# Information retrieval Exercise 3

### Aharon Sharim, 052328524

## The basic algorithm

For the basic algorithm, we've decided to leverage Lucene's default configuration.

We used the StandardAnalyzer, which uses common stop words for the English language to tokenize both the documents and the query. We also used the ClassicSimilarity to calculate the relevance score between the query and the documents, which implements tf-idf similarity, with log-scaling and normalization.

We "played" the minimum score threshold to get what we felt a good balance between precision and recall, while giving up a bit on precision to gain some recall on queries which have small amount of "Truth" documents.

## The improved algorithm

For the improved algorithm we've tried to combine several mechanisms:

### Adding synonyms for words

We first focused on the queries which returned no results from the old algorithm. We've found a few of them contained words which weren't exactly in the document, but rather contained similar words, or names which were split into two:

Viet nam / Viet cong, for example, are two formal names, but a standard indexer would recognize each as two words with "viet" as a 50% similarity. We tried to merge these two names, for example, to "Vietnam" and "Vietcong".

Some queries had for example the word "Coup", while the truth document rather had the word "rebellion". We tried to add the word "rebellion", in that case, to the query.

Unfortunately, both those measures didn't prove to be very effective in changed the score for the better. Perhaps a larger corpus is required for it to be effective.

### Using a better Analyzer

We've switched the analyzer to Lucene's 'EnglishAnalyzer', which does stemming for English words on-top of the normal stop words elimination.

This proved to be a small but noticeable improvement to both the precision and recall in several cases.

### Trying different scoring methods

We tried several combinations of scoring methods by subclassing the ClassicSimilarity class. We mainly focused on the idea that in such a relatively small amount of documents, using log – scale was probably hurting more then it help.

What we've found is that just adding up the TF instead of using a log scale improved relevancy in many cases.

### Dynamic threshold

Finally, we was that in some queries we get no results about the threshold, and that simply reducing the threshold hurt the precision of most other quries.

Hence, we've decided to implement a dynamic – cutoff scheme, in which we take the threshold to be a percentage out of the highest score received.

This change allowed us to get results in the queries with natively low scores, while usually not hurting (or not hurting too much) the precision on other queries.

## Comparison

Following is a table that displays the comparison between both algorithms.

