Name: Zhoucheng Li

Andrew ID: zhouchel

Homework 3

1 Error Analysis

After running 20 queries using the default cosine similarity, only the query (qid = 10) ranked the relevant document the top place. The left 19 queries failed to retrieve the relevant document as the top rank document.

An analysis was performed on these 19 cases. From a semantic perspective, all errors are due to failures of understanding the problem. Since we are using similarity to retrieve the best answer, these errors could be split into several categories:

Error Type	Query ID	Total Number
Redundant information	1, 2, 3, 4, 7, 8, 12, 14, 15, 17, 18, 19, 20	13
Stemming/ Lemmatization related	4, 5, 6, 7, 8, 11, 12, 16, 17	9
Phrase related	9, 20	2
Synonym related	13, 19	2

Redundant information means that the relevant answer has redundant information that is not asked by the query. Since the cosine similarity relays on the overlap of query vector and answer vector, redundant information may cause the similarity decrease.

Meanwhile, long documents may have more stop words compared to short documents, stop words may also decrease the cosine similarity. And a tokenizer could solve this issue. But it's still not sufficient to reduce the redundant information.

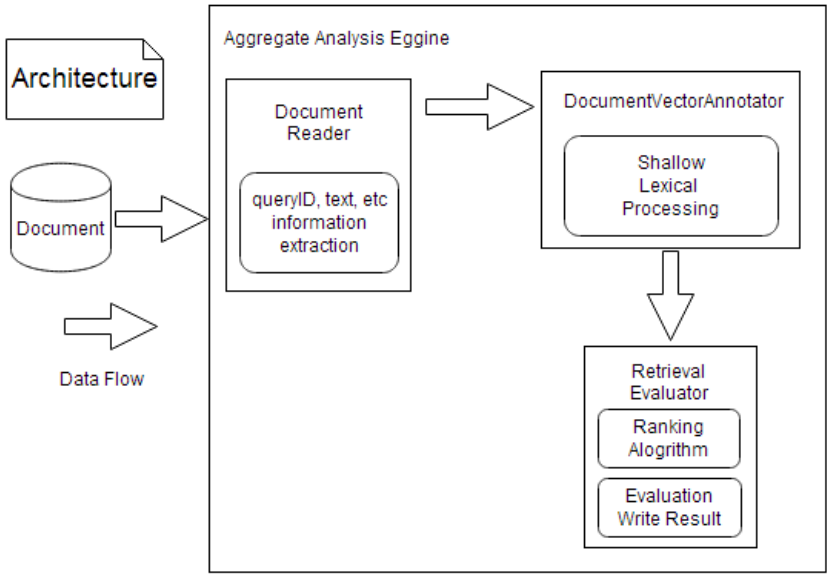
Stemming/ Lemmatization related means the document has some inflected form of the words contained in the query. For example, call vs. called, China vs. China's, 4-minute vs. 4 minute and devil's vs. devil. Or something like become vs. became, bit vs. bite. If these inflected words are all reduced to their root forms, the cosine similarity could be increased. Tokenizer might help improve this problem.

Phrase related means some terms in a query should be treated as a whole term. If partial of the phrase appears in a document, we shouldn't consider this term as "overlapped". For example, in qid = 9, the query asks about "first spaceship", while the top irrelevant answer gives information about "first manned spaceship". But phrases are supported by query operators (NEAR or Window), or by using another distance metric, I didn't put much effort on this case.

Synonym related means the related answer uses synonyms or words that have similar meaning of the words in a query. For example, deep vs. depth (qid = 13) and N.J. vs.New Jewsey (qid = 19). Again, synonyms need support from query operator (SYNONYM), so I ignore this case.

Most cases have one error type while some have multiple error types.

2 System Architecture



Document Reader: The document reader is the same as that in Task 1.

Document Vector Annotator: Apparently, to reduce the errors, the most effective way is to adopt a more advanced Tokenizer before computing the cosine similarity because tokenizer could cover most error cases in this problem. So a more advanced tokenizer is used to do some shallow lexical processing like upper-case to lower-case convert, stop words removal and stemming in *DocumentVectorAnnotator*.

I adopt Lucene 4.3.0 since newest version of Lucene is not compatible with JDK 1.6. Lucene has provided a powerful configurable stop words analyzer to all three kinds of processing methods mentioned above. And I configured to use Potter Stem Filter to do stemming part.

Retrieval Evaluator (casConsumer): The casConsumer would fetch all tokens and reconstruct the global dictionary. Then use the chosen algorithm to rank documents, calculate the MRR and write results to file system.

Type Systems: The type systems are identical to the original ones.

