Assignment 4
11-791: DEIIS

Submitted by: Ankur Gandhe, ankurgan

October 28, 2013

1 Task 1

The task of this assignment is to create a vector space retrieval model using UIMA.

1.1 Basic System

Below we define the "baseline" system's architecture and implementation:

1. Document Reader: Document reader was given pre-implemented to annotate documents in the test file. We did not make any changes to it

2. DocumentVectorAnnotator: For each document present in the cas, we created token annotations. First, the text was tokenized using a pattern matcher. Next, the word text and their frequencies were obtained by creating a Token $\rightarrow$ frequency HashMap. In the end, token annotations were created and added as Token FSList to the document. The TokenFSList then becomes the vector of this document

3. Document Score class: We created a cas consumer class called DocumentScore for easy storing and ranking of documents

4. Retrieval Evaluator(Cas comsumer): To process the cas, we get the document annotation and create Token $\rightarrow$Frequency map for each document. We also get the relevance value for it. All this information is stored in a DocumentScore object and added to a list . When all documents have been processed, the list of documents is traversed and the rank of relevant document for each qid is computed by computing the cosine similarity of query token vector with document token vectors. One we finish running through all qid, the final MRR is computed.

1.2 Error Analysis

The performance of this system is good, but not great. We get the rank of 1 for the first three but a rank of 3 and 2 for the last two queries. The final output looked like:

```
Score: 0.452 rank=1 rel=1 qid=1 Classical music may never be the most popular music
Score: 0.098 rank=1 rel=1 qid=2 Climate change and energy use are two sides of the same coin.
Score: 0.463 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=2 rel=1 qid=5 Old friends are best
(MRR) Mean Reciprocal Rank ::0.767
```

The performance was low for last two queries due to several reasons:

- The tokens were not being lower cased before construction of Tokens. Hence, in our next iteration, we lower cased our words and then used them to construct the document vectors. Doing so, yielded a system with same MRR but the errors appeared in different sentences

  ```
  Score: 0.452 rank=1 rel=1 qid=1 Classical music may never be the most popular music
  Score: 0.294 rank=1 rel=1 qid=2 Climate change and energy use are two sides of the same coin.
  Score: 0.463 rank=2 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: 0.158 rank=1 rel=1 qid=5 Old friends are best
  (MRR) Mean Reciprocal Rank ::0.767
  ```

  As we can see, the rank of the fifth answer improves, but the rank of third answer goes down. Hence, it seems like a combination of the two approaches would work in this example file. However, it might lead to over-fitting and we must look for other document vectors.

- We are not using stop word lists here. We modified our pipeline to read the stop words list and easily check to discard stop words from the document vector. This, however, proved detrimental and we decided to not use stop words.

- Other similarity measure: We tried using Jaccard similarity and it seemed to give same MRR for the above 2 cases. But the scores of documents with qid were much more discriminative and hence we decided to use JaccardSimilarity for our final system.

- Using Lemma: Instead of using tokens as is, we could use document vectors formed out of word lemmas and then compute similarity. Using lemma vectors, we again got a similar MRR :

  ```
  Score: 0.231 rank=1 rel=1 qid=1 Classical music may never be the most popular music
  Score: 0.143 rank=1 rel=1 qid=2 Climate change and energy use are two sides of the same coin.
  Score: 0.231 rank=2 rel=1 qid=3 The best mirror is an old friend
  Score: 0.13 rank=3 rel=1 qid=4 If you see a friend without a smile, give him one of yours
  Score: 0.154 rank=1 rel=1 qid=5 Old friends are best
  (MRR) Mean Reciprocal Rank ::0.767
  ```

  Using lemmas, the obtained scores seem much more smooth across different qid's and hence a combination of lemma score and lower cased tokens should be a good final metric.

1.3 Final Architecture

Most of the initial pipeline (DocumentReader) remain the same.

1. DocumentVectorAnnotation: We use stanford CoreNLP package for doing tokenization and lemmatization of the input document. The token text, frequency, and lemma are stored in Token Annotations

2. Utils: To keep the mathematical components separate, we create functions to compute CosineSimilarity. JaccardSimilarity and WeightedScoreCombination in the Utils.java

3. RetrievalEvaluator: Like before, the retrieval evaluator reads the documents in series and stores them in a list of DocumentScore object. The document score object now contains the document vector for tokens, lower-cased tokens and lemmas. This way, this architecture can be easily extended to include other kinds of document vectors, or document information if required. The retrieval process then proceeds as described before. The only difference is in scoring. The final score for each query-document similarity is now a combination of the 3 scores : token similarity, Lowered token similarity and token Lemma similarity.

Using the best combination scores, we obtain a MRR of 0.867 :

```
Score: 0.209 rank=1 rel=1 qid=1 Classical music may never be the most popular music
Score: 0.143 rank=1 rel=1 qid=2 Climate change and energy use are two sides of the same coin.
Score: 0.231 rank=1 rel=1 qid=3 The best mirror is an old friend
Score: 0.118 rank=3 rel=1 qid=4 If you see a friend without a smile, give him one of yours
Score: 0.13 rank=1 rel=1 qid=5 Old friends are best
(MRR) Mean Reciprocal Rank ::0.867
```