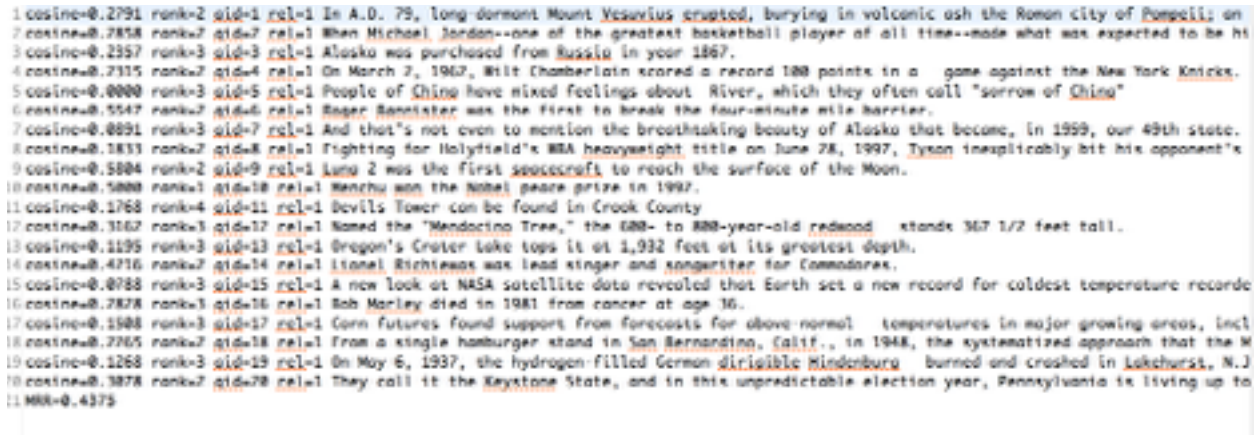
Andrew id: qiqis
October 18, 2014

hw3-report

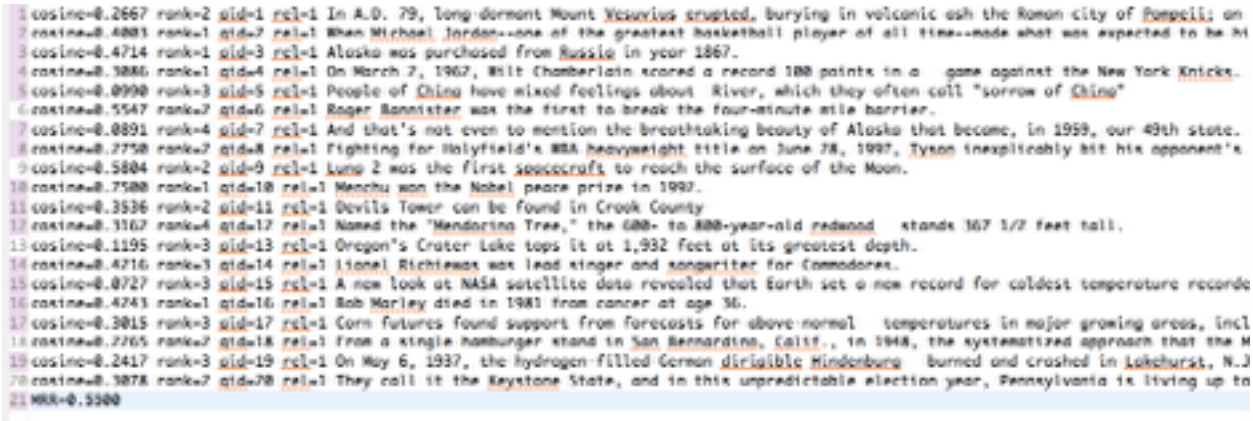
Error Analysis

The following picture is the result by using default tokenization algorithm, cosine similarity method without stemming algorithms. MRR is 0.4375 according to these algorithms.



Stemming Mismatch

As the figure shows below, I implement another way of tokenization by using Stanford Lemmatizer. By using this method, MRR is increased to 0.5500 which means that due to the Stemming ignorance, the original method does not performance well. Therefore, we can say that Stemming mismatch is a error type.



