11-791 : Design and Engineering of Intelligent Information Systems
Homework 4

Name : Soumya Batra
Andrew ID : soumyab

***** All taks along with all BONUS tasks have been completed *****

TASK 0 : Generating TypeSystem

We tried to modify the typesystem by adding score and document ID attributes to the Document type and creating a new type for Document Set that contains all documents for a particular query ID.

For this to work, we wished to create a list of documents in processCAS method of RetreivalEvaluator class and use it in the collectionProcessComplete method of the same class. However, at the time of collectionProcessComplete method, the list was full of Low Level Exceptions. Populating the list after creating new document objects also didn't help.
Hence, we discarded this algorithm and started from scratc with a new one.

Therefore, **no modifications** were made to the typesystem.

TASK 1 : Extracting bag of words feature vector

1. DocumentVectorAnnotator is used to extract bag of words from the input text collection.
2. The text is tokenized based on delimitors : blank space( ) , full stop(.), comma(,), excalamation mark(!), question mark(?).
3. For every document, we use a HashMap with words as keys and their frequency as the corresponding mapping.
4. This HashMap is converted to a the FSList feature vector using the pre-provided Utils class.
5. For improving performance, we do variation as
   5.1 converting all tokens to lowercase
   5.2 using stopwords list

TASK 2 : Computing Cosine Similarity

1. computeCosineSimilarity() method inside RetreivalEvaluator class is used to calculate cosine similarity between any two documents.
2. This method is called inside collectionProcessComplete() method of the same class to compare different cosine similarities.
3. We do bonus tasks by calculating **Dice Coefficient** and **Jaccard Coefficient** .

TASK 3 : Computing Mean Reciprocal Rank

1. Inside the collectionProcessComplete() method of RetreivalEvaluator class, we compare cosine similarities of all candidate sentences with respect to query sentence. Using this, we calculate rank for a query ID.
2. All the ranks are stored in an arraylist.
3. Finally, the method compute_mrr() uses the rank arraylist to compute the mean reciprocal rank

TASK 4 and 5: Improving the system

Let us now compare the accuracy and time taken by various strategies we used:

1. Extracting bag of words using delimitors : " .,!?"

```
Score: 0.34749371855330996      rank=1  rel=1 qid=1 Classical music may never be the most popular music
Score: 0.07919922552711958      rank=1  rel=1 qid=2 Climate change and energy use are two sides of the same coin.
Score: 0.36800118055998265      rank=1  rel=1 qid=3 The best mirror is an old friend
Score: 0.20521309615767264      rank=2  rel=1 qid=4 If you see a friend without a smile, give him one of yours
Score: 0.20521309615767264      rank=2  rel=1 qid=5 Old friends are best
 (MRR) Mean Reciprocal Rank ::0.8
Total time taken: 0.714
```

As wee see above, using cosine similarity on simple bag of words extraction using the above said delimitors, we get

1.1 MRR = 0.8
1.2 Total time taken = 0.714 s.

2. Converting all tokens to lower cases

```
Score: 0.34749371855330996      rank=1  rel=1 qid=1 Classical music may never be the most popular music
Score: 0.23759767658135875      rank=1  rel=1 qid=2 Climate change and energy use are two sides of the same coin.
Score: 0.36800118055998265      rank=1  rel=1 qid=3 The best mirror is an old friend
Score: 0.20521309615767264      rank=2  rel=1 qid=4 If you see a friend without a smile, give him one of yours
Score: 0.10540925533894598      rank=1  rel=1 qid=5 Old friends are best
 (MRR) Mean Reciprocal Rank ::0.9
Total time taken: 0.825
```

2.1 MRR = 0.9
2.2 Total time taken = 0.825s

Comparing with approach 1, we see that we get a higher MRR as above and a little more time is taken(due to more processing). However, if closely observed we see that we also get better confidence values for similarity.

For query 2, the similarity score has improved from 0.791 to 0.238, which is a significant improvement.

It is not difficult to see that this is a better approach than the first one.

