Angela Santin Ceballos
Student #: s0839470
06/10/2014

TEXT TECHNOLOGIES ASSIGNMENT 1

Introduction

This report describes the implementation and results for the following three algorithms for document-query similarity computation and ranking:

1. Overlap: uses the overlap between non-capitalized words to rank the documents
2. Tf-idf: uses tf-idf as the weighting function for words and the weighted sum as the similarity function between the document and the query s(Q, D)
3. Rocchio: As in 2, it uses td-idf and the weighted sum formula for an initial ranking, but then optimizes the original query using Rocchio's algorithm

Description of the Algorithms

Overlap

For the simple overlap algorithm, I checked whether each word in the query was also in the document. If it was, the overall weight of this particular document was increased by one. The greater the document weight, the greater the document relevance ranking. Hence, the more words shared between the query and a document, the higher the ranking for that particular document will be.

tf-idf

The tf-idf algorithm was implemented similarly to overlap but using the tf-idf formula. This formula allows us to compute the importance of the appearance of a word in a document more reliably. It does this by taking into account how long the document is, and how common the word is over the given corpus. This raised the mean average precision from the initial 15.27% to 32.42%, as expected.

Rocchio's Pseudo-relevance Algorithm

Before applying Rocchio's algorithm I used Python's NLTK module to stem all of the words in the query, Q_0, and the documents into stems. Also, the technical level of the documents suggested that domain-specific terms would be more indicative of similarity than other universally common words. Thus I decided that removing all stop words (common words) would help reduce dimensionality while not affecting precision by too much.

Once I had the set of stems, I used tf-idf to weigh every one within each document. I used the weighted sum formula to calculate s(Q_0, D) for each documents over all of its tokens. I used s(Q0, D) to rank the documents.

Angela Santin Ceballos
Student #: s0839470
06/10/2014

To implement Rocchio's:

1. Ideally a user would indicate which documents are relevant to their query. However, in the absence of a user, I identified the R top ranked docs based on their s(Q, D) and called this set the Relevant document set (R).
2. For the documents in R, I calculated the tf-idf for every stem in the document in isolation (not relative to any query)
3. For each stem I summed the weights over all relevant documents and divided that sum over |R| to find the average stem weight. Putting all stems together I obtained a new query vector, Q', which contains the average weight of each stem across relevant documents for a particular query
4. I added the weights of Q' to my original Q_0 to obtain Q_1, the enhanced query, in this manner:

   $$Q1 = A*Q\_1 + B*Q'$$

   where A and B are parameters that I optimized empirically
5. Finally,I used the enhanced query, Q1, as the new query. I ran tf-idf with the weighted sum formula on Q1 and the entire document set.

Results and Discussion

Algorithm	**Best MAP obtained (%)**
Overlap	15.27
tf-idf + weighted sum	32.42
Rocchio's Algorithm (A = 4, B = 8, R = 13)	50.34

As shown above, implementing Rocchio's resulted in a considerable improvement in the MAP value compared to the other two simpler approaches. The parameters A and B were taken to be equal to 4 and 8, as suggested by the textbook based on evaluation on relevant academic papers. The value of R was found empirically to be 13 documents.

Although Rocchio's algorithm often includes a non-relevant set of documents as well, I found that discounting the weights of non-relevant documents did not make the algorithm perform significantly better but slowed down performance considerably. Thus I decided that it was best to incorporate the weights of the relevant set only into the expanded query.

The algorithm could be improved using Latent Semantic Analysis to reduce the dimensionality even further (but we haven't learnt it yet).