To simply the coding, I created another Doc class which implements Comparable interface and is only used in casConsuemr. This class contains all information for a ranking algorithm.

```
public class Doc implements Comparable<Doc>{
  public HashMap<String, Integer> docVector;
  public int queryId;
  public int relevance;
  public double score;
  public String text;
  public int docLength;
```

3 Ranking Algorithms

3.1 Baseline

Simple tokenizer + Cosine Similarity

3.2 Extra algorithms

I used 3 extra ranking algorithms in this homework. (Github)

1. ***Lucene tokenizer + Cosine Similarity***

   Except for the tokenizer, the cosine similarity calculation is identical to that in baseline algorithm.

2. ***Lucene tokenizer + lnt.ltc***

Term Frequency (tf) and Inverse Document Frequency (idf) are valuable information. tf represents the importance of the term in a document. And terms that occur in many documents in the corpus are less discriminative compared to rare terms.

Given a term, idf is often calculated as:

$$idf = \log\left(\frac{N+1}{df}\right)$$

where N is the corpus size and df is the document frequency (under all 20 queries) that contains this term.

The lnc.ltc algorithm still uses cosine similarity metric, but the tf is normalized and idf is used to add more weight to rare terms. It can be described as:

$$\frac{\sum d_i \cdot q_i}{\sqrt{\sum d_i^2} \cdot \sqrt{\sum q_i^2}} = \frac{\sum(\log(tf)+1)\cdot(\log(qtf)+1)\cdot idf}{\sqrt{\sum(\log(tf)+1)^2}\cdot\sqrt{\sum((\log(qtf)+1)\cdot idf)^2}}$$

where this equation should only apply to terms that are in both query and document.

3. ***Lucene tokenizer + BM25***

The Okapi BMxx model can be written as:

$$\text{BMxx} = \sum_{t\in q}\left(\log\frac{N-df_t+0.5}{df_t+0.5}\right)\frac{tf_t}{tf_t+k_1\left((1-b)+b\frac{\text{doclen}}{\text{avg\_doclen}}\right)}\frac{(k_3+1)qtf_t}{k_3+qtf_t}$$

The first pair of brackets represents “idf weight”. The middle part of the equation represents “tf weight” while the last part represents “user weight”.

BMxx indicates different parameter settings. For BM25, we usually select $k_1 = 1.2$, $b = 0.75$ and$k_3 = 0 - 1000$.

4 Experiments

The tested results are shown below.

Tokenizer	Metric	MRR	Issues resolved (except qID=10)
Basic tokenizer	Cosine Sililarity	0.4375	-
Lucene tokenizer	Cosine Sililarity	0.6542	3,4,5,9,11,14,16,20
Lucene tokenizer	lnc.ltc	0.6458	3,4,5,9,11,14,16
Lucene tokenizer	BM25 (k = 1.2, b = 0.75)	0.6542	3,4,5,9,11,14,16,20

From the above table, we can find out that what has been illustrated in error analysis, adopting tokenizers could help reduce errors in ***Redundant Information*** (especially for stop words related cases) and ***Stemming/Lemmatization related***. However, even I tried some more advanced ranking algorithms like lnc.ltc and BM25, the MRR is not improved compared to Lucene tokenizer+Cosine Similarity. I think this is because the corpus size is too small and lnc.ltc/BM25 are still similarity based ranking algorithm. This kind of algorithms might doing well in search engine domains but are not so good in Question Answering area.

One interesting thing is that for queryID = 9, the error type belongs to phrase related error. However, this error is solved in all 3 algorithms after using Lucene tokenizer. I found this is because after tokenizing, the top two documents have the same document length and only two terms are different. So they have the same ranking score. Since in original input file, for each query, all relevant documents appear before irrelevant documents. Once we meet a tie, we will use the relevant document. So actually, the MRR is this homework is not reliable since there is a tie.

"luna 2 first spacecraft reach surface moon"
"eagle first manned spacecraft reach surface moon"

More tests on BMxx using different parameters combinations. Again, parameters can only affect rankings of small portion of the documents. I cannot see much differences in MRRs.

Fix k = 1.2

b	0.20	0.40	0.60	0.75	0.90
MRR	0.65	0.67	0.66	0.65	0.65

Fix b = 0.6

k	0.5	0.75	1	1.5	2
MRR	0.6625	0.6625	0.6625	0.6625	0.6542

To sum up, since we only have small amount of test documents for each query, the MRRs of different algorithms are not so reliable and we cannot differentiate the pros and cons of each algorithm. Another conclusion is that tokenizer is the most affective part in whole system.