Hw4 Documentation, LTI 11791

Abhimanu Kumar
Andrew Id - ABHIMANK

November 6, 2013

1 Main Task

I first created a simple vector space retrieval model and then further improved it by performing detailed error analysis.

1.1 Basic System

I describe below the provided barebone classes that were modified further to :

1. DocumentVectorAnnotator: This is where I create the tokens and counts for the strings present in the query and documents. I created a token list and stored it as an FSList.

2. RetrievalEvaluator: This is the biggest logical unit of the code for this assignement. Here I find the cosine similarity between the query and the answer string. Inside each token a string is stored as its text and the term frequencies and then each question and answer has a vector of tokens. The similarity between query and answer is calculated by computing the cosine similarity between the respective question and the answer token vectors.

The resulst for the above basic system are:

- Classical music may never be the most popular music. Score: 0.452 rank=1 rel=1 qid=1
- Climate change and energy use are two sides of the same coin. Score: 0.098 rank=1 rel=1 qid=2
- The best mirror is an old friend Score: 0.462 rank=1 rel=1 qid=3
- If you see a friend without a smile, give him one of yours. Score: 0.25 rank=3 rel=1 qid=4
- Old friends are best Score: 0.0 rank=2 rel=1 qid=5

(MRR) Mean Reciprocal Rank ::0.7666666666666667

1.2 Analysis

The last sentence above can be made to have non-zero cosine score by making all strings lower case; two different strings "Old" and "old" appear in answer and query respectively but they syntactically the same. Following are similar modifications to achieve similar improvements:

Lemmatization of tokens: Many of answer strings contain words that are morphological variants of query words. For example, the query contains the word 'love', whereas the answer contains the phrase 'loves'. Lemmatizing the string increases the cosine similarity. Stanford NLP toolkit is used for this. Although, the cosine similarity increased, it didnt lead to any increase in the MRR. The reason behind this is because both the correct and the incorrect answer strings have the same number of occurrences of the lemma of the string.

Stop words: Removing stop words intuitively means removing unimportant words. It also decreases the effective sentence length thus in turn increasing the cosine similarity. The cosine similarity increased, but still the MRR was same. This is because the candidate answer strings were similar in length and contained same number of stop words.

Removing punctuation Removing punctuations from both query and answers increases the MRR. The new score is 0.80

Converting tokens into lower case: Converting all the strings to lower case in answers as well as queries increases MRR to 0.90 .

Hence we see that more intelligent preprocessing increases the MRR from 0.76 to 0.90

2 Bonus

Besides Cosine Similarity, I impelemented Jaccard similarity between query and corressponding answers:

$$Jaccard(Q, A) = \frac{|Q \cap A|}{Q \cap A}. \tag{1}$$

The MRR without all preprocessing is 0.80, a decrease of 0.10.