As can be seen, the improvements we've made to the algorithm are not a "Total" win to all cases, but there are many more cases which were improved (mostly cases which returned no results in the basic algorithm), with only a few losses on F1 score overall.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Basic** |  |  |  | **Improved** |  |
| **Precision** | **Recall** | **F1** |  | **Precision** | **Recall** | **F1** |
| 0.54545456 | 0.8571429 | 0.333333 |  | 0.75 | 0.85714287 | 0.4 |
| 0.18181819 | 1 | 0.153846 |  | 0.2 | 1 | 0.166667 |
| 0.07692308 | 0.25 | 0.058824 |  | 0.055556 | 0.25 | 0.045455 |
| 0.033333335 | 0.2 | 0.028571 |  | 0.083333 | 0.2 | 0.058824 |
| 0 | 0 | 0 |  | 0 | 0 | 0 |
| 0.5714286 | 0.8888889 | 0.347826 |  | 0.777778 | 0.7777778 | 0.388889 |
| 0 | 0 | 0 |  | 0 | 0 | 0 |
| 0.2857143 | 1 | 0.222222 |  | 1 | 1 | 0.5 |
| 1 | 0.375 | 0.272727 |  | 0.857143 | 0.75 | 0.4 |
| 0.75 | 1 | 0.428571 |  | 0.75 | 1 | 0.428571 |
| 0.18181819 | 1 | 0.153846 |  | 0.125 | 0.5 | 0.1 |
| 1 | 0.7142857 | 0.416667 |  | 1 | 0.71428573 | 0.416667 |
| 0.25 | 0.6666667 | 0.181818 |  | 0.5 | 1 | 0.333333 |
| 0 | 0 | 0 |  | 0 | 0 | 0 |
| 0.8 | 0.8 | 0.4 |  | 0.8 | 0.8 | 0.4 |
| 0.5 | 1 | 0.333333 |  | 0.285714 | 0.6666667 | 0.2 |
| 1 | 1 | 0.5 |  | 0.4 | 1 | 0.285714 |
| 0 | 0 | 0 |  | 0 | 0 | 0 |
| 0.4 | 0.4 | 0.2 |  | 0.5 | 0.4 | 0.222222 |
| 0 | 0 | 0 |  | 0 | 0 | 0 |
| 1 | 0.5 | 0.333333 |  | 0.5 | 0.5 | 0.25 |
| 0.25 | 1 | 0.2 |  | 0.166667 | 1 | 0.142857 |
| 0 | 0 | 0 |  | 0 | 0 | 0 |
| 1 | 1 | 0.5 |  | 0.5 | 1 | 0.333333 |
| 0 | 0 | 0 |  | 0 | 0 | 0 |
| 0 | 0 | 0 |  | 0 | 0 | 0 |
| 1 | 0.3333333 | 0.25 |  | 0.117647 | 0.6666667 | 0.1 |
| 0 | 0 | 0 |  | 0.181818 | 0.8 | 0.148148 |
| 1 | 0.5 | 0.333333 |  | 1 | 0.5 | 0.333333 |
| 0.5 | 0.4 | 0.222222 |  | 1 | 0.4 | 0.285714 |
| 0 | 0 | 0 |  | 0.071429 | 0.14285715 | 0.047619 |
| 0.09090909 | 1 | 0.083333 |  | 0.2 | 1 | 0.166667 |
| 1 | 0.5 | 0.333333 |  | 1 | 0.5 | 0.333333 |
| 1 | 1 | 0.5 |  | 1 | 1 | 0.5 |
| 0.33333334 | 1 | 0.25 |  | 0.333333 | 1 | 0.25 |
| 0 | 0 | 0 |  | 0 | 0 | 0 |
| 0 | 0 | 0 |  | 0 | 0 | 0 |
| 0 | 0 | 0 |  | 0 | 0 | 0 |
| 1 | 0.4444445 | 0.307692 |  | 1 | 0.44444445 | 0.307692 |
| 0.7777778 | 0.7777778 | 0.388889 |  | 0.875 | 0.7777778 | 0.411765 |
| 0.16666667 | 0.3333333 | 0.111111 |  | 0.5 | 0.33333334 | 0.2 |
| 0 | 0 | 0 |  | 0 | 0 | 0 |
| 1 | 0.5 | 0.333333 |  | 0.666667 | 1 | 0.4 |
| 0 | 0 | 0 |  | 0.222222 | 1 | 0.181818 |
| 0.5 | 0.2 | 0.142857 |  | 0.129032 | 0.8 | 0.111111 |
| 0.75 | 0.5 | 0.3 |  | 0.6 | 0.16666667 | 0.130435 |
| 0 | 0 | 0 |  | 0.25 | 0.33333334 | 0.142857 |
| 0 | 0 | 0 |  | 0.111111 | 1 | 0.1 |
| 0.6666667 | 0.75 | 0.352941 |  | 0.666667 | 0.5 | 0.285714 |
| 1 | 1 | 0.5 |  | 1 | 1 | 0.5 |
| 0 | 0 | 0 |  | 0.153846 | 0.6666667 | 0.125 |
| 0 | 0 | 0 |  | 0 | 0 | 0 |
| 1 | 1 | 0.5 |  | 0.666667 | 1 | 0.4 |
| 0.11111111 | 0.5 | 0.090909 |  | 0.2 | 0.5 | 0.142857 |
| 0.75 | 0.5 | 0.3 |  | 0.636364 | 0.5833333 | 0.304348 |
| 0 | 0 | 0 |  | 0 | 0 | 0 |
| 0 | 0 | 0 |  | 0.5 | 0.5 | 0.25 |
| 0.5 | 0.375 | 0.214286 |  | 0.714286 | 0.625 | 0.333333 |
| 0.2 | 0.5 | 0.142857 |  | 0.25 | 0.5 | 0.166667 |