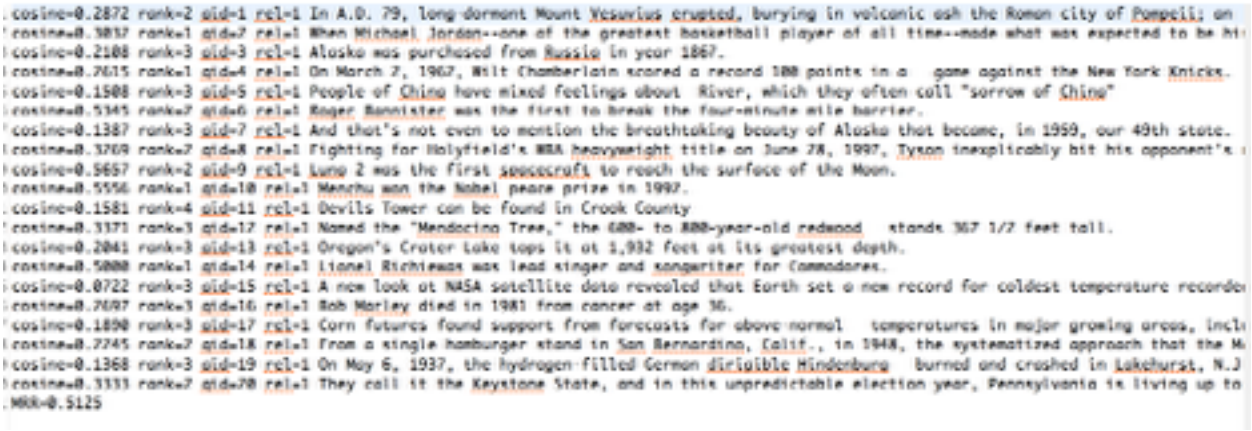
Take the qid=3 for example. The original rank is 3. After the Stemming match, the rank is 1. The original one does not match word “purchase” with “purchased” which affects its performance.

```
qid=3 rel=99      In which year did a purchase of Alaska happen?
qid=3 rel=1       Alaska was purchased from Russia in year 1867.
```

Therefore, for 5 queries, the most relevant answers were not selected due to Stemming Mismatch.

Tokenization Mistake

As the figure shows below, I implement another way of tokenization by using Stanford Tokenizer. By using this method, MRR is increased to 0.5125 which means that due to the Tokenization mistakes, the original method does not performance well. Therefore, we can say that tokenization mistake is a error type.



Take the qid=2 for example. The original rank is 2. After the Stanford Tokenization algorithm, the rank is 1. The original one does not split word according the punctuation such as “-” which ignores the similarity between Jordan and affects its performance.

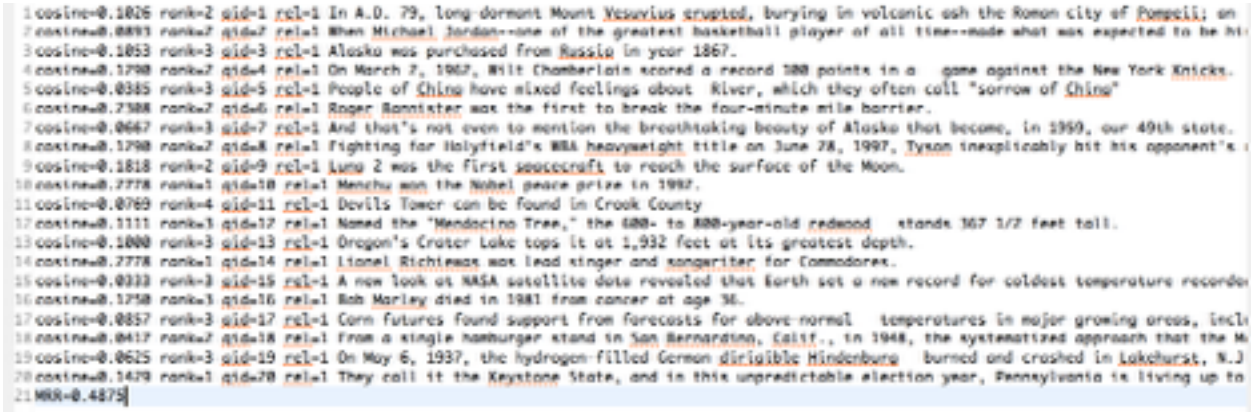
```
qid=2 rel=99      What has been the largest crowd to ever come see
Michael Jordan
qid=2 rel=1       When Michael Jordan--one of the greatest basketball
player of all time--made what was expected to be his last trip to play
in Atlanta last  March, an NBA record 62,046 fans turned out to see
him and the  Bulls.
```

Therefore, for 3 queries, the most relevant answers were not selected due to Tokenization Mistake.

Similarity Measure Flaw

Jaccard Similarity

As the figure shows below, I implement another way of calculating similarity by using Jaccard Similarity way. By using this method, MRR is increased to 0.4875 which means that due to the choice of different similarity calculation way, the original method does not performance well. Therefore, we can say that Similarity Measure Flaw is a error type.



Take the qid=20 for example. The original rank is 2. After the Jaccard Similarity, the rank is 1.