3. **Using stopwords from the given list 'stopwords.txt'**

```
Score: 0.4524183825710685       rank=1  rel=1 qid=1 Classical music may never be the most popular music
Score: 0.33593757963539017      rank=1  rel=1 qid=2 Climate change and energy use are two sides of the same coin.
Score: 0.3090169943749474       rank=2  rel=1 qid=3 The best mirror is an old friend
Score: 0.11197919321179671      rank=2  rel=1 qid=4 If you see a friend without a smile, give him one of yours
Score: 0.13834459884937497      rank=1  rel=1 qid=5 Old friends are best
 (MRR) Mean Reciprocal Rank ::0.8
Total time taken: 0.915
```

We observe that

3.1 MRR = 0.8
3.2 Total time taken = 0.915s

Comparing with above approach, we see that MRR has reduced. However, the confidence values for all the cases has increased. Eg. For case 1, similarity score has improved from 0.347 to 0.452, etc.

Now, if we observe carefully we see that this has happened only because after removing stopwords from winning sentence 'My best friend is the one who brings out the best in me' , its tokens remain as : { best, friend, one, brings}. This has more in common with the query sentence.

Hence, this is mere luck that we got a wrong answer.

Thus, using lowercases, proper delimiters and using stopwords is the best startegy among all three.

TASK 4 – If we look at only the given test data, best system to improve MRR performance measure is the one that uses lowercase tokens and delimitors as { .,?!}.

However, it can be easily seen that the best system performance-vise is the system that uses lowercase tokens, delimitors as { .,?!} and stopwords list.

TASK 5 – In order to improve efficiency, we do the following:

5.1 Since the first 3 lines of the stopwords document are not being used, we do not input them as stopwords

5.2 We use a hashmap of query ID mapped to the correct query, since there is only one query for a given query ID

5.3 We use a hashmap of query ID mapped to the correct answer sentence, since there is only one correct answer for a query ID

5.4 We re-use the data wherever possible

5.5 We get the data once in a local variable and use it wherever needed.

By doing above, we managed to reduce total time taken by our system from 0.915 s to 0.693s

```
Score: 0.4524183825710685       rank=1  rel=1 qid=1 Classical music may never be the most popular music
Score: 0.33593757963539017      rank=1  rel=1 qid=2 Climate change and energy use are two sides of the same coin.
Score: 0.3090169943749474       rank=2  rel=1 qid=3 The best mirror is an old friend
Score: 0.11197919321179671      rank=2  rel=1 qid=4 If you see a friend without a smile, give him one of yours
Score: 0.13834459884937497      rank=1  rel=1 qid=5 Old friends are best
 (MRR) Mean Reciprocal Rank ::0.8
Total time taken: 0.693
```

Reasons why cosine similarity failed for query 3 and 4

1. **Query 3:**
   As explained earlier, this case fails when we use stopwords. This is because after removing stopwords from winning sentence ‘My best friend is the one who brings out the best in me’ , its tokens remain as : { best, friend, one, brings}. Also, the frequency of the word best = 2 here. This adds to a better confience value for this sentence.

2. **Query 4:**
   The reason is frequency of matching words. It is not necessary that a matching word appearing more than once in a candidate sentence is correct, which has happened in this case.

   Some alternatives to further improve the system:
   1. Synonym capability may be added. We can get synonyms of all words, not present in the stopword list and compare these with the candidate documents.
   2. All the verb forms (eg. Do, does, doing, has done,etc.) may be obtained for all verb forms to improve the serach space for matchilng words.

   To achieve the above, we can use the NLP tool : WordNet.

JAVADOCS

Javadocs for all annotators can be found in resources/javadocs folder of the project.

BONUS

A. Dice Coefficient :

Let us compare the results with cosine similarity when we use Dice Coefficient:

1. Extracting bag of words and using delimitors : " .,!?"

```
Score: 0.1111111111111111       rank=1  rel=1 qid=1 Classical music may never be the most popular music
Score: 0.019230769230769232     rank=1  rel=1 qid=2 Climate change and energy use are two sides of the same coin.
Score: 0.15     rank=1  rel=1 qid=3 The best mirror is an old friend
Score: 0.03418803418803419      rank=3  rel=1 qid=4 If you see a friend without a smile, give him one of yours
Score: 0.0      rank=2  rel=1 qid=5 Old friends are best
 (MRR) Mean Reciprocal Rank ::0.7666666666666667
Total time taken: 0.76
```

We see that:

1.1 MRR = 0.77
1.2 Total time taken = 0.76s

The MRR is lower than that obtained using cosine similarity

2. Converting all tokens to lowercases

```
Score: 0.1111111111111111       rank=2  rel=1 qid=1 Classical music may never be the most popular music
Score: 0.057692307692307696     rank=1  rel=1 qid=2 Climate change and energy use are two sides of the same coin.
Score: 0.15     rank=1  rel=1 qid=3 The best mirror is an old friend
Score: 0.03418803418803419      rank=3  rel=1 qid=4 If you see a friend without a smile, give him one of yours
Score: 0.04     rank=1  rel=1 qid=5 Old friends are best
 (MRR) Mean Reciprocal Rank ::0.7666666666666667
Total time taken: 0.981
```

