TTS **s0943941** **Extension Given: 30/10/12**

Report

Overlap - This design is straightforward. For every document, I check it against every individual query word. If the word occurs in the document, I add 1. This forms the basis for each algorithm implemented in my dataset, though it is a simplistic approach.

Tf-Idf – I first discover the required statistics about the document set and the query set. I loop through the document set first, and find the average document length, and the total amount of documents in the file. Then I create a hash map linking each the number of times each word in the query appears in the documents. This speeds up the lookup times during the iteration. Next, I loop over the documents and the queries, counting the amount of times the current query word appears, and calculate the rank using calcRank().

Best - Rather than improving my retrieval rate by changing the algorithm, I decided to focus on processing the malformed words in the documents (since the previous Lab assignment had been on spelling correction). The process goes like this:

For each word in the document, I check to see if it belongs to a corpus file (given in the labs). If it is not a valid word, I perform two types of analysis (involving three different hashmaps) – comparing the SoundEx code (again using code from the lab) and the first letters of the document word against the current query word we are analysing. If either criteria matches, a list of potential spelling corrections for the word is returned. (I initially checked the last three words as well, and while this increased the performance, it went over the given time limit by 5 minutes).

From this list of potential matches, I ensure that the Levenshtein distance is small (1). In this, I assume that the words are generally correct, and that the user will only make small mistakes (with greater chance for a larger word to be misspelled). From this smaller list, I check if the queryWord has been returned as a possible correction. If it has, I assume the user has misspelled the word. I remove the misspelled word from the document, and add the word onto the end (possible because this is a bag of words).

There are three downsides to this approach. One occurs in the case where a number of common words are corrected to rarer words, as this will drastically increase the weight of this document under tf-idf. The second is the runtime – I averaged around 27 minutes for the entire set, not practical for working with real data. Finally, the corpus I possess and the SoundEx algorithm are intended for English language words. Some of the words in the dataset were foreign (I recognized French and Spanish, and there could be more). These will be corrected to English words by my processing algorithm. I attempted to correct these problems, but ran out of time.

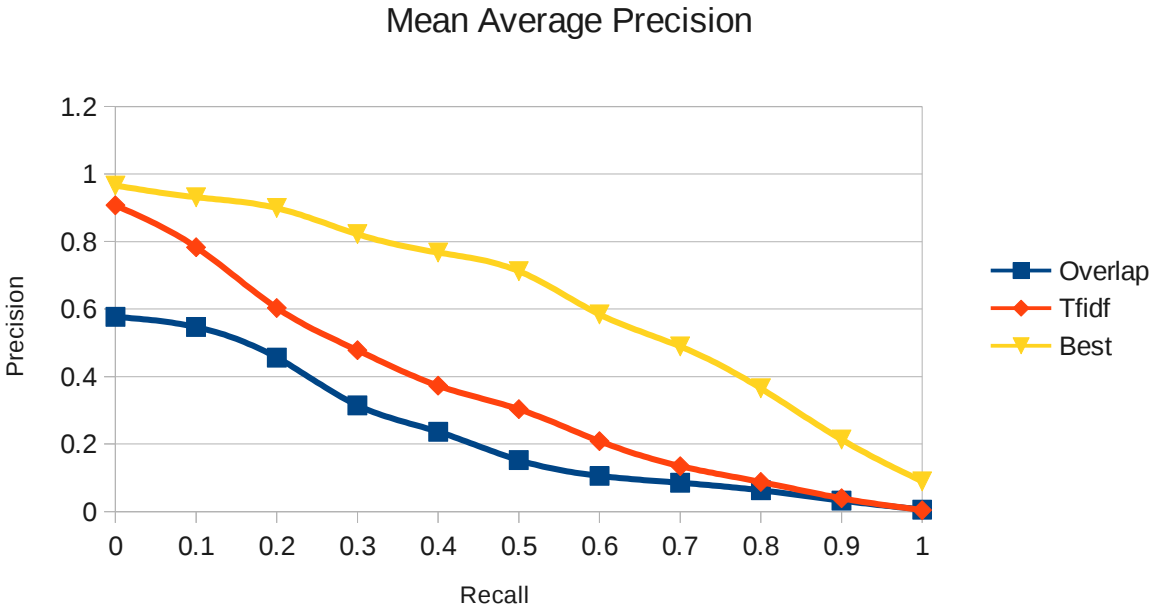
A final improvement that I tried was a removal of special characters. This was after realizing that the corpus given had been preprocessed, and a number of words did not exist within it. Printing out the list of offending words, I discovered a number of hyphenated words. I decided to treat these as independent words, replacing the hyphens with spaces. In the case of ‘white-tail’, both words can be considered valuable keywords. In the case of ‘quick-and-dirty’ (hypothetical example), the ‘and’ is trivial and will be made irrelevant by tf-idf.

The ranking itself is the same as tf-idf, only with a second loop through each document word at the step of processing each query. Note that the lookup table from tf-idf is also updated, to keep the measurements as accurate as possible.

Results: Overlap Average Precision: 0.1871
Tf-Idf Average Preciosn: 0.3334
Best Average Precision: 0.6168

With all three ranking algorithms, the precision is high for small amounts of documents. This is trivial – precision means nothing with a small of amount of data. The downward trend of each graph is indicative of larger document size – harder to be precise.

Comparing the two, tf-idf bests overlap at all steps (save for maybe the last). This is what we expect. Best performs better than both at all levels, so it is undoubtedly an improvement.


Mean Average Precision
Precision
1.2
1
0.8
0.6
0.4
0.2
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Recall
Overlap
Tfidf
Best