Text Technologies for Data Science: Assessment 1

s1107496

October 6, 2014

1 Introduction

This report describes the implementation of two information retrieval algorithms, overlap.py and tfidf.py, as well as refinements to the latter as implemented in best.py.

2 overlap.py and tfidf.py

2.1 overlap.py

In overlap.py, each query and document was first transformed to lowercase and tokenized, with tokens matching the simple regular expression `[A-Za-z0-9]+`. For every combination of document and query, the score returned was the size of the intersection of the token sets for the document and query. This method resulted in an average precision of 0.1527.

2.2 tfidf.py

tfidf.py implemented the standard tf-idf algorithm, with k=2 and using the natural logarithm. For speed, df-scores for each token were pre-calculated. The tf-idf algorithm resulted in an average precision of 0.3248.

3 best.py

best.py included a number of refinements over tfidf.py, as listed below. The average precision achieved by best.py was 0.3508.

3.1 Stop word removal

After initial tokenization using the same regular expression as overlap.py, all tokens from the English-language stop word list provided by `nltk` were removed. This was performed for both queries and documents, and increased average precision by 0.0088.

3.2 Character n-grams

To account for morphological variation, each token was broken into constituent n-grams, which were then themselves added to the token list. This procedure was performed for both queries and documents. As stated in the 29th September, 2014 Text Technologies lecture, character n-grams of length 4 or 5 work best for European languages - in this case, 4-grams yielded an average precision advantage of 0.003 over 5-grams. Adding both 4- and 5-grams decreased average precision. Overall, this method increased average precision by 0.0066.

3.3 Bigrams

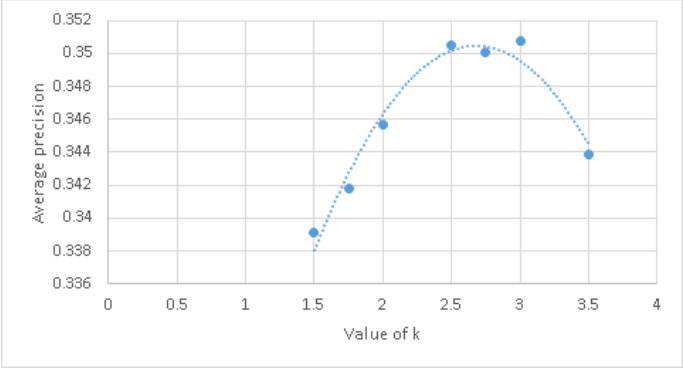
In both documents and queries, to capture meaningful pairings of words, bigrams from the stop-word-sanitized token list were themselves added to the token list. This improved average precision by 0.0033, and was found to be more effective than trigrams or both bigrams and trigrams.

3.4 Extended token set

For both documents and queries, the text was additionally tokenized via the regex `[A-Za-z0-9\.\-\_\/]+`, which preserves e.g. the tokens "I.B.M." and "n-tuples". These extended tokens were then added to the token list, if not already present, with all punctuation intact. This resulted in a precision increase of 0.0022.

3.5 `k` optimization

The value of `k` within the standard formulation of tf-idf was hand-optimized to a value of 3.0, yielding a 0.0051 increase in average precision. The figure below demonstrates the observed effect of k on precision.



3.6 Unsuccessful techniques

The following techniques were implemented, but uniformly had a negative effect on average precision and, hence, were discarded:

- Weighting references - A scheme whereby documents had their tf-idf score slightly increased proportionally to the scores of all documents that referenced them reduced average precision to 0.1949.
- Synonymy - For each token, synonyms of that token were calculated via the Expected Mutual Information Measure as presented in the Text Technologies lecture on 29th September 2014. Tokens were considered synonymous when the EMIM of the two terms was 2 standard deviations above the average. For performance reasons, EMIM was only calculated for tokens appearing in 10%-0.67% of documents. For each query, all tokens with synonyms had their synonyms added to the token list. Overall, this led to a .0090 reduction in average precision, with a significant increase in computation time.