Text Technologies – Exercise 2

s0840449

1 Decisions

An examination of the queries and documents showed two clear problems to be dealt with: the frequent misspellings of words in documents, and the far more frequent use of non-English words in documents. As the queries all used correctly spelt English words, this meant that there was a very low overlap between the words in a query and the words in the documents. To correct this issue, I could either 'fix' the documents to have correctly spelt, English words, or try to expand the queries to include relevant words such as misspellings and translated versions of the query words.

As spelling correction and (especially) translation are both very difficult and very expensive to compute, I choose to try and expand the queries instead. The obvious method for doing this was Pseudo-Relevance Feedback (PRF), which expands a query with high scoring words from relevant documents, and then re- runs the query to select the final set of relevant documents.

To rank the documents in the initial and final stages I choose to build on top of the tf.idf model, as it had reasonable performance even with the original small number of similar words. My first approach for selecting words from the relevant documents was to rank the words by their document-only tf.idf, and then select the top words over the whole set of relevant documents. I tested a few methods to choose how many *instances* of each high-scoring word to select, and found that adding only one instance per word gave me the best average precision (MAP), of around 0.65.

To get a better ranking for words which appeared only once or twice in each relevant document, I tried combining the relevant documents in a 'mega- document' before calculating and selecting words. I also expanded the query vector with tf.idf scores for each of the words (including the original query words, if they ranked highly enough) as opposed to unique instances. This improved the MAP to around 0.73.

The final tuning I applied was to search over a space defined by $\{0.1 \leq k \leq 3.0,\ 15 \leq n_d \leq 30,\ 2n_d - 10 \leq n_w \leq 2n_d + 10\}$ for the area of highest average precision (MAP). The results of the search can be seen in Figure 2. This search showed that there was a large area of 'good' MAP value (above 0.7), with a peak performance of 0.78 at $k = 0.75$, $n_d = 19$, $n_w = 38$. As the search space around this area was generally good and did not fall off sharply, I felt that it was acceptable to use these values for the default constants without worrying too much about overfitting the data.

An important performance-based design decision that should be mentioned was to build an inverted index from words to the documents that contained them. This allowed me to examine only documents with some overlap for each query, as opposed to scanning all 4500 documents.

2 Relative Performance

The interpolated recall/precision plot for the overlap, tf.idf, and my 'best' method can be seen in Figure 1. As can be seen, the best method strictly dominates all of the other algorithms at any recall value, which means that it should always be used over the other methods, regardless of whether you have a requirement for low recall or not. It also retains good precision for much higher recall values, only falling below 0.6 precision at around 0.85 recall. This constrasts with the other algorithms, which decay semi-exponentially. The result is that even in a high-recall situation, my 'best' algorithm performs well.

