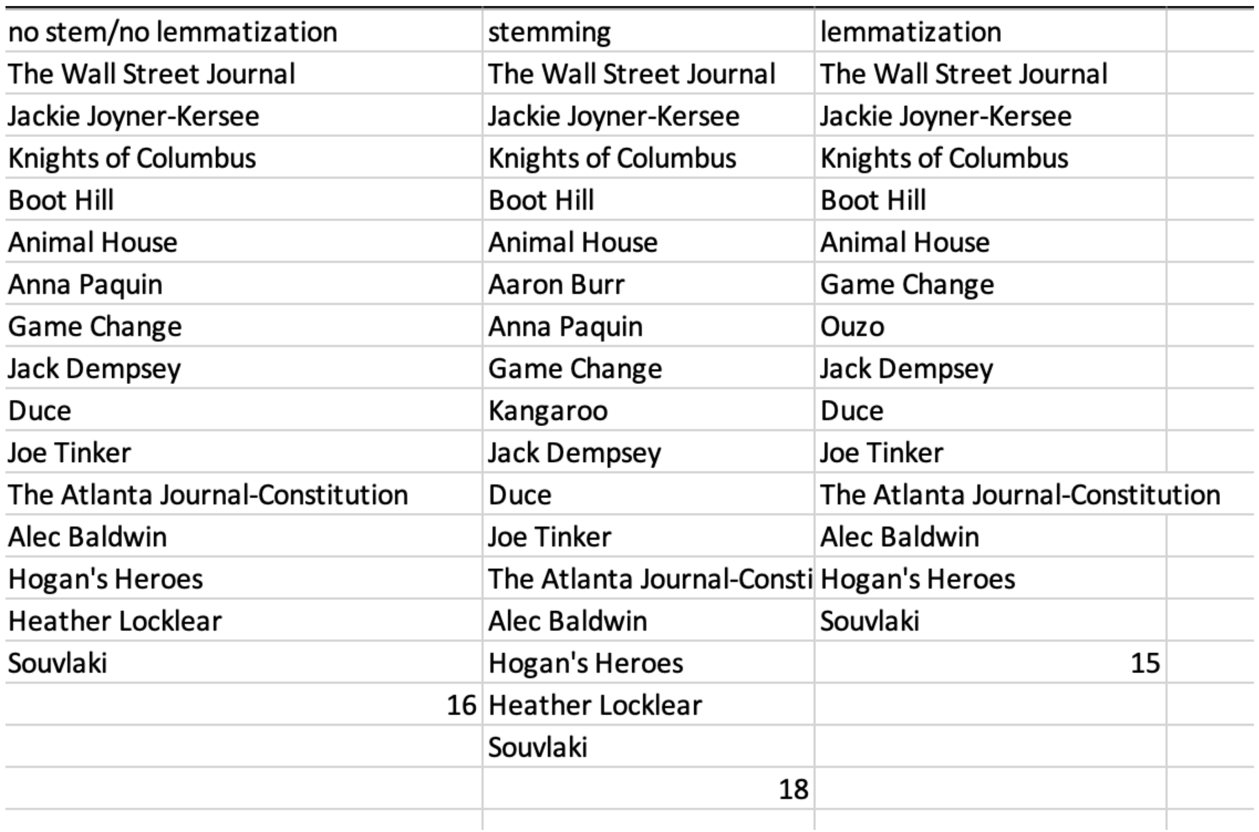
4) a) (50 pts) Error analysis: Perform an error analysis of your best system. How many questions were answered correctly/incorrectly? Why do you think the correct questions can be answered by such a simple system? What problems do you observe for the questions answered incorrectly? Try to group the errors into a few classes and discuss them.

The best results are that my QA system can answer the following questions, which are 20 questions from 100 questions. Before discussing these, I firstly wrote a function to find the document from the wiki-dataset. And for the common of most correct answers is that clue has the particular terms that help choose the correct indexes. For example, Jackie Joyner-Kersee has three terms basketball, Olympics, and UCLA from the clue and category. However, those questions lack such terms as the particular terms contained by the clue, category, or the wiki-dataset for the wrong answers. And one more thing, the documents like Michal Jackson, which has more than 6000 words. The clue is "Man in the Mirror," which shows almost the end of the document, which is why it is not ranked 1.



