11-791 HW4: Engineering and Error Analysis with UIMA

Xiaohua Yan

Language Technologies Institute

1 Task 1: Building Vector Space Retrieval Model using UIMA

In this task, we are required to design a simple vector space retrieval system using the UIMA framework. The vector space model is a widely used search engine model that represents queries and documents as vectors of terms and uses the similarity between these vectors as a model of relevance for document ranking. In the case of this assignment, we need to implement the cosine similarity between a pair of query of retrieved document, based which the **Mean Reciprocal Rank** (MRR) is computed as a measure of the performance of the retrieval system. The MRR is defined as follows:

$$\text{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\text{rank}_i}$$

1.1 Design

In this section I explain in more details about the implementation of the retrieval system including the design of the system and the concrete methods that do the required annotation, computation and so forth.

Format of input data The format of the input file is as follows.

```
qid=1 rel=99 qid=1 rel=0 qid=1 rel=1
qid=1 rel=0
qid=2 rel=99
qid=2 rel=1 qid=2 rel=0
John loves Mary
John and Mary are friends Mary is loved by John Mary likes John
...
...
...
```

It contains information about the query IDs, the relevant values and the corresponding text string. The relevant values of 1 indicates correct retrieval and relevant value of 0 denotes wrong retrieval results. The relevant value of 99 denotes the query itself. For example, "qid=1 rel=99 John loves Marry" means that the text string for qid 1 is "John loves Marry".

Type system The baseline retrieval system I designed has 2 major types: `Document` and `Token`. Table 1 lists the features of each type.

Table 1. Baseline type system for the retrieval system

Type Name	SuperType	Description	Features	Range
Document	Annotation	Annotation type for each query and document	relevanceValue queryId text tokenList	Integer Integer String FSList
Token	Annotation	Annotation type for each token in the document	text frequency	String Integer

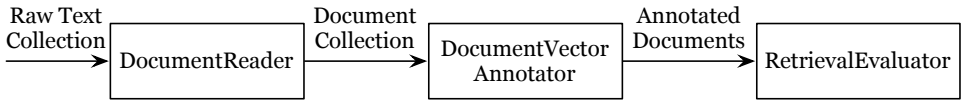
Note that the `queryId` feature of `Document` also identifies the ID of the query to which the retrieved documents correspond. The most important feature of `Document` is `tokenList`, which represents each document as a bag of token types with their corresponding frequency in each document. This feature fits our purpose of computing the cosine similarity between two documents, which is defined by

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{||A||||B||} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i^2)} \times \sqrt{\sum_{i=1}^{n}(B_i^2)}}$$

where A_i and B_i indicate the elements of two vectors A and B, represented by two bags of words of the (query, document) pair.

1.2 Implementation

The pipeline of the retrieval system can be illustrated as follows.



In the first stage of the pipeline, the DocumentReader component reads as input the raw data file that contains a collection of queries and respective retrieved documents. The values of `queryId`, `relevanceValue` and `text` of each document are set by this analysis engine. This information is kept in a CAS for further processing.

Based on the covered text of each document stored in the CAS, the tokens of each document are annotated by the `DocumentVectorAnnotator` analysis engine, which are further used to represent each document as a bag of words by counting the frequencies of tokens and store them as `HashMaps`. Note that punctuations

are omitted when annotating the tokens. And the value of `tokenList` of each document is set after the corresponding `HashMap` is obtained.

The last analysis engine of the pipeline, namely `RetrievalEvaluator`, is a CAS consumer that takes the annotated documents as input and prints out the performance evaluation results based on the MRR scores. In order to improve the efficiency of the CAS consumer, I stored all the necessary indexes and values relevant to queries and retrieved documents in `HashMaps`. For example, I stored the `queryId` of the documents and their corresponding indexes in the document collection as a `HashMap` so that I do not need to iterate over all the documents in the text collection when processing a certain query, assuming no specific order of the queries and documents in the raw input file. This is also the case for the `relevanceValue` feature.

The final output of the baseline retrieval system is as follows:

```
Score: 0.452267 rank = 1 rel = 1 qid = 1 Classical music may never be
the most popular music
Score: 0.102062 rank = 1 rel = 1 qid = 2 Climate change and energy use
are two sides of the same coin.
Score: 0.507093 rank = 1 rel = 1 qid = 3 The best mirror is an old friend
Score: 0.258199 rank = 3 rel = 1 qid = 4 If you see a friend without a smile,
 give him one of yours
Score: 0.000000 rank = 2 rel = 1 qid = 5 Old friends are best
 (MRR) Mean Reciprocal Rank ::0.7666666666666667
Total time taken: 0.701
```

2 Task 2: Error Analysis

As can be observed from the final output of Task 1, there are two errors for Query 4 and 5, whose ranks are 3 and 2 respectively. In this section I discuss the error analysis I did in order to overcome these errors.

