Mario Piergallini
mpiergal

Homework 4 Report: Engineering and Error Analysis with UIMA

For this assignment, after the initial setup (running JCasGen, etc.), the first step was to fill out the DocumentVectorAnnotator. This was fairly simple. First, I put the stop words into a HashSet. Then I used a simple regular expression to find tokens (in this case, being sequences of alphanumeric characters or apostrophes). I skipped over the token if it was in the stop words set. For each token, I put it into a HashMap with the word as the key and the number of times it occurred as the value. The term frequency was increased if the same word was encountered again. Afterwards, I used the Utils to convert this HashMap of tokens into an FSList.

After that, it was necessary to fill out the RetrievalEvaluator. First I created two classes. One was the Sentence which contained the text of the sentence, the qID, relevance value, the list of words in the document with their frequencies and a cosine similarity variable. The second was the QIDSet which simply contained the query and the list of answers associated with it. Then I populated a list of sentences using the FSIterator. At the same time, using the tokenList from each document, I generated a HashMap for the vocabulary, with the overall frequency for each word. I also put those values into the Sentence. Then I mapped each document to a HashMap with the qId associated with the QIDSet.

To calculate the cosine similarity measure, I used tf-idf instead of simple term frequencies. So first I calculated the idf for each term by cycling through the vocabulary and counting the number of sentences that contained the word. Then for each answer sentence, I simply summed the product of the tf-idf for the words that occurred in both documents to get the numerator. To get the divisor, I individually summed the squares of the tf-idf values for all the words in the query and answer sentence. Then I took the square root of those sums and multiplied them. The cosine similarity is then just the numerator over the divisor.

After this, I ranked the answers for each query according to this cosine similarity using a simple insertion sort. Then I calculated the mean reciprocal rank by iterating through the query IDs, then going through the sorted list of answers until I found the correct one, and summing the inverse of the ranks of the correct answers. Then I took the average. At the same time, I printed out the scores and ranks of the correct answers.