11-791 Design and Engineering of Intelligent Information System & 11-693 Software Methods for Biotechnology Homework 4

Engineering and Error Analysis with UIMA

Name: Bo Ma

Andrew id: bom

Task 1
Building Vector space Retrieval Model using UIMA

The following is the process of implementation of my system.

1. I read the sentence, the query id and the rel number from the input file.

2. I tokenize the terms in the sentence string. I use the \\s+ to split the word in the sentence. In this step, I remove the stop words. And put the terms and it's frequency in a hash map, so it is easy for me to find and use them in the next step; This hash map is the bag of words feature vector from the input text collection

3. I use the feature vector to compute the cosine similarity between the two sentences in the text collection. Looping the pair of query and other sentence and compute the dot product and sentence length, in this way I get the score for each document.

4. Use the score from the last step, rank the document whose rel value is 1. I loop all the sentence whose rel value is 0 and find the sentence whose score is bigger than the document whose rel value is 1. In this way, we can calculate the Mean Reciprocal Rank computation in the next step.

5. Use the rank from the last step, compute the Mean Reciprocal Rank.

The result:

```
Score: 0.612372     rank=1    rel=1 qid=1 Classical music may never be the most popular music
Score: 0.462910     rank=1    rel=1 qid=2 Climate change and energy use are two sides of the same coin.
Score: 0.5                    rank=2     rel=1 qid=3 The best mirror is an old friend
Score: 0.0                    rank=3     rel=1 qid=4 If you see a friend without a smile, give him one of yours
Score: 0.235702     rank=1    rel=1 qid=5 Old friends are best
 (MRR) Mean Reciprocal Rank ::0.7666666666666667
Total time taken: 0.996
```

Task 2
Error Analysis

In this part, I will show my improvement about the system. Some of the query has very low rank and effect the performance of the system a lot.

1. At first the result of my system is the following.

```
Score: 0.3333       rank=1    rel=1 qid=1 Classical music may never be the most popular music
Score: 0.1          rank=1    rel=1 qid=2 Climate change and energy use are two sides of the same coin.
Score: 0.5          rank=1    rel=1 qid=3 The best mirror is an old friend
```

```
Score: 0.0952       rank=3   rel=1 qid=4 If you see a friend without a smile, give him one of yours
Score: 0  rank=3   rel=1 qid=5 Old friends are best
 (MRR) Mean Reciprocal Rank ::0.7333334
Total time taken: 1.253
```

As we can see query id 3 and query id 4 rank low. The value of MRR is just 0.733. When I loot at the detailed information about the query 4 and query 5. I can see the follow result.

```
Score: 0  rel=99    qid=4 The shortest distance between new friends is a smile
Score: 0.2581       rel=0     qid=4 Wear a smile and have friends; wear a scowl and have wrinkles
Score: 0.0952       rel=1     qid=4 If you see a friend without a smile, give him one of yours
Score: 0.369        rel=0     qid=4 Behind every girls smile is a best friend who put it there
Score: 0  rel=99    qid=5 It takes a long time to grow an old friend
Score: 0.3826       rel=0     qid=5 Old wine and friends improve with age
Score: 0.3361       rel=0     qid=5 With clothes the new are the best, with friends the old are the best
Score: 0  rel=1     qid=5 Old friends are best
```

Checking the content of the query 4 and query 5 , I find that
1, I can change all the words into lowercase, this will make the words “Old” and “old” matched.
2, I should remove the process the “,” and “.” , this will make the “smile,” match with “smile”.
3, I can remove the stopwords. Stopwords is more frequent than other
4, I also think about stemming strategy.

2. Then **I convert all the character into lower case**.

When we look in the query id 5, we can see the relevant sentence get the score 0 , it is because the “Old” terms cannot match with the “old” term in the query. Thus, in this step , I choose to convert all the character into lower case. Since it really makes sense to match the words in this way, this strategy helps me a lot. We can match more terms without considering their cases.

```
Score: 0.612372     rank=1   rel=1 qid=1 Classical music may never be the most popular music
Score: 0.412910     rank=1   rel=1 qid=2 Climate change and energy use are two sides of the same coin.
Score: 0.5          rank=2   rel=1 qid=3 The best mirror is an old friend
Score: 0.0952       rank=3   rel=1 qid=4 If you see a friend without a smile, give him one of yours
Score: 0.235702     rank=1   rel=1 qid=5 Old friends are best
 (MRR) Mean Reciprocal Rank ::0.7666666666666667
Total time taken: 0.996
```

Compare:
As we can see in the result, the rank of query 5, improves a lot. At first it just rank 3 and get score 0, now the rank is 1 and score is 0.235702.

