COMP6490 Assignment 1

By Guyue HU
u5608260

Q2

I think **ircl_prn** (Interpolated Recall - Precision Averages) might be one of the trec_eval's measures appropriate for measuring a search system for government web sites.

There are several reasons:

1. Because we don't know if the search system is used as a web surfer or a professional searcher, the two important metrics "precision" and "recall" should both be considered in an appropriate measure. The **ircl_prn** "measures precision (percent of retrieved docs that are relevant) at various recall levels (after a certain percentage of all the relevant docs for that query have been retrieved)[1]", which considered both "precision" and "recall" in one measure.
2. **ircl_prn** counts for the highest precision for recall above a certain point, which fits natural human user habits: willing to look at a few more documents if can get higher relevant percentage.
3. **ircl_prn** is not a single number, but a recall-precision graph, which can provide more insights for a designer to look into and improve the design.

Q3

Generally speaking, lucene does not do very well in the **ircl_prn** measure. At first I let lucene return the 5 top ranked result. The **ircl_prn** measure is very low:

ircl_prn.0.00	all	0.2898
ircl_prn.0.10	all	0.2898
ircl_prn.0.20	all	0.2833
ircl_prn.0.30	all	0.1866
ircl_prn.0.40	all	0.1785
ircl_prn.0.50	all	0.1785
ircl_prn.0.60	all	0.1032
ircl_prn.0.70	all	0.1032
ircl_prn.0.80	all	0.1032
ircl_prn.0.90	all	0.1032
ircl_prn.1.00	all	0.1032

1. Cited from the "help" documents of eval_trec.8.1

Then I tried to let lucene return 20 top ranked result, the precision is increased, but still not satisfactory. Even at the lowest recall level 0.00, the precision is no more than 0.33, which means for every 3 returned results only 1 is relevant. The performance is not acceptable.

ircl_prn.0.00	all	0.3189
ircl_prn.0.10	all	0.3149
ircl_prn.0.20	all	0.3084
ircl_prn.0.30	all	0.2291
ircl_prn.0.40	all	0.2211
ircl_prn.0.50	all	0.2160
ircl_prn.0.60	all	0.1280
ircl_prn.0.70	all	0.1140
ircl_prn.0.80	all	0.1140
ircl_prn.0.90	all	0.1140
ircl_prn.1.00	all	0.1140

There are 2 topics lucene did quite well, including:
18 Shipwrecks
42 homelessness
Both have constant 1.0 precision for recall from 0.0 to 1.0.

There are several topics lucene did badly, including:
1 mining gold silver coal
6 physical therapists
9 genealogy searches
22 Veteran's Benefits
28 Early Childhood Education
31 deafness in children
32 wildlife conservation
35 arctic exploration
37 coin collecting
39 National Public RadioVTV
41 Electric Automobiles
All have 0.0 precision for recall from 0.0 to 1.0.

Q4

By analysing the topics which lucene did well and not so well, I conclude the following:

1. Lucene did well in “single word” topics.
2. Lucene could do well with topics starting with both capital letter (“***Shipwrecks***”) and lower case letter (“***homelessness*** ”)
3. All the topics Lucene did badly contains more than 1 words.
4. Most of the topics Lucene did badly contains some kind of inflexion. For example: ***therapist -> therapists, collect -> collecting, etc.***

In conclusion, the number of words in a topic and the inflexion are the major factors inflecting the performance of lucene.

The changes I plan to make to this basic lucene including two aspects:

1. **increase the number of returned results.** For a document collection including 34,000 documents and complicated topics, maybe the returned 5 or 20 top ranked result are not large enough to judge the true performance of a search system.

2. **Stemming words both in the index and queries.** As analyzed above, inflexion is one of the major reasons responsible for poor performance. Stemming is reducing inflected words to their word stem. For example **ducks -> duck**. Through stemming, some originally not matched words may be able to match. Stemming may be especially useful to those badly performed topics with 0 precision (may result from the system can not find matching documents).

Q5

(1)

I first increased the number of returned result to each query to the top 50 ranked results, by changing the following section of code (when calling r.doSearch()).

```
for (int i = 0; i < topics.size(); i++) {
        String t = topics.get(i);
        t = t.replace("/", " OR ");
        String id = idx.get(i);
        System.out.println(t);
        r.doSearch(id, t, 50, writer);
}
```

(2)

I tried to stem words in both the indexer and the topics using ***PorterStemmer***. At the same time I converted all the capital letters into lowercase letters. Both are done in a class MyStemmer.java.

