

11-791 Homework 4 Report

Yuanchi Ning

Andrew ID: yuanchin Date: Oct. 2013

1. Design

1.1 Goal

Given the goal that extracting bag of words feature vector from the input text collection, computing the cosine similarity between two sentences, and computing the Mean Reciprocal Rank for all sentences in the text collection, this assignment is focused on designing, implementing and improving these procedures. In addition, this report discusses some extra incorporation with annotations from Stanford CoreNLP and some bonus similarity measures instead of cosine similarity.

1.2 System Design

Architectural Aspect (type system):

Since in this text collection, all the documents are represented within only one sentence, there is no need to create `Sentence` annotator type. And because that the `Document` type has the feature `relevanceValue`, which indicates whether the sentence is query or right answers or wrong answers, so it's unnecessary to create new types like `Question` and `Answer`. Besides, since the similarity measures based on the tokens in the sentence are used to calculate the score of each document, the `Document` and `Token` type in the original type system are enough in this assignment.

The design pattern is not complicated. Since most part of the program is given and the given codes are already very reasonable, there isn't much flexibility as well as necessity on re-design the system. Since different similarity measures are implemented in the system, an abstract class of scoring inherited by different methods could be applied when implementing the cosine similarity, dice coefficient and Jaccard coefficient measures. But in this case, these three methods are quite simple and have more simplicity if implementing separately, therefore the simplest implementation of the methods is adopted.

Algorithmic Aspect:

For Task 1, the first step is to construct the document vector in order to convert both the query document and the answer document into representation of vector space. Though the simple splitting methods can be used to get normal tokens within the sentences so that the token term vectors can be further used to calculate the similarity between two documents, it is more reasonable to adopt **stems** instead of tokens to form the term vector. Therefore, the Stanford CoreNLP tool is integrated in the system to do the sequential **tokenization**, **sentence splitting**,

POS tagging and **lemmatization** process of each document.

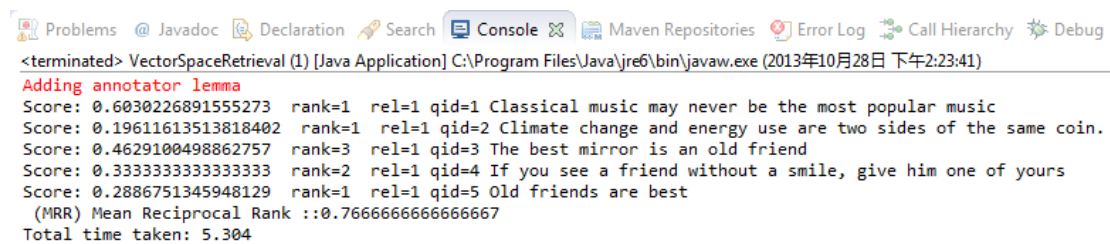
Besides the default similarity measure cosine similarity, the **dice coefficient** and **Jaccard coefficient** similarity measures are also implemented in the system to compare different scoring performance on the specific text collection.

2. Implementation

2.1 Task 1

The implementation of this task is conceptually simple. By add the dependency of Stanford CoreNLP toolkit, the pipeline containing tokenization, sentence splitting (required by most Annotators), POS tagging, lemmatization can be easily described using the `java.util.Properties` class. Utilizing the result (stems and the corresponding stem frequency) of document processing, the term vector is created using the data structure `HashMap`.

And the cosine similarity method is used to score each answer document. The experiment result is shown as below:



```

<terminated> VectorSpaceRetrieval (1) [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (2013年10月28日 下午2:23:41)
Adding annotator lemma
Score: 0.6030226891555273 rank=1 rel=1 qid=1 Classical music may never be the most popular music
Score: 0.19611613513818402 rank=1 rel=1 qid=2 Climate change and energy use are two sides of the same coin.
Score: 0.4629100498862757 rank=3 rel=1 qid=3 The best mirror is an old friend
Score: 0.3333333333333333 rank=2 rel=1 qid=4 If you see a friend without a smile, give him one of yours
Score: 0.2886751345948129 rank=1 rel=1 qid=5 Old friends are best
(MRR) Mean Reciprocal Rank ::0.7666666666666667
Total time taken: 5.304
  
```

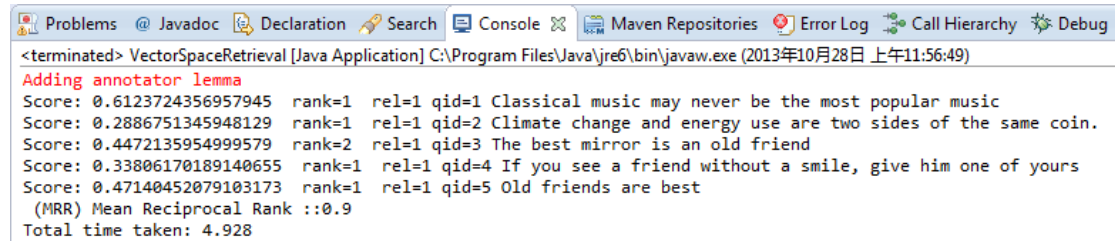
Fig1. Task1 result when using cosine similarity

2.2 Task 2

When analyzing the result of Task 1, we can see that some wrong answer documents are ranked higher than the correct answer document. Taking the third query as an example, we can observe that since the document “The best antiques are old friends” is shorter than the correct document “The best mirror is an old friend”, the score of the former one is higher although the overlapping part (inner product in the cosine similarity) is the same.

