Home work 4: Engineering and Error Analysis with UIMA

Task:

The task is to find the similarity between the question and the answer provided in the input document. The similarity is measured using the Cosine Similarity score. The answers are ranked based on the scores and the rank corresponding to the correct answer is noted to calculate the Mean Reciprocal Rank(MRR). After analysis the score, should be improved if there is scope by some means.

Program Flow:

The base pipeline along with the type systems was provided as part of the archetype. The CollectionReader (DocumentReader.java) reads the input document.txt file line by line and passes the control to the annotator – DocumentVectorAnnotator.java. In this file, the token and its frequency are measured and updated in the tokenList of the Document type. Once the frequency of the tokens of all the sentences in the document has been calculated the control moves to the RetrievalEvaluator.java (CASConsumer) which evaluates the CosineSimilarity between the question and answers and calculates the Mean Reciprocal Rank. As bonus, I have also implemented the **Jaccard** Coefficient and **Dice** coefficient.

Error Analysis:

Background:

Our goal should be to maximise the score of the correct answer. And if you observe the answer options, you will note that the question and answers do not always have the same cases everywhere. For example, in query 2, 'climate' is in lower case whereas in the correct answer the word 'Climate' is in sentence case. So, the comparison should be case sensitive.

If you observe, the answers are much longer than the questions. So, the denominator of the cosine similarity will always be huge if we don't take any measure. And if you notice, there are few words that does not add much meaning to the sentence like 'a', 'an', 'the' , etc. Removing those words will help in reducing the denominator in the score calculation formula.

The baseline implementation includes the lowercase check and also includes the elimination of words that does not add much meaning to the sentence. Based on the implementation, I got the MRR as follows:

Cosine Similarity

```
Score: 0.6123724356957945  rank=1     rel=1 qid=1 Classical music may never
be the most popular music
Score: 0.46291004988627577 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: 0.2357022603955159  rank=1     rel=1 qid=5 Old friends are best
```

```
 (MRR) Mean Reciprocal Rank ::0.7666666666666667
```

Jaccard Coefficient

```
Score: 1.0606601717798212  rank=1     rel=1 qid=1 Classical music may never
be the most popular music
Score: 0.8920101987448574  rank=1     rel=1 qid=2 Climate change and energy
use are two sides of the same coin.
Score: 0.7734590803390136  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: 0.3143131398684654  rank=1     rel=1 qid=5 Old friends are best
 (MRR) Mean Reciprocal Rank ::0.7666666666666667
```

Dice Coefficient

```
Score: 1.3156515806127753  rank=1     rel=1 qid=1 Classical music may never
be the most popular music
Score: 1.177569409688475   rank=1     rel=1 qid=2 Climate change and energy
use are two sides of the same coin.
Score: 1.0 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: 0.4782926234762005  rank=1     rel=1 qid=5 Old friends are best
 (MRR) Mean Reciprocal Rank ::0.7666666666666668
```

The Mean Reciprocal Rank is the same for all three coefficients as it is calculated based on the rank and the rank is same in all three methods. Only the scores are different as they have a different formula.

```
jaccard_Score = query_docum_common / (Math.sqrt(query) + Math.sqrt(docum) –
Math.sqrt(query_docum_common));
```

The jaccard score is as simple as getting the frequency of common words in question and answer divided by the union of words in question and in the answer.

```
 dice_Score = (2 * query_docum_common) / (Math.sqrt(query) + Math.sqrt(docum));
```

The dice score is as simple as getting twice the frequency of common words in question and answer divided by the union of words in question and in the answer + the common words in question and answer.

Performance Enhancement - Implementing Lemmatization/Stemming:

If you observe the query 4, the correct answer has got a score 0. This is because the algorithm thinks that there is no common word in the question and the answer.

```
qid=4 rel=99      The shortest distance between new friends is a smile
qid=4 rel=0       Wear a smile and have friends; wear a scowl and have wrinkles
qid=4 rel=1       If you see a friend without a smile, give him one of yours
qid=4 rel=0       Behind every girls smile is a best friend who put it there
```

If you observe, the question has the word 'friends' and the correct answer has the word 'friend'. The only difference is the singularity. So, getting the root word of the tokens will help in coming up with a good solution. I implemented the Stanford pipeline to get the lemma of the token and used that to compare the distance between the question and answer.

Cosine Similarity

```
Score: 0.6123724356957945  rank=1     rel=1 qid=1 Classical music may never
be the most popular music
Score: 0.3086066999241838  rank=1     rel=1 qid=2 Climate change and energy
use are two sides of the same coin.
Score: 0.25     rank=2     rel=1 qid=3 The best mirror is an old friend
Score: 0.36514837167011077 rank=1     rel=1 qid=4 If you see a friend
without a smile, give him one of yours
Score: 0.4714045207910318  rank=1     rel=1 qid=5 Old friends are best
 (MRR) Mean Reciprocal Rank ::0.9
```

Jaccard Coefficient

```
Score: 1.0606601717798212  rank=1     rel=1 qid=1 Classical music may never
be the most popular music
Score: 0.5433265588567421  rank=1     rel=1 qid=2 Climate change and energy
use are two sides of the same coin.
Score: 0.3333333333333333  rank=2     rel=1 qid=3 The best mirror is an old
friend
Score: 0.6113694871156081  rank=1     rel=1 qid=4 If you see a friend
without a smile, give him one of yours
Score: 0.7227190746477863  rank=1     rel=1 qid=5 Old friends are best
 (MRR) Mean Reciprocal Rank ::0.9
```

Dice Coefficient

```
Score: 1.3156515806127753  rank=1     rel=1 qid=1 Classical music may never
be the most popular music
Score: 0.78504627312565    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: 0.8536870611335536  rank=1     rel=1 qid=4 If you see a friend
without a smile, give him one of yours
Score: 0.956585246952401   rank=1     rel=1 qid=5 Old friends are best
```

```
(MRR) Mean Reciprocal Rank ::0.9
```

The lemma/stem of the tokens has improved the rank of the correct answer of query 4 and also has boosted the MRR from 0.7666666666666667 to 0.9.

References:

http://www.stanford.edu/~maureenh/quals/html/ml/node68.html – Information about the Jaccard coefficient and Dice coefficient are available in this link.