Engineering and Error Analysis with UIMA

Name: Xiang Li Andrew ID: xiangl2 E-mail: lixiangconan@gmail.com

1. Task Requirement

In this task, we need to use UIMA framework to implement a simple vector space retrieval system. Specifically, the vector space model would be used so that query and document can be represented by vector of terms. Then we can use cosine similarity measure to retrieve suitable answer. Finally, the system should compute the Mean Reciprocal Rank (MRR) metric for tracing retrieval performance.

2. Overall Structure

Using UIMA framework, the whole system should be composed of three parts:

- Document Reader
- Document Vector Annotator
- Retrieval Evaluator

Document Reader reads input documents from file system and does some simple processing. Then document Vector Annotator adds more annotations preparing for vector space retrieval. Finally, Retrieval Evaluator converts each document to a vector, does similarity search, and measures system performance using MRR.

3. Components Details

3.1 Document Reader

In general, we hope that Document Reader could deal with input documents from different source and accept different input formats. However, for this task, we just assume that all input documents come from a single folder and each document only contains one sentence. So the Document Reader could simply read input document from given directory, then split it into three parts, which are query ID, reference value and text string, and finally store information in Document annotator for each document.

3.2 Document Vector Annotator

After obtaining Document annotator for each input file, we should do further analysis on it so that we can represent it using vector of terms. In Document Vector Annotator, we first tokenize the text string by Stanford CoreNLP library. Then, we can count the occurrence frequency for each term using a hash table. Finally, we store terms and corresponding frequencies for each document in its token list.

3.3 Retrieval Evaluator

The Retrieval Evaluator mainly contains two part: the first part save necessary

information from the JCAS object after the system analysis each document using Document Vector Annotator, and the second part summarizes all the information we get and outputs a suitable answer for each query after the system processed all the input documents.

In the first part, nearly all the information in the JCAS object would be saved, since all of them are useful for further retrieval procedure. Such information includes query ID, reference value, text string and token list. In this part, the system also adds all the terms appear in each document to a global dictionary.

In the second part, the system need to do several things in order. Firstly, each document is converted to a vector according to its token list. The dimension of the vector would be the same as the number of terms in the global dictionary. Secondly, for each query ID, the system calculates the cosine similarity between the query document and all the candidate documents. Since each element in the vectors is non-negative, the cosine similarity should fall in the range of [0, 1]. Thirdly, the system orders the candidate documents for each query according to their cosine similarity to the query document. A candidate document having relatively smaller cosine similarity to the query document would be given a relatively higher rank, which means that it has higher probability to be is a right answer. Finally, the system outputs all the rank 1 documents as the retrieval results, and calculate MRR to measure its performance.

4. Error Analysis

Using the given input file, the system could generate following result:

```
Score: 0.45226701686664544   rank=1   rel=1  qid=1 Classical music may never be the most popular music
Score: 0.09805806756909201   rank=1   rel=1  qid=2 Climate change and energy use are two sides of the same coin.
Score: 0.4629100498862757   rank=1   rel=1  qid=3 The best mirror is an old friend
Score: 0.25   rank=3   rel=1  qid=4 If you see a friend without a smile, give him one of yours
Score: 0.0   rank=3   rel=1  qid=5 Old friends are best
 (MRR) Mean Reciprocal Rank ::0.7333333333333334
```

According to this result, we can see that the correct answers were given the highest rank for the first three queries. However, for the last two queries, the correct answers were given the lowest answer. So we need to find why the system doesn't perform well on the last two queries.

For the query 5, the question is:

```
qid=5   rel=99  It takes a long time to grow an old friend
```

The candidate answers are:

```
qid=5   rel=0   Old wine and friends improve with age
```

```
qid=5   rel=0   With clothes the new are the best, with friends the old are the best
qid=5   rel=1   Old friends are best
```

According to the system, the cosine similarity from these three answers to the question are 0.0, 0.0568, and 0.0. This is quite a surprising result, because the cosine similarity between the correct answer and the question is only 0.0! However, as human beings, we can easily find that there are many similar parts, such as “Old” (“old”), “friends” (“friend”) in the correct answer and the question. Then we know that the system makes mistakes because there are many different forms of a word, but a computer cannot automatically link them together. Therefore, to improve the performance of the system, we can convert each word to its base form before doing other process.

In addition to this, we also find that there are other kinds of useless information in the token list and dictionary. The dictionary for these five queries is:

```
[they, Behind, influences, of, time, are, Everybody, Old, role, smile, dying, wine, ., without, ,, ;, Wear, My, put, give, most, use, mirror, wrinkles, The, distance, absorbed, best, me, One, With, may, a, brings, new, 's, to, old, shortest, change, plays, him, same, every, long, has, who, takes, hear, friends, age, when, climate, be, clothes, oneself, and, knows, classical, popular, scowl, If, friend, other, genres, have, one, improve, from, It, Classical, between, sides, antiques, yours, music, girls, with, is, it, coin, important, Climate, you, the, in, two, never, Pop, see, Energy, an, there, grow, wear, energy, out]
```

Observing it, we find that some words in the dictionary are just different forms of one word, while other words, such as “.”, “.”, “is” and “the” are actually meaningless for comparing the similarity between two sentences. Therefore, we can also use a stop words list to filter the tokens before building the dictionary. Using this method, we are hopeful to get a dictionary with fewer noises, which also helps are to generate better feature vectors.

5. System Improvement

5.1 Stemming and Stop Words List

After doing the error analysis, we find that we can improve the performance of our system by two ways: the first way is converting each word to its base form; the second way is using stop words list to filter the tokens. Since these two ways are irrelevant with generating the annotations, we can simply update the Retrieval Evaluator to improve the performance of the system. In practice, to make the comparison more convenient, we build a new Retrieval Evaluator, and run both the new evaluator and the original evaluator in the pipeline.

In general, this new Retrieval Evaluator is similar to the original one, except that it uses stemming and stop words list for preprocess. That is to say, for each document,

we firstly use stemming method offered by Stanford CoreNLP library to convert its tokens to the base form. After that, we remove the tokens that are in the stop word list. Then, we use the rest of the tokens to generate the global dictionary and feature vector for each document. Finally, we calculate the cosine similarity and rank the answers using the same method we as we use in the original Retrieval Evaluator.

5.2 Experiment Results

We test our new Retrieval Evaluator with the test data we used before. This time, we get a much smaller global dictionary, which is shown below:

```
[genre, old, shortest, change, pop, bring, long, every, time, hear, age, role, everybody, smile, dying, climate, wine, clothes, oneself, behind, without, ha, classical, popular, scowl, antique, friend, put, give, one, influence, use, improve, mirror, distance, girl, side, absorbed, best, music, may, important, coin, play, know, two, wrinkle, never, new, see, take, grow, wear, energy]
```

Apparently, this dictionary has less meaningless words. Because of this, the Mean Reciprocal Rank is also better than the original one:

```
Score: 0.6123724356957945   rank=1   rel=1  qid=1 Classical music may never be the most popular music
Score: 0.4629100498862757   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: 0.3651483716701107   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
```

According to this result, we can see that the system now perform well on the last two queries, as well as on the first two queries. So the global performance of the new Retrieval Evaluator is better than the original one. However, we also recognize that for the second query, the original pipeline gives the correct answer rank 1, while the new pipeline gives the correct answer rank 2. This means that the new pipeline still has its drawback, and we can continue improve it using other methods.