Homework 3
11-791

Name: **Shriphani Palakodety**
Andrew ID: **spalakod**

October 22, 2014

1 Task Description

1.1 Task I

In this task, we implemented a simple Information Retrieval (IR) system. For the implementation in Task 1, we used a vector-space representation for queries and documents. The coefficients were the terms and the weights on the coefficients were the term-frequency components. Similarity between document and query was computed using the cosine similarity. The resulting MRR was 0.4375.

1.2 Task II

In this task, I did error analysis and implemented additional modules to address some of the errors. The following sections describe the error analysis and the newer implemented components.

2 Error Analysis

The original implementation had 2 major components that influenced IR scores: (i) Document representation; (ii) Query-Document similarity computation. Thus, the set of errors possible fall into the large categories. In terms of representation, there is the tokens themselves and the weights assigned to them that matter.

The error categories thus are (i) tokenization; (ii) similarity-function and (iii) other.

Ideally, a correct tokenization should be robust to variants of the token in query and documents (for example `vandalize` and `vandalized`) should contribute to the same underlying token (achieved via stemming or lemmatizing for example). The tokenization should discard punctuation and the like in documents and retain clean and simple versions of the base tokens.

The token-weights are important since a poor weighting scheme can ruin any benefits that are achieved with a good similarity function.

Tokenization	`1, 5, 6, 7, 8, 11, 12, 13, 14, 17, 18, 20`
Similarity	`3, 4, 9`
Other	`2, 15, 16, 19`

Table 1: Error analysis of queries

The similarity function-related errors are due to a poor choice of similarity functions.

The "other" category exists for errors that could not be easily ascribed to either tokenization or similarity function limitations.

Table 1 divides up the errors in the provided dataset into these categories.

As we can see, by addressing tokenization, we can hope to improve system performance. Improving tokenization is thus (i) simple to implement; (ii) is estimated to provide a significant performance boost. A tokenizer that (i) splits on spaces, punctuation, and digits (ii) stems the resulting tokens was deployed. The resulting MRR was 0.6417.

It is not entirely clear how to ascribe errors to the weighting category. To check if any improvement could be obtained, two similarity functions were implemented (i) cosine-similarity with TF-IDF (ii) Jaccard similarity.

TF-IDF weighting weighs down the TF component in the document-vectors by how common the term is in the entire corpus. Using TF-IDF weighting, a term's coefficient in the document vector is going to be `tf * log(N)/(1 + df)` where `N` is the number of documents in the corpus and `df` is the number of documents in the corpus that contain the term in question.

Using TF-IDF coefficients and cosine similarity and using the tokenization mechanism from part 1, the resulting MRR was 0.5875.

Jaccard similarity is defined in terms of set operations. In particular, the similarity between two documents is computed as follows. Let the set of terms of the first document be A and that of the second be B. Then the Jaccard similarity between the two documents is given by $\frac{|A \cap B|}{|A \cup B|}$

Using Jaccard similarity the resulting MRR was 0.5958. Clearly the similarity functions employed reduced the performance. This is possible because IR similarity functions are designed for larger datasets that typically exhibit Zipfian characteristics. Possibly those don't apply to this dataset and we don't see any evidence of improvement.

3 Design

In this section, I will explore some patterns for designing an information retrieval system. The canonical example of such a system is one from Google - Map-Reduce. The Map phase applies a function to a list of objects (thus this step is parallelizable). The Reduce phase is analogous to a Σ operation.

In our setting, the Map-Reduce framework can be leveraged to first build document-vectors that only compute TF in the map stage. And then compute

the DF terms in the Reduce phase (details available in the original pagerank paper).

This technique could be leveraged in UIMA quite easily (parallelize the CAS annotators and implement the reduce phase in the CAS consumer). This allows us to fit several patterns (for example, since the MAP phases are responsible for just computing TF values and the Reduce phase for the rest of the computation, the system thus delegates responsibility like GRASP pattern encountered in class).

Simple frameworks like this have been tried and tested several times. Thus the MR framework is a good pattern that can be leverage for Vector-space IR settings.

4 Links

The results for the additional implemented methods are available online (clickable links):

- Better tokenization: `https://gist.github.com/shriphani/c84eee099a9b1bda9a2e`
- Jaccard Similarity: `https://gist.github.com/shriphani/8ebfbd3645c1aee28de2`
- Cosine similarity with TF-IDF weighting: `https://gist.github.com/shriphani/ec5e974145ffeade3bc9`