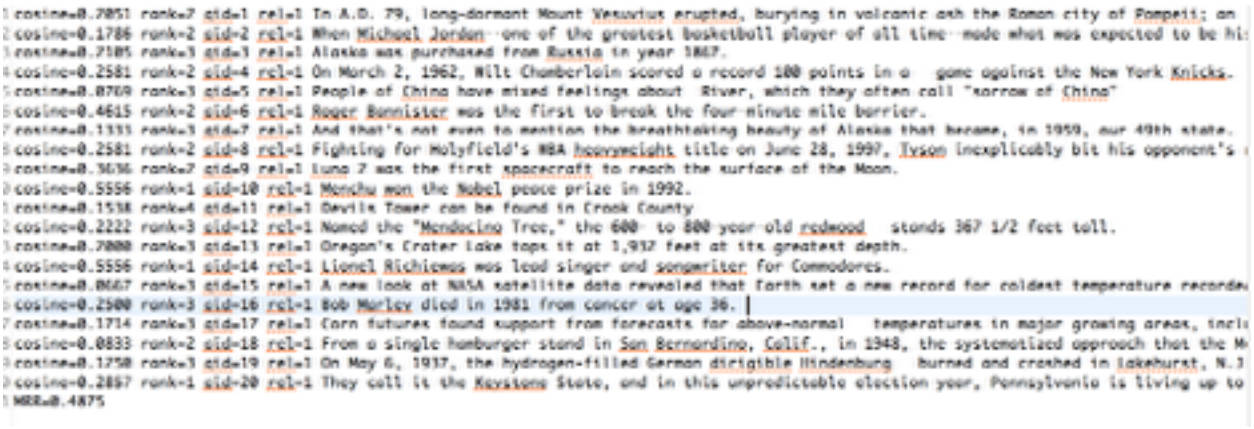
```
qid=20     rel=99     What is the Keystone State?
qid=20     rel=1      They call it the Keystone State, and in this
unpredictable election year, Pennsylvania is living up to its name.
```

Therefore, for 2 queries, the most relevant answers were not selected due to Similarity Measure Flaw.

Dice Coefficient Similarity

As the figure shows below, I implement another way of calculating similarity by using Dice coeffcient Similarity method. By using this method, MRR is increased to 0.4875 which means that due to the choice of different similarity calculation method, the

original method does not performance well. Therefore, we can say that Similarity Measure Flaw is a error type.



Take the qid=20 for example. The original rank is 2. After the Dice coefficient Similarity, the rank is 1.

```
qid=20     rel=99     What is the Keystone State?
qid=20     rel=1      They call it the Keystone State, and in this
unpredictable election year, Pennsylvania is living up to its name.
```

Therefore, for 2 queries, the most relevant answers were not selected due to Similarity Measure Flaw.

Others

For the ties which means that the there are some same similarity which cases ambiguous in rank. In this way, the MRR is affected by the error.

At the same time, different token sequence can express different meanings. However, the cosine similarity does not take this into consideration. So in this way, this can affect the rank of different tokens.

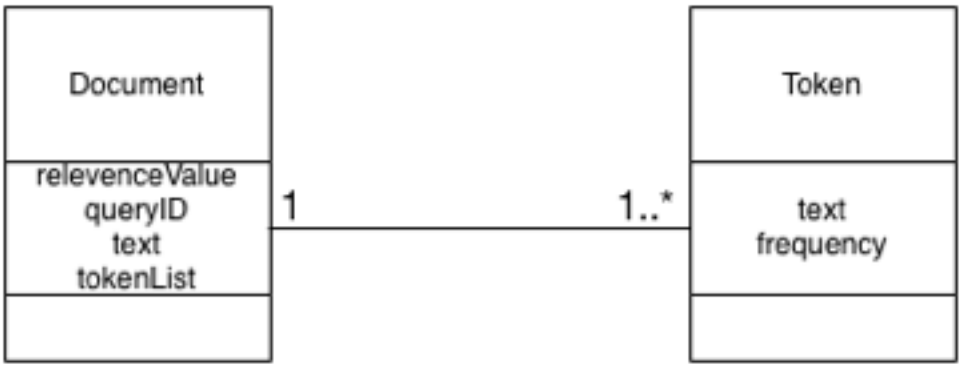
Conclusion

	Error Type	Frequency
1	**Stemming Mismatch**	5
2	**Tokenization Mistake**	3
3	**Similarity Measure Flaw**	2
4	**Ties**	3
5	**Token Sequence**	10

Architecture

Type system

The following figure shows the data structure of this program.



Document is the structure to store the relevenceValue, queryID, text and tokenList for each document.

Token is to store the token with the relative frequency for each document.

In this way, one Document is corresponding to one or more Tokens.

