Document analysis
Lab and assignment 1: Information retrieval
u5782545 Zhaolian Zhou

Q2 Appropriate TREC measures

1.For measuring a search system for government web sites, I think use num_rel_ret (Total number of relevant documents retrieved over all queries) and num_rel (Total number of relevant documents over all queries) are appropriate.

The reason is these two elements can indicate the recall of the search result, which I would like to choose as my main measure element.

Recall = P(relevant items retrieved / relevant items) [1]

We will consider the MAP as another element to measure the search system as well.

2. Usually precision is more important than recall in IR systems in general-purpose search engine. As described in Deeppak Garg and Deepika Sharma's article[2], these search engines are specialized on an area and provides huge collection of documents related to that specific area. Government web sites search system, however, can be considered as a specialized search engines. By using such search engines, searcher usually need to know all the relevant information on a topic[3]. Recall can be more important in this case.

So we choose recall as the main element to measuring the search system.

Q3 Indexing and querying

1. With 100 files retrieved, of the overall 108 relevant documents, 57 relevant documents are retrieved. For most of the queries, the recall are good, except some queries especially as Query 32 and Query 37.

2. For query 32, there are only 7 relevant retrieved with 30 relevant files. And for query 37, no relevant documents are retrieved. Similarly, for query 36, only 1 out of the 6 relevant files are retrieved; query 38, 1 out of 4 and query 43, 2 out of 6.

Q4 Improving performance

To improving lucene's performance on this test collection, I would try the following methods.

1. Lower case all the terms and documents, which will improve the matchment of same word under different cases.

2. Stemming. By stemming the words both in terms and documents, will increase the performance on matching the words talk about the same thing but with different forms.

3. Remove the stop words. Stop words are words which are filtered out before or after processing of natural language data.[4] However, as stop word removal aims at "identifyng noisy terms that may hurt precision" [5], it may have no help in promoting the recall.

4. Changing the boolean of terms. We noticed that in the gov.topic file, query 16, 19, and 38 contains the boolean keywords "and", as we want to improve the recall of the search engine, we may want to use "OR" instead of "AND" to ensure as many as files are returned.

5. Wild cards. Noticed that there are several "es" in the query, replace that into "*".

Q5 Modifications to Lucene

I campared the StandardAnalyzer and EnglishAnalyzer given in the lucene library. StandardAnalyzer use the StandardTokenizer, StandardFilter, LowercaseFilter and StopFilter to complete both of my aims.

EnglishAnalyzer rolls in an EnglishPossesiveFilter, KeywordMarkerFilter and PorterStemFilter. EnglishAnalyzer rolls in some English stemming enhancements, which should work well for our English text documents. [6]

1. In the AddDoc function of DocAdder class, by change all the line read into buffer to lower case before indexing. And also change each query into lower case.

Sections of code:

SimpleSearchRanker.java: (lower casing all the queries)

```
while(scan2.hasNextLine()){
    String topic = scan2.nextLine().toLowerCase();
    String[] topic_splited = topic.split(" ", 2);
    topic_body2.add(topic_splited);
}
```

DocAdder.java: (lower casing all the files)

```
//Change the line in file to lower case.
while ((line = br.readLine()) != null) {
    if (first_line == null)
        first_line = line.toLowerCase();
    content.append(line.toLowerCase() + "\n");
}
doc.add(new StoredField("PATH", f.getPath()));
doc.add(new TextField("FIRST_LINE", first_line, Field.Store.YES));
doc.add(new TextField("CONTENT", content.toString(), Field.Store.YES));
```

And also, I try to use the StandardAnalyzer which can do lowercase and stop words filter to gain the similar aim.

Sections of code:

FileIndexBuilder.java: using the StandardAnalyzer.

```
public FileIndexBuilder(String index_path) {

    // Specify the analyzer for tokenizing text.
    // The same analyzer should be used for indexing and searching
    // See the lucene "analysers-common" library for some more options;
    // the .jar file is included in the lib/ directory, and there is
    // good documentation online
    // Use the standard analyzer to convert all terms to lower case.

    //_analyzer = new SimpleAnalyzer();
    _analyzer = new StandardAnalyzer();
```

2. To do the stemming, use EnglishAnalyzer instead of SimpleAnalyzer.

Sections of code:

FileIndexBuilder.java: using the EnglishAnalyzer.

```
public FileIndexBuilder(String index_path) {

    // Specify the analyzer for tokenizing text.
    // The same analyzer should be used for indexing and searching
    // See the lucene "analysers-common" library for some more options;
    // the .jar file is included in the lib/ directory, and there is
    // good documentation online
    // Use the standard analyzer to convert all terms to lower case.

    //_analyzer = new SimpleAnalyzer();
    _analyzer = new EnglishAnalyzer();
```

3. Replace the substring of “and” to “ ” to ensure the search logic between each words in the query are “OR” relationship.