We observe that:

2.1 MRR = 0.77
2.2 Total time taken = 0.981s

This is again lower than that obtained from cosine similarity and same as before. However, a much higher confidence is achieved when using lower cases (see queries 2 and 5)

3. **Using stopwords from the given list ‘stopwords.txt’**

```
Score: 0.2222222222222222       rank=2  rel=1 qid=1 Classical music may never be the most popular music
Score: 0.125    rank=1  rel=1 qid=2 Climate change and energy use are two sides of the same coin.
Score: 0.16666666666666666      rank=2  rel=1 qid=3 The best mirror is an old friend
Score: 0.041666666666666664     rank=3  rel=1 qid=4 If you see a friend without a smile, give him one of yours
Score: 0.07142857142857142      rank=1  rel=1 qid=5 Old friends are best
 (MRR) Mean Reciprocal Rank ::0.6666666666666667
Total time taken: 0.782
```

We observe that:

3.1 MRR = 0.67
3.2 Total time taken = 0.782 s

Again, this is lower than that obtained from cosine similairty as well as above two methods. Again, this has happened by chance and in general, using lowercases, proper delimitors and stopwords will give a better result.

B. Jaccard Coefficient :

Let us compare the results with cosine similarity when we use Jaccard Coefficient:

1. **Extracting bag of words and using delimitors : “ .,!?”**

```
Score: 0.058823529411764705     rank=1  rel=1 qid=1 Classical music may never be the most popular music
Score: 0.009708737864077669     rank=1  rel=1 qid=2 Climate change and energy use are two sides of the same coin.
Score: 0.08108108108108109      rank=1  rel=1 qid=3 The best mirror is an old friend
Score: 0.017391304347826087     rank=3  rel=1 qid=4 If you see a friend without a smile, give him one of yours
Score: 0.0      rank=2  rel=1 qid=5 Old friends are best
 (MRR) Mean Reciprocal Rank ::0.7666666666666667
Total time taken: 0.699
```

We see that:

1.1 MRR = 0.77
2.2 Total time taken = 0.699s

The MRR is lower than that obtained using cosine similarity

2. **Converting all tokens to lowercases**

```
Score: 0.058823529411764705     rank=2  rel=1 qid=1 Classical music may never be the most popular music
Score: 0.0297029702970297       rank=1  rel=1 qid=2 Climate change and energy use are two sides of the same coin.
Score: 0.08108108108108109      rank=1  rel=1 qid=3 The best mirror is an old friend
Score: 0.017391304347826087     rank=3  rel=1 qid=4 If you see a friend without a smile, give him one of yours
Score: 0.020408163265306122     rank=1  rel=1 qid=5 Old friends are best
 (MRR) Mean Reciprocal Rank ::0.7666666666666667
Total time taken: 0.872
```

We observe that:

2.1 MRR = 0.77
2.2 Total time taken = 0.872s

This is again lower than that obtained from cosine similarity and same as before. However, a much higher confidence is achieved when using lower cases (see queries 2 and 5)

3. **Using stopwords from the given list 'stopwords.txt'**

```
Score: 0.125    rank=2  rel=1 qid=1 Classical music may never be the most popular music
Score: 0.06666666666666667      rank=1  rel=1 qid=2 Climate change and energy use are two sides of the same coin.
Score: 0.09090909090909091      rank=2  rel=1 qid=3 The best mirror is an old friend
Score: 0.02127659574468085      rank=3  rel=1 qid=4 If you see a friend without a smile, give him one of yours
Score: 0.037037037037037035     rank=1  rel=1 qid=5 Old friends are best
 (MRR) Mean Reciprocal Rank ::0.6666666666666667
Total time taken: 0.82
```

We observe that:

3.1 MRR = 0.67
3.2 Total time taken = 0.82 s

Again, this is lower than that obtained from cosine similairty as well as above two methods. Again, this has happened by chance and in general, using lowercases, proper delimitors and stopwords will give a better result.

Overall Conclusion

Cosine Similarity is better than Dice Coefficient and Jaccard Coefficient for measuring similarity between two documents. It gives **better confidence value and MRR**.