11791 Homework 4 report - Engineering and Error Analysis with UIMA

Zhengzhong Liu
Language Technology Institute
Carnegie Mellon University
Pittsburgh, PA 15213

1 Initial design and implementations

In this task, we are require to implement a Cosine similarity answer ranker and implement a Mean Reciprocal Rank evaluator, which are straightforward and share a lot of similarities with previous homework. One requirement in this task is to follow the predefined framework and implement the functionalities. The main type system and architecture is designed, which is sufficient to finish this task. Hence, I didn't try to change the architecture extensively but instead focusing on analyzing problems of the ranking algorithm.

1.1 System architecture

Notably, this is not constructed as a typical UIMA setup, where there is no collection reader implemented. Instead, the VectorSpaceRetriever deals with input and JCas creation. This method make it easy to use, however, it might not be flexible in reality because it is a little bit hard to scale up. The DocumentReader class split each line into a document. This is in fact an analysis engine, which annotate each line as a document, but not a collection reader. Here I didn't change the package name to reflect this fact, because this probably act as such functionality.

Most of the ranking task are postpone into the RetrievalEvaluator. The instruction also points out that main logic should be implemented here, a better idea might be to split this up into two different anwyslis engine. But in order to follow the provided architecture, this is not changed. And it is considered that this will not significantly hamper the development.

1.2 DocumentVectorAnnotator

In the document vector annotator, one is require to implement the methods to annotate token frequency in each document. This require one iteration only to calculate the count, and one iteration to assign these token to the documents. Note that as we are using a Bag Of Word approach in terms of cosine similarity, the position of each token does not matter, so they are simply put into the Doc object with its frequency. These information will be used during cosine similarity. To do tokenization, the StanfordCoreNlp tokenizer is employed in order to handle the detailed problems of punctuations.

1.3 Cosine similarity

After gathering the frequency, computing cosine is straightforward. One just calculate the dot product and respect length of query and document. If we are only interested in ranking, we don't need to calculate the length of the "Question" all the time because it will be a constant for all the answers. In order to plug in new scoring methods. The cosine similarity is not direct implemented as a function. An QueryScorer interface is created instead, there are two functions to implement in this interface, one is the calculateScore method, the other one is the name of this method. Cosine similarity class

implement this interface. Then in the pipeline, we can calculate the score using various methods, which is reflected in the implementation as followed:

```
List<QueryScorer> scorers = new ArrayList<QueryScorer>();
scorers.add(new CosineQueryScorer());
scorers.add(new JaccardScorer());
scorers.add(new RelativeFreqScorer());
```

The other two scorer will be intruded later in this report.

In designing cosine similarity, one need to which words should be compared. In this task, the lower cased words are compared and punctuations are removed.

1.4 Evaluation

To evaluate the system, I created a new Class called Answer to store the information of each answer, including the correct relevance, the system predicted score, and the answer content, etc. This class implements that comparable methods so that it is easy to sort or compare this. The comparison is done by comparing this predicted score. With this class, the methods for evaluation can be implemented conveniently. To compute MRR, one just need to know the position of the correct answer in the list, which is a rank determination problem the complexity is $O(n)$. However, sorting the entire list will give a $O(nlogn)$ method, these two methods will not differ a lot in performance because number of answers in examples are relatively small. Both these methods are implemented in the system. The purpose for implementing the sorting based algorithm is to help error analysis: we need to know how each answers are ranked so that we would like to see that complete rank list. In the following section, the results are all generated using the sorting based algorithm. The submitted homework use the rank determination method.

2 Error analysis

2.1 Initial results

The initial implementation using Cosine similarity gives the following result:

```
====Result of Cosine ====
Score : 0.301511, rank=1 ,rel=1,qid=1,Classical music may never be the most popular music
Score : 0.235702, rank=2 ,rel=0,qid=1,Everybody knows classical music when they hear it
Score : 0.178174, rank=3 ,rel=0,qid=1,Pop music has absorbed influences from most other genres of popular music
Score : 0.204124, rank=1 ,rel=1,qid=2,Climate change and energy use are two sides of the same coin.
Score : 0.000000, rank=2 ,rel=0,qid=2,Old wine and friends improve with age
Score : 0.000000, rank=3 ,rel=0,qid=2,With clothes the new are the best, with friends the old are the best
Score : 0.330049, rank=1 ,rel=0,qid=3,My best friend is the one who brings out the best in me
Score : 0.308607, rank=2 ,rel=1,qid=3,The best mirror is an old friend
Score : 0.111111, rank=3 ,rel=0,qid=3,The best antiques are old friends
Score : 0.198762, rank=1 ,rel=0,qid=4,Wear a smile and have friends; wear a scowl and have wrinkles
Score : 0.192450, rank=2 ,rel=0,qid=4,Behind every girls smile is a best friend who put it there
Score : 0.172133, rank=3 ,rel=1,qid=4,If you see a friend without a smile, give him one of yours
Score : 0.105409, rank=1 ,rel=1,qid=5,Old friends are best
Score : 0.079682, rank=2 ,rel=0,qid=5,Old wine and friends improve with age
Score : 0.037268, rank=3 ,rel=0,qid=5,With clothes the new are the best, with friends the old are the best
 (MRR) Mean Reciprocal Rank ::0.7666666666666667
```

As can be seen, there are two questions that the cosine similarity cannot give perfect ranking, which are query 3 and query 4. The problem of query 3 is obvious, that longer answer is matched because it has a higher chance of matching some terms. The problem of query 4 is instead quite complicated, it require understanding of the meaning of these metaphors [1]. Term matching thus could not help much for the latter problem. So it would be better to try resolve the problem of query 3 first.