Improvements can be achieved by adding filter on stop words. Utilizing the stopwords.txt provided, we filtered the stop words and only added the stems not in the stop word list into the token list to form the term vector.

After optimize this part, the experiment result is shown as below:



```

<terminated> VectorSpaceRetrieval [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (2013年10月28日 上午11:56:49)
Adding annotator lemma
Score: 0.6123724356957945 rank=1 rel=1 qid=1 Classical music may never be the most popular music
Score: 0.2886751345948129 rank=1 rel=1 qid=2 Climate change and energy use are two sides of the same coin.
Score: 0.4472135954999579 rank=2 rel=1 qid=3 The best mirror is an old friend
Score: 0.33806170189140655 rank=1 rel=1 qid=4 If you see a friend without a smile, give him one of yours
Score: 0.47140452079103173 rank=1 rel=1 qid=5 Old friends are best
(MRR) Mean Reciprocal Rank ::0.9
Total time taken: 4.928

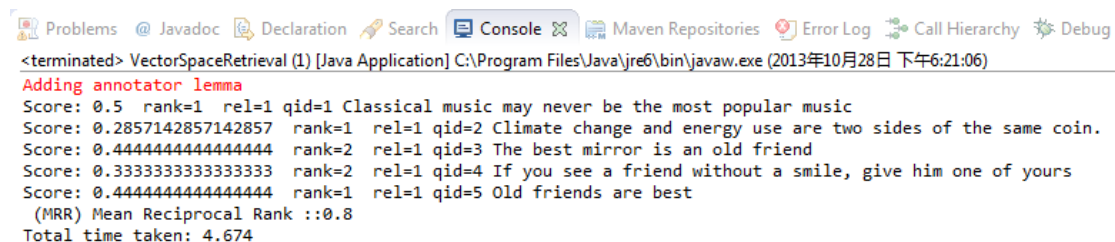
```

Fig2. Task2 result when using cosine similarity

2.3 Bonus

As mentioned above, the system integrated the Stanford CoreNLP tool to process the sentence and get the stems. This makes the term vector more reasonable to do the subsequently scoring part. Although by utilizing the Stanford CoreNLP, the efficiency is not good as the simplest tokenization method, it is worthwhile to make the performance better, especially when applied in the large data sets.

Besides the suggested cosine similarity measurement, the dice coefficient and the Jaccard coefficient similarity scoring methods are also implemented. And the experiments results are shown as below:

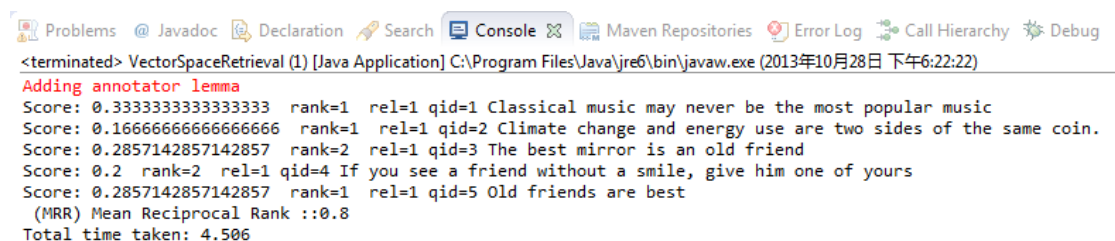


```

<terminated> VectorSpaceRetrieval (1) [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (2013年10月28日 下午6:21:06)
Adding annotator lemma
Score: 0.5 rank=1 rel=1 qid=1 Classical music may never be the most popular music
Score: 0.2857142857142857 rank=1 rel=1 qid=2 Climate change and energy use are two sides of the same coin.
Score: 0.4444444444444444 rank=2 rel=1 qid=3 The best mirror is an old friend
Score: 0.3333333333333333 rank=2 rel=1 qid=4 If you see a friend without a smile, give him one of yours
Score: 0.4444444444444444 rank=1 rel=1 qid=5 Old friends are best
(MRR) Mean Reciprocal Rank ::0.8
Total time taken: 4.674

```

Fig3. Task2 result when using dice coefficient



```

<terminated> VectorSpaceRetrieval (1) [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (2013年10月28日 下午6:22:22)
Adding annotator lemma
Score: 0.3333333333333333 rank=1 rel=1 qid=1 Classical music may never be the most popular music
Score: 0.16666666666666666 rank=1 rel=1 qid=2 Climate change and energy use are two sides of the same coin.
Score: 0.2857142857142857 rank=2 rel=1 qid=3 The best mirror is an old friend
Score: 0.2 rank=2 rel=1 qid=4 If you see a friend without a smile, give him one of yours
Score: 0.2857142857142857 rank=1 rel=1 qid=5 Old friends are best
(MRR) Mean Reciprocal Rank ::0.8
Total time taken: 4.506

```

Fig4. Task2 result when using Jaccard coefficient

The best method of the three similarity measures in this text collection is the cosine similarity measure. But we can see from the detailed query results that for different specific query, the three different similarity measures perform differently.

Summary

In this assignment, I felt that I am more skilled of using the UIMA framework. And beyond that, the assignment helped me broaden my knowledge in areas of natural language processing, text analysis, similarity measures and made me aware of the widely use of vector space.