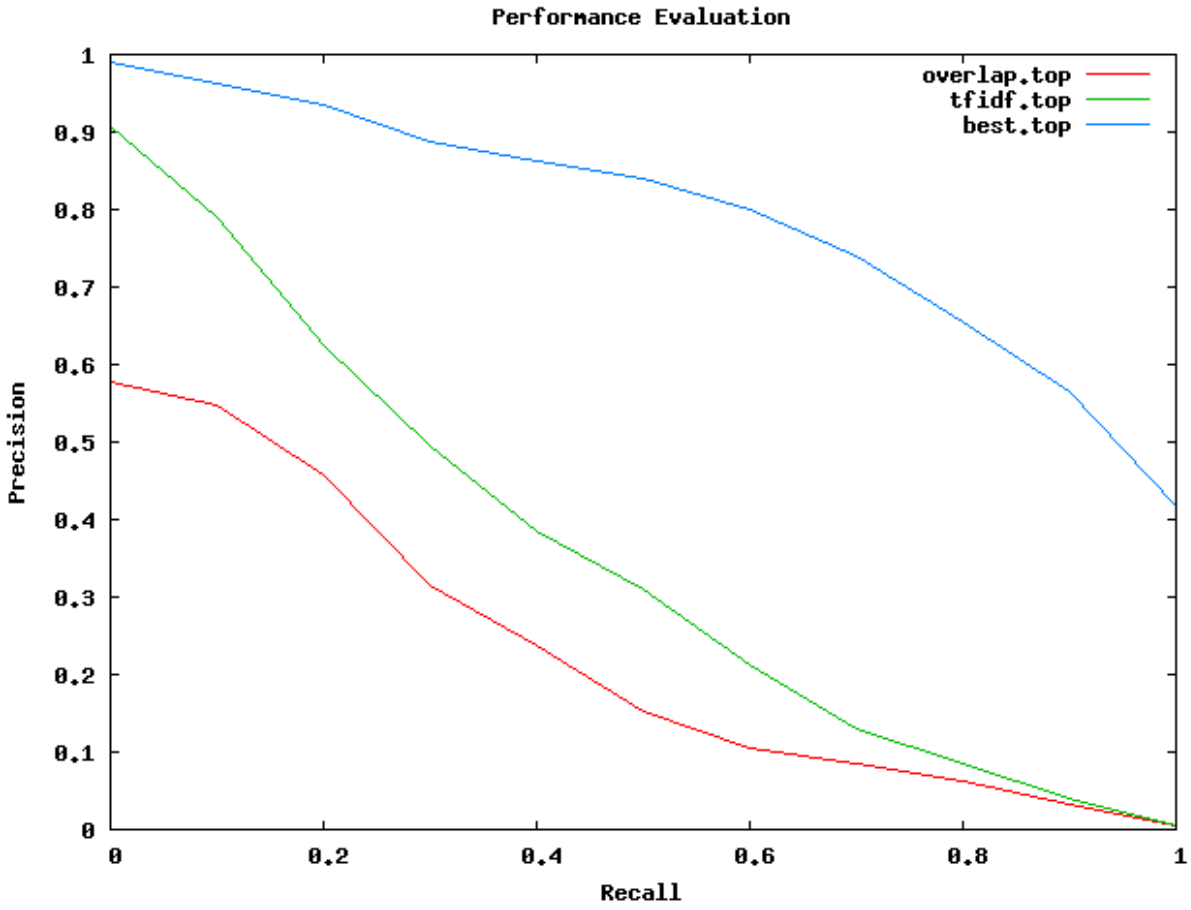

Figure 1: A comparison of the performance of the three algorithms.

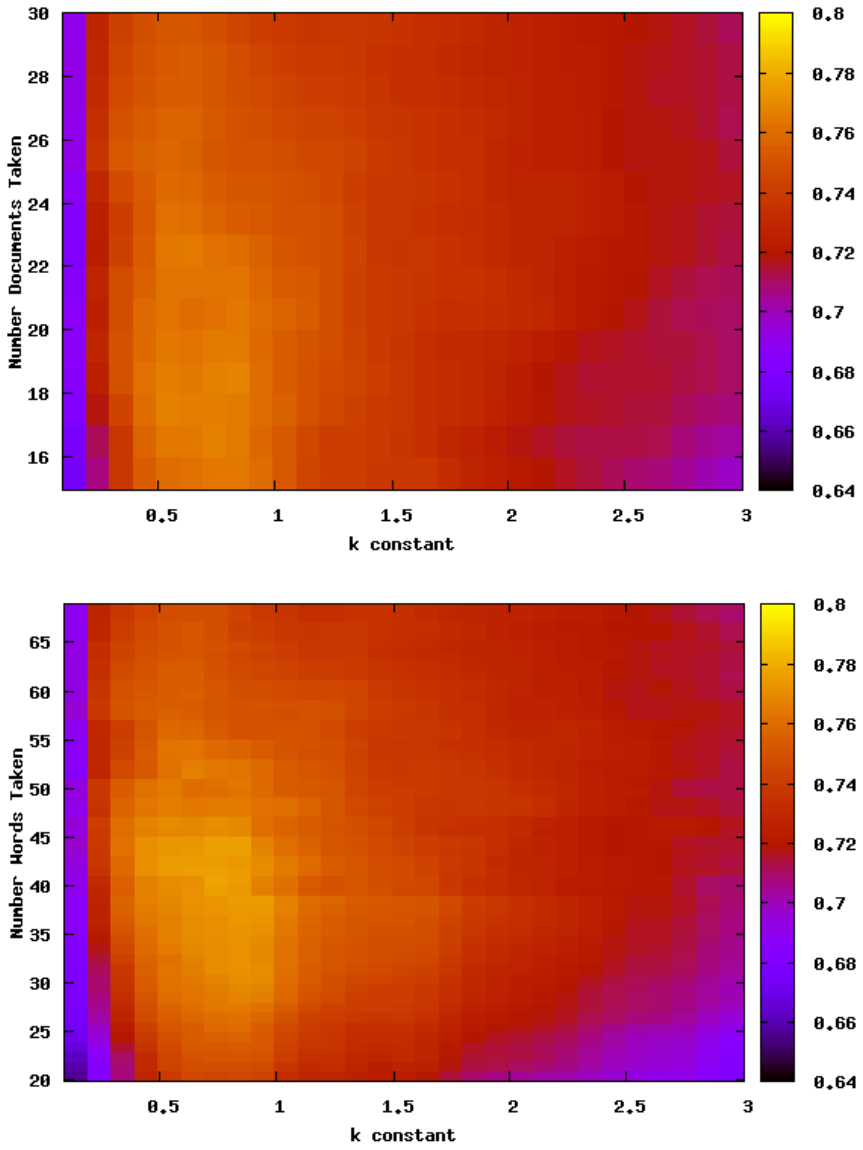

Figure 2: The results of searching the space $\{0.1 \leq k \leq 3.0,\ 15 \leq n_d \leq 30,\ 2n_d - 10 \leq n_w \leq 2n_d + 10\}$. Both graphs show the average precision as a colour from black (low) to yellow (high). The top graph shows a comparison of 'k' against the number of documents taken, whilst the bottom shows a comparison of the same 'k' against the number of words taken. A clear patch of high MAP value can be seen at around k = 0.75, $n_d = 19$, $n_w = 38$.