2.2 Relative frequency

In order to account for the length of document, I introduce the relative frequency metric that consider penalize the term frequency of longer documents. The idea is simply calculate the term frequency by dividing it with the document length. And the final score is given by dot product (cosine similarity

[1] Actually, it is even controversial whether the relevant document marked by the given data is really relevant with the query.

normalization will remove the effect of dividing by document length). The updated result is shown as bellowed:

```
====Result of Relative Frequency ====
Score : 0.333333, rank=1 ,rel=1,qid=1,Classical music may never be the most popular music
Score : 0.250000, rank=2 ,rel=0,qid=1,Everybody knows classical music when they hear it
Score : 0.166667, rank=3 ,rel=0,qid=1,Pop music has absorbed influences from most other genres of popular music
Score : 0.250000, rank=1 ,rel=1,qid=2,Climate change and energy use are two sides of the same coin.
Score : 0.000000, rank=2 ,rel=0,qid=2,Old wine and friends improve with age
Score : 0.000000, rank=3 ,rel=0,qid=2,With clothes the new are the best, with friends the old are the best
Score : 0.428571, rank=1 ,rel=1,qid=3,The best mirror is an old friend
Score : 0.384615, rank=2 ,rel=0,qid=3,My best friend is the one who brings out the best in me
Score : 0.166667, rank=3 ,rel=0,qid=3,The best antiques are old friends
Score : 0.333333, rank=1 ,rel=0,qid=4,Wear a smile and have friends; wear a scowl and have wrinkles
Score : 0.250000, rank=2 ,rel=0,qid=4,Behind every girls smile is a best friend who put it there
Score : 0.230769, rank=3 ,rel=1,qid=4,If you see a friend without a smile, give him one of yours
Score : 0.250000, rank=1 ,rel=1,qid=5,Old friends are best
Score : 0.142857, rank=2 ,rel=0,qid=5,Old wine and friends improve with age
Score : 0.071429, rank=3 ,rel=0,qid=5,With clothes the new are the best, with friends the old are the best
 (MRR) Mean Reciprocal Rank ::0.8666666666666668
```

Here we see clearly that the performance is improved, by correct the problem of query 3 mentioned above. However, the problem of query 4 is still not solved. As discussed above, this problem is not easy to be handled simply using term matching with a justifiable theory. In this homework, I didn't try to tackle this query again.

2.3 Jaccard coefficient (bonus task)

To investigate other methods, I also check the Jaccard coefficient suggested in the bonus task. The result is shown as below:

```
====Result of Jaccard ====
Score : 0.200000, rank=1 ,rel=0,qid=1,Everybody knows classical music when they hear it
Score : 0.181818, rank=2 ,rel=1,qid=1,Classical music may never be the most popular music
Score : 0.066667, rank=3 ,rel=0,qid=1,Pop music has absorbed influences from most other genres of popular music
Score : 0.176471, rank=1 ,rel=1,qid=2,Climate change and energy use are two sides of the same coin.
Score : 0.000000, rank=2 ,rel=0,qid=2,Old wine and friends improve with age
Score : 0.000000, rank=3 ,rel=0,qid=2,With clothes the new are the best, with friends the old are the best
Score : 0.300000, rank=1 ,rel=1,qid=3,The best mirror is an old friend
Score : 0.266667, rank=2 ,rel=0,qid=3,My best friend is the one who brings out the best in me
Score : 0.090909, rank=3 ,rel=0,qid=3,The best antiques are old friends
Score : 0.166667, rank=1 ,rel=0,qid=4,Wear a smile and have friends; wear a scowl and have wrinkles
Score : 0.166667, rank=2 ,rel=0,qid=4,Behind every girls smile is a best friend who put it there
Score : 0.100000, rank=3 ,rel=1,qid=4,If you see a friend without a smile, give him one of yours
Score : 0.076923, rank=1 ,rel=1,qid=5,Old friends are best
Score : 0.062500, rank=2 ,rel=0,qid=5,Old wine afFnd friends improve with age
Score : 0.043478, rank=3 ,rel=0,qid=5,With clothes the new are the best, with friends the old are the best
 (MRR) Mean Reciprocal Rank ::0.7666666666666667
```

We see that the MRR score is between Jaccard and Cosine in this case. However, by looking into the score, we see some differences in their discriminative power. For example, in query 5, in the cosine rank list, the ratio of the first two answer scores is : 1.329, in Jaccard, the ratio is 1.225. This result shows that Jaccard probably will have less discriminative power in some queries.

3 Conclusion

In this task, I use an Interface to make extension of implementing different scoring functions much easier. An new class is used to store information of answers (instead of using a bunch of meaningless HashMap), this make the program easier to be understand and extendable.

For error analysis, we see that how error analysis can help improve the system. However, we also see the fact that error analysis will direct our system in one domain only, the error found in this domain might not persist in other ones. Also, without enough data, error analysis is difficult. On the other hand, by doing error analysis, we should not focus on the evaluation metric only, we also could look inside to see if there are other aspects that can be reflected from a more detailed analysis (for example, discriminative power of different methods, which may show that robustness of one method).