The basic idea is: First tokenize a string into words. Then for each words one by one, use a PoterStemmer instance stem to convert a word into its stem. Finally combine all the words into a stemmed string.

Codes are shown as below:

```
package search;

import java.util.StringTokenizer;
```

```
import org.tartarus.snowball.ext.PorterStemmer;

public class MyStemmer {
	public static String Stem(String content) {
		String stemed_content = "";
		String word;
		StringTokenizer st = new StringTokenizer(content);
		PorterStemmer stem = new PorterStemmer();

		while (st.hasMoreTokens()) {
			word = st.nextToken();
			stem.setCurrent(word);
			stem.stem();
			word = stem.getCurrent();
			stemed_content = stemed_content + word.toLowerCase() + " ";
			//System.out.println(stemed_content);
		}

		return stemed_content;
	}

	public static void main(String[] args) {
	    System.out.println(Stem("Mining mined mine"));
	}
}
```

I called ***MyStemmer*** in DocAdder.java with codes:

```
first_line = MyStemmer.Stem(first_line);
doc.add(new TextField("FIRST_LINE", first_line, Field.Store.YES));
String cont = MyStemmer.Stem(content.toString());
doc.add(new TextField("CONTENT", cont, Field.Store.YES));
```

And in SimpleSearchRanker.java with codes:

```
for (int i = 0; i < topics.size(); i++) {
	String t = topics.get(i);
	t = t.replace("/", " OR ");
	t = MyStemmer.Stem(t);
	String id = idx.get(i);
	System.out.println(t);
	r.doSearch(id, t, 50, writer);
}
```

Q6

(1)

After increasing the number of returned results to 50, the **ircl_prn** measure did improved compare to the measure to 20 results per topic. As shown below:

```
ircl_prn.0.00	all	0.3288
```

ircl_prn.0.10	all	0.3288
ircl_prn.0.20	all	0.3166
ircl_prn.0.30	all	0.2398
ircl_prn.0.40	all	0.2318
ircl_prn.0.50	all	0.2304
ircl_prn.0.60	all	0.1470
ircl_prn.0.70	all	0.1356
ircl_prn.0.80	all	0.1355
ircl_prn.0.90	all	0.1264
ircl_prn.1.00	all	0.1264

However, the improvement is not obviously.

(2)

After the stemming modification, the overall performance of lucene did not improve. the **ircl_prn** measure:

ircl_prn.0.00	all	0.3161
ircl_prn.0.10	all	0.3155
ircl_prn.0.20	all	0.3111
ircl_prn.0.30	all	0.2530
ircl_prn.0.40	all	0.2476
ircl_prn.0.50	all	0.2303
ircl_prn.0.60	all	0.1463
ircl_prn.0.70	all	0.1258
ircl_prn.0.80	all	0.1248
ircl_prn.0.90	all	0.1248
ircl_prn.1.00	all	0.1248

However, for those topics which the original version did extremely badly (as summarized in Q3), including topic number **1, 6, 9, 22, 28, 31, 32, 35, 37, 39, 41**, the modification did improved the performance on most of them. The topics upon which lucene's performance improved includes:
1 mining gold silver coal
9 genealogy searches
22 Veteran's Benefits
28 Early Childhood Education
32 wildlife conservation
35 arctic exploration
37 coin collecting
41 Electric Automobiles
All of these topics have a precision more than 0.0 after the modification of stemming.

For the three which did not improve after stemming:
6 physical therapists
31 deafness in children
39 National Public RadioVTV
all includes words that ***PorterStemmer*** does not stem properly: *therapist, children, TV*

In conclusion, my first idea "**increase the number of returned results**" did leads to improvements of performance overall. But one thing should be noticed is that the improvement is obvious when the number is increased from a very small amount to a moderate amount (return number 5 -> 20, overall performance 0.2898 -> 0.3189), but from a moderate amount to a large amount, the improvement is less obvious (return number 20 -> 50, overall performance 0.3189 -> 0.3288). The conclusion about this idea is, in order to get a more reasonable result from trec_eval measure, the system should return moderate amount of top ranked results comparable to the document collection size.

My second idea "**stemming words**" is focusing on the topics that the original system did especially badly, and the performance of those topics did improved by the modified system. However, the overall performance did not improve, performance on other topics may be lowered by the modification. This means stemming solved certain problems, but at the same time, it introduced new problems to the system. But generally speaking, I think stemming is a good idea, because it at least partially solved the problem it supposed to solve. For the new problems introduced by the stemming, we may need more analysis and other modifications.

- The End of Assignment 1 -