2.1 Lowercasing and Stemming

As for any NLP task, it is instrumental to do some preprocessing for the input text collection. For instance, **stemming** is likely to be helpful because it preserves the lemma of each token which eliminates the difference between tokens with the same lemma such that similarity can be computed more effectively in the sense of bag-of-words assumption. This is especially the case for the 5th query "It takes a long time to grow an old friend" with its correct retrieved document being "Old friends are best". In the baseline retrieval system, the cosine similarity between the correct document and the corresponding query is 0 because there is no token overlap between them if there is no preprocessing. However, it is clear that "Old friends" in the document is highly similar with "old friend" in the query text.

Therefore, I implemented lowercasing and stemming of the input text using Morpha Stemmer (http://mvnrepository.com/artifact/edu.washington.cs.knowitall/morpha-stemmer) while computing the cosine similarity in the same way as in Task 1. The output is as follows, with an improved MRR score being 0.8.

```
Score: 0.603023 rank = 1 rel = 1 qid = 1 Classical music may never
be the most popular music
Score: 0.306186 rank = 1 rel = 1 qid = 2 Climate change and energy
 use are two sides of the same coin.
Score: 0.507093 rank = 2 rel = 1 qid = 3 The best mirror is an old
 friend
Score: 0.344265 rank = 2 rel = 1 qid = 4 If you see a friend without
 a smile, give him one of yours
Score: 0.316228 rank = 1 rel = 1 qid = 5 Old friends are best
 (MRR) Mean Reciprocal Rank ::0.8
Total time taken: 0.722
```

One interesting thing to note is that though the rank of Query 5 is improved, the rank for Query 3 degrades by 1 at the same time, which indicates that further improvement is needed.

2.2 Stop Words Removal

Another possible way to improve the performance of the retrieval system is to remove the stop words (e.g. "in", "of", etc.) before the similarity is computed for a pair of documents. This is also a commonly used practice in NLP tasks since stop words in the text produce noise while the removal of them does little harm to the original semantics. Thus, I implemented stop words (without stemming) removal based on the list of stop words provided in the resource folder, and the final output is as follows, with an improved MRR score of 0.8.

```
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: 0.500000 rank = 2 rel = 1 qid = 3 The best mirror is an old
 friend
Score: 0.182574 rank = 2 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 ::0.8
Total time taken: 0.729
```

Clearly, the final MRR is the same as that after doing stemming.

2.3 Combining Stop Words Removal and Stemming

Since both the removal of stop words and stemming have been approved effective, one intuition is that combining these two approaches can further improve the performance. Accordingly, I implemented these approaches at the same time, which produces the following 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: 0.500000 rank = 2 rel = 1 qid = 3 The best mirror is an old
 friend
Score: 0.365148 rank = 1 rel = 1 qid = 4 If you see a friend without
 a smile, give him one of yours
Score: 0.471405 rank = 1 rel = 1 qid = 5 Old friends are best
 (MRR) Mean Reciprocal Rank ::0.9
Total time taken: 0.729
```

At this time, the rank of Query 4 improves by 1, which further improves the final MRR to be 0.9, though the rank of Query 3 still did not change.

2.4 Modification of Similarity Measure

Since the only error left by now is for Query 3, it is necessary to examine the reason of this error. Taking a closer look at the location of the error, the cosine similarity of the correct document is slightly slower than a competing document merely because the incorrect one has one more count of the lemma "best", while the original incorrect document is much longer than that of the correct one. Therefore, further performance improvement is possible by tailoring the similarity measure to place less emphasis on document length. One such way to do this is to improve the magnitude of the denominator of the cosine similarity. For example, the modified cosine similarity can be defined as

$$\text{modified cosineSim} = \frac{A \cdot B}{||A||||B||} = \frac{\sum_{i=1}^{n} A_i \times B_i}{(\sum_{i=1}^{n}(A_i^2) \times \sum_{i=1}^{n}(B_i^2))^{3/4}}$$

In fact, I implemented this modified similarity measure, and obtained the final output as follows:

```
Score: 0.276670 rank = 1 rel = 1 qid = 1 Classical music may never
be the most popular music
Score: 0.181838 rank = 1 rel = 1 qid = 2 Climate change and energy
 use are two sides of the same coin.
Score: 0.250000 rank = 1 rel = 1 qid = 3 The best mirror is an old
 friend
Score: 0.156023 rank = 1 rel = 1 qid = 4 If you see a friend without
 a smile, give him one of yours
```

```
Score: 0.228863 rank = 1 rel = 1 qid = 5 Old friends are best
 (MRR) Mean Reciprocal Rank ::1.0
Total time taken: 0.718
```

Finally, the enhanced retrieval system achieved an MRR score of 1.0 on the testing data.

3 Discussion and Conclusion

In this assignment, I implemented the baseline retrieval system as required, with good design patterns and implementation. I did an in-depth error analysis, which finally helped the retrieval system obtain an MRR score of 1.0 on the testing data. Admittedly, the tuning of the system during error analysis may lead to over-fitting when new testing data is available. However, I believe better performance of the system can still be obtained with more carefully designed similarity measure and ranking strategies, which I will leave for future work due to time limitation.