Data structure

For TokenVector: I use HASHMAP to store the token vector for each document. And because of high efficiency of hash map. find function and this will highly improve the performance of the program.

```
/** store the term and term frequency */
HashMap<String, Integer> hs= new HashMap<String, Integer>();
```

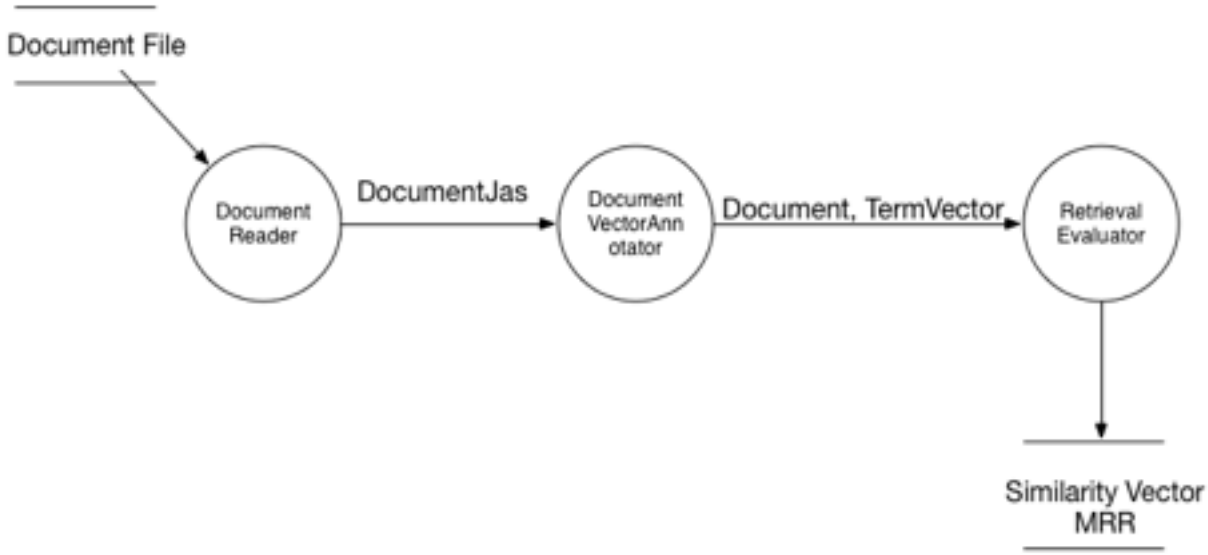
For similarity for each document and group, I used the following structure to store.

```
/** similarity for all **/
public ArrayList<ArrayList<Double>> similarityAll=new ArrayList<ArrayList<Double>>();
```

For document information, I use a class DocInfo to store all the related info about Document as the figure shows below:

```
class DocInfo
{
  int qid;
  int rel;
  Map<String, Integer> vector;
  String text;

  public DocInfo(int qid,int rel,Map<String,Integer> vector,String text) {
    // TODO Auto-generated constructor stub
    this.qid=qid;  //the id for the documents
    this.rel=rel;  // the relevant for each documents
    this.vector=vector; //the vector of the tokenlist and frequency
    this.text=text;  //the text for each documents
  }

  public Map<String, Integer> GetVector()
  {
    return vector;
  }

  public int GetQid()
  {
    return qid;
  }

  public int GetRel()
  {
    return rel;
  }

  public String GetText()
  {
    return text;
  }
}
```

System structure



The system data flow is shown as the figure above. And DocumentReader is responsible for reading document data from file and store in Document Jas. DocumentVectorAnnotator is responsible for analyzing the term frequency of each term and create Term vector for each document. RetrievalEvaluator is responsible for calculate the similarity between each document and give the rank, finally calculating the MRR.

Flow for calculating the similarity

Without using a complex structure to store all data from file, I calculated the similarity between each query and document during the traversal.

Knowledge Base

Tokenization algorithm- Stanford Tokenization

PTBTokenizer is a an efficient, fast, deterministic tokenizer. On a typical 2010 computer, it will tokenize text at a rate of about 200,000 tokens per second.

This is much more efficient than the tokenization function which only consider the whitespace.

Stemming algorithm- StanfordLemmatizer

The simple vector model does not understand die and dies denote the same thing in most contexts. StanfordLemmatizer is used to stem words more efficiently.

Similarity Algorithm

Jaccard Index

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}.$$

The Jaccard coefficient measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets

Dice coefficient

$$QS = \frac{2C}{A+B} = \frac{2|A \cap B|}{|A| + |B|}$$

A and B are the number of species in samples A and B, respectively, and C is the number of species shared by the two samples; QS is the quotient of similarity and ranges from 0 to 1.