The Wall Street Journal
Jackie Joyner-Kersee
Cairo
Knights of Columbus
Komodo dragon
Boot Hill
Animal House
Aaron Burr
Anna Paquin
Game Change
Ouzo
Kangaroo
Jack Dempsey
Duce
Joe Tinker
The Atlanta Journal-Constitution
Alec Baldwin
Hogan's Heroes
Souvlaki
20

b) Lastly, what is the impact of stemming and lemmatization on your system? That is, what is your best configuration: (a) no stemming or lemmatization; (b) stemming, or (c) lemmatization? Why?
a)no stemming or lemmatization
Using a simple analyzer is the same as the effect of no stemming or lemmatization.
I created a customed Analyzer by adding different filters to stemming or lemmatization.
b)stemming c)lemmatization
From the picture following, I found that stemming is the most impact on my system. But, the best configuration is with stem and lemmatization.



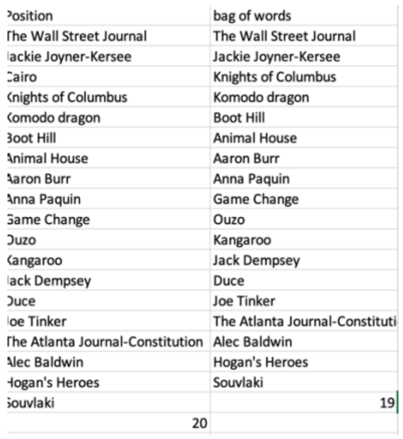
no stem/no lemmatization	stemming	lemmatization
The Wall Street Journal	The Wall Street Journal	The Wall Street Journal
Jackie Joyner-Kersee	Jackie Joyner-Kersee	Jackie Joyner-Kersee
Knights of Columbus	Knights of Columbus	Knights of Columbus
Boot Hill	Boot Hill	Boot Hill
Animal House	Animal House	Animal House
Anna Paquin	Aaron Burr	Game Change
Game Change	Anna Paquin	Ouzo
Jack Dempsey	Game Change	Jack Dempsey
Duce	Kangaroo	Duce
Joe Tinker	Jack Dempsey	Joe Tinker
The Atlanta Journal-Constitution	Duce	The Atlanta Journal-Constitution
Alec Baldwin	Joe Tinker	Alec Baldwin
Hogan's Heroes	The Atlanta Journal-Consti	Hogan's Heroes
Heather Locklear	Alec Baldwin	Souvlaki
Souvlaki	Hogan's Heroes	15
16	Heather Locklear	
	Souvlaki	
	18	

5) (50 pts) Improving retrieval (GRAD STUDENTS ONLY): Improve the above standard IR system using natural language processing and/or machine learning. For this task you have more freedom in choosing a solution and I encourage you to use your imagination. For example, you could implement a positional index instead of a bag of words; you could use a parser to extract syntactic dependencies and index these dependencies rather than (or in addition to) words; you could use supervised learning to implement a reranking system that re-orders the top 10 (say) pages returned by the original IR system (hopefully bringing the correct answer to the top).

In this project, first, I tried to "enlarge" the queries and categories by searching every term's (filtered by the stopword_filter) synonyms. However, the results became worse, and the reason is that I may create different queries for searching, leading to the wrong answers.

Later, I used the positional index instead of a bag of words and proximity Searches. Lucene supports the proximity Searches, which use the "~" symbol at the end of a phrase. (For example, The format is that "Jakarta apache "~10, which is to search for an "apache" and "Jakarta" within ten words of each other.) Moreover, getting a window of words around a match can be helpful in a Question Answering System. However, the default of Lucene fields does not contain the position information. You need to use the field methods and set index options to the doc&freq&position information.

The result is a little better than the bag of words. Total hits in rank 1 are 20, which achieve 20% accuracy at position 1, and the previous is 19.



Position	bag of words
The Wall Street Journal	The Wall Street Journal
Jackie Joyner-Kersee	Jackie Joyner-Kersee
Cairo	Knights of Columbus
Knights of Columbus	Komodo dragon
Komodo dragon	Boot Hill
Boot Hill	Animal House
Animal House	Aaron Burr
Aaron Burr	Anna Paquin
Anna Paquin	Game Change
Game Change	Ouzo
Ouzo	Kangaroo
Kangaroo	Jack Dempsey
Jack Dempsey	Duce
Duce	Joe Tinker
Joe Tinker	The Atlanta Journal-Constituti
The Atlanta Journal-Constitution	Alec Baldwin
Alec Baldwin	Hogan's Heroes
Hogan's Heroes	Souvlaki
Souvlaki	19
20	