3. Use the TextTokenizer provide by the UIMA
When I look in the query id 4, I find that the relevant sentence get very low score which is just 0.0952. The reason for this is that, it cannot match the term “smile,” with the term “smile”. So I should split the Punctuation and the english words. To improve this I use the TextTokenizer provided by the UIMA. Then I use the **TextTokenizer** in the UIMA and it will process the punctuation. In this way, I improve the score of the relevant sentence in the query 4. Its rank increase from 3 to 2.

Now here is the new result that I got:

```
Score: 0.45226701686664544            rank=1   rel=1 qid=1 Classical music may never be the most popular music
Score: 0.294174202707276  rank=1   rel=1 qid=2 Climate change and energy use are two sides of the same coin.
Score: 0.5070925528371099  rank=1   rel=1 qid=3 The best mirror is an old friend
Score: 0.25         rank=2   rel=1 qid=4 If you see a friend without a smile, give him one of yours
Score: 0.15811388300841897            rank=1   rel=1 qid=5 Old friends are best
 (MRR) Mean Reciprocal Rank ::0.9
```

Compare:

As we can see in the result, the rank of query 3 and query 4 improves a lot.   For the query 3, it rank increase to the 1, and the rank of the query 4, increase from the 3 to 2.

4. **I removes the stopwords.**

I think the stopwords really affect the score of the cosine similarity, because some words like "a" , "the" "and" their frequency is very high, but they are almost meaningless to find the relation of two sentence. We just need the key words to get the cosine similarity score the two sentence. This will improve the score of the sentence, because we move the noisy terms like "a", "the".

```
Score: 0.6123724356957945  rank=1     rel=1 qid=1 Classical music may never be the most popular music
Score: 0.43301270189221935            rank=1     rel=1 qid=2 Climate change and energy use are two sides of the same coin.
Score: 0.5              rank=2     rel=1 qid=3 The best mirror is an old friend
Score: 0.16903085094570328            rank=1     rel=1 qid=4 If you see a friend without a smile, give him one of yours
Score: 0.23570226039551587            rank=1     rel=1 qid=5 Old friends are best
 (MRR) Mean Reciprocal Rank ::0.9
Total time taken: 1.049
```

Compare:

Although from the above result, we cannot see the change of the MRR, but the similarity score of the sentence increases a lot. The query 4 rank change from 2 to 1, so this strategy improve some of the sentence.

5. **do the stemming strategy.**

I also do the term sterming, which I change the term like "computers " into the "computer" , "friends" into "friend". Since, I word had different forms, so if we do the stemming, it will help use increase the cosine similarity score of the two sentence.

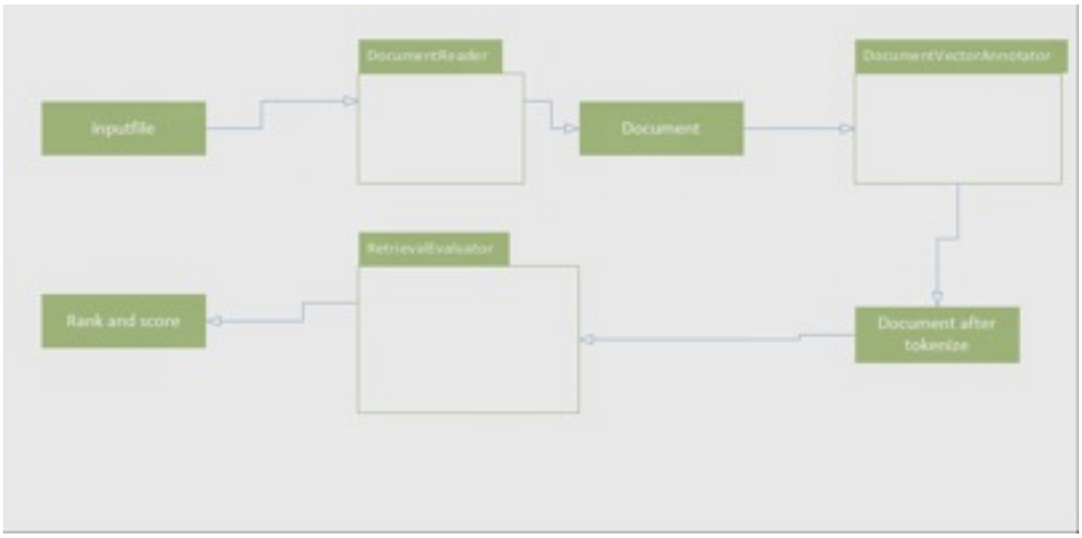
```
Score: 0.6123724356957945  rank=1     rel=1 qid=1 Classical music may never be the most popular music
Score: 0.43301270189221935            rank=1     rel=1 qid=2 Climate change and energy use are two sides of the same coin.
Score: 0.5              rank=2     rel=1 qid=3 The best mirror is an old 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: 0.888
```