SimpleSearchRanker.java:

```
Scanner scan2 = new Scanner(topic_file2);
ArrayList<String[]> topic_body2 = new ArrayList<String[]>();
while(scan2.hasNextLine()){
    String topic = scan2.nextLine().toLowerCase();
    String[] topic_splited = topic.split(" ", 2);
    topic_body2.add(topic_splited);
}
PrintStream myout2 = new PrintStream(retrieve_file2);
for(String[] query: topic_body2){
    //Replace "AND" to "" to implicit OR judgement.
    if(query[1].toLowerCase().contains("and")){
        query[1]=query[1].toLowerCase().replace("and", "");
    }
    System.out.println(query[1]);
    r2.doSearch(query, 100, myout2);
}
scan2.close();
```

4. Wild cards.

Sections of code:

SimpleSearchRanker.java:

```
for(String[] query: topic_body2){
    //Replace "AND" to "" to implicit OR judgement.
    if(query[1].toLowerCase().contains("and")){
        query[1]=query[1].toLowerCase().replace("and", "");
    }
    //Wild Cards
    if(query[1].toLowerCase().contains("es")){
        query[1] = query[1].toLowerCase().replace("es", "*");
    }
    r2.doSearch(query, 100, myout2);
}
scan2.close();
```

Q6 Rerunning the modified Lucene

1. After preporcessing all the files and queries by lower case the strings, the result seems don't change. In my point of view, that's because the SimpleAnalyzer can lowercase them as well.[7] No change will happen if we do that again.

After change from SimpleAnalyzer to StandardAnalyzer, the MAP (Mean Average Precision) has improved from 0.2061 to 0.2622. However, the number of relevant documents retrieved decrease from 57 to 56, which means the recall of the system decreased. This situation meet the expectations described in Question 4, item 3.

2. After change from SimpleAnalyzer to EnglishAnalyzer, num_rel_ret element increased from 57 to 70, which means the recall of the system increased from 52.77% to 64.81%. Which is pretty signaficant improvement. And also, the MAP increased from 0.2061 to 0.2474. Take topic 37 as example, under SimpleAnalyzer, there is no relevant documents retrieved at all. After change to Englishanalyzer, as contrast , all of the two relevant documents are retrieved. Same of topic 47, num_rel_ret increased to 6 from 2.

3. To replace the "and" keyword in the topic to " " seems don't works very well. No changes occurred after such operation.

4. Wild cards seems doesn't work as well, which I guess after apply the wild cards, it negative the performance on stemming of EnglishAnalyzer.

References

[1] *C. Manning, P. Raghavan and H. Schütze, Introduction to information retrieval. New York: Cambridge University Press, 2008.*

[2] D. Garg and D. Sharma, "Information Retrieval on the Web and its Evaluation", *International Journal of Computer Applications*, vol. 40, no. 3, pp. 26-31, 2012.

[3] D. Inkpen "Information Retrieval on the Internet"

[4]"Stop words", *Wikipedia*, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Stop_words. [Accessed: 10- Aug- 2016]

[5]R. Blanco and A. Barreiro, "Static Pruning of Terms in Inverted Files", in *Advances in Information Retrieval: 29th European Conference on IR Research*, Rome, Italy, 2007.

[6]W. EnglishAnalyzer?, "What's the difference between Lucene StandardAnalyzer and EnglishAnalyzer?", *Stackoverflow.com*, 2016. [Online]. Available: http://stackoverflow.com/questions/17011854/whats-the-difference-between-lucene-standardanalyzer-and-englishanalyzer. [Accessed: 09- Aug- 2016].

[7] "Lucene - Analysis", *www.tutorialspoint.com*, 2016. [Online]. Available: http://www.tutorialspoint.com/lucene/lucene_analysis.htm. [Accessed: 09- Aug- 2016].