Compare:

Althought the MRR value does not change, the score for the query 4 and query 5 increases a lot compare with not doing stemming strategy. For the query 4, I find that the non relevant sentence is quite similar to the query sentence. Thus, in this query, it need more strategy to improve the performance.

Summary

Here is the design pattern of my system.



The above figure is my understanding of the UIMA framework of this homework. The system use the Document Reader as the Collection Reader. The Document Annotator is used to make the sentence and words annotation. The CAS consumer is the Retrieval Evaluator which get the performance and evaluation of the system.
There are just two type system the “document ” and the “token”. The documentvectorannotator just annotate the document and token.
We use the tokenized result and then compute the similarity score and rank all the document.

Component Description:

1. **DocumentReader**

   we get the type system information from the instructor. And the reader is quite helpful. I process the input data and extract the rel value, query id and sentence string from the input collection. After the input we put this information in the CAS for futher processing.
2. **DocumentVector Annotator**

   In the last step, we get the document from the input. Then, this step we will make these documents into the bag of words. I use the string split function to split the word in the sentence and put them in the token annotation. For task 2 in this step, I will change this method, and use the **TextTokenizer** provided by the UIMA, which can process the stemming, stop words.
3. **Retrieval Evaluator**

   After the token annotation, I get all the tokens in the document. In this step, I will use different similarity metric to measure the difference between query sentence and other sentence. Then, I rank the document with their ranking and compute the MRR value for the all query.

BONUS

I also implement two other score algorithms.
The first one is dice coefficient, Jaccard coefficient

computeDiceCoefficient:

```
Score: 0.5          rank=1  rel=1 qid=1 Classical music may never be the most popular music
Score: 0.42857142857142855    rank=1  rel=1 qid=2 Climate change and energy use are two sides of the
same coin.
Score: 0.5          rank=1  rel=1 qid=3 The best mirror is an old friend
Score: 0.16666666666666666    rank=2  rel=1 qid=4 If you see a friend without a smile, give him one of
yours
Score: 0.2222222222222222     rank=1  rel=1 qid=5 Old friends are best
 (MRR) Mean Reciprocal Rank ::0.9
Total time taken: 1.127
```

computeJaccardCoefficient:

```
Score: 0.3333333333333333     rank=1  rel=1 qid=1 Classical music may never be the most popular music
Score: 0.2727272727272727     rank=1  rel=1 qid=2 Climate change and energy use are two sides of the
same coin.
Score: 0.3333333333333333     rank=1  rel=1 qid=3 The best mirror is an old friend
Score: 0.09090909090909091    rank=2  rel=1 qid=4 If you see a friend without a smile, give him one of
yours
Score: 0.125     rank=1  rel=1 qid=5 Old friends are best
 (MRR) Mean Reciprocal Rank ::0.9
Total time taken: 1.205
```

similarity measures	cosine similarity	dice coefficient	Jaccard coefficient
rank of query 1	1	1	1
rank of query 2	1	1	1
rank of query 3	2	1	1
rank of query 4	3	2	2
rank of query 5	1	1	1
MRR	0.766666	0.9	0.9
score	0.23570	0.1111	0.125

I build a table to compare the result of three different method. We can see that using the dice coefficient and Jaccard coefficient similarity metric, we improve the rank result of the query 3 and query 4. And our MRR really improves from the 0.7666 to the MRR.

I think this improvement is made by two things.

First, use new tokenizing method.

Second, change the cosine similarity to the dice coefficient or Jaccard coefficient.