**Description of Implementation**

*I have given a line-by-line breakdown of the code with through comments in the my\_retriever.py file. This section aims to tie those comments back to the mathematical concepts.*

For each of the implementations, a cosine similarity measure is used to measure the similarity between a document and the query. This measure is defined and subsequently refined to:

is constant as each document is compared against the same query, therefore we need not compute it. The similarity measure can now be computed in two parts. () can be computed ahead of query processing, while () must be computed for each query provided. Both components mentioned above are stored as lists, “self.doc\_l” and “qd\_sum” respectively, as experimentation showed that this was processed faster than a dictionary. “self.doc\_l[0]” gives the length of document one, index one gives the length of document two and so on - the same logic applies to qd\_sum.

Each implementation also needs the size of the collection. This is found by taking the set of document numbers on the second level of the dictionary, then taking the length of that set.

For TF-IDF we also need to compute document frequency of a word . This is stored in a dictionary mapping the word to . It is computed using a dictionary comprehension which iterates across terms in the index and records the number of documents on the second level of the dictionary for the given word. We also need to compute the IDF for each word. This is computed using a dictionary comprehension which iterates across the dictionary, dividing each of these values by and taking logarithms. We finally compute TF-IDF with a nested dictionary comprehension cycling through each document for a given word and multiply the corresponding by the IDF for that word.

**Binary Weighting:** To compute the program loops through each word in the Inverted File Index ( dictionary) - then a nested loop iterates over the second level of the dictionary, for each document containing the current word. For each document containing the given word the document’s element in self.doc\_l is incremented. is computed by taking the intersection of terms in the query and the index and incrementing qd\_sum for the given document.

**Term-Frequency Weighting:** To compute the same logic is used as above, but instead of incrementing the relevant element in the document length array, we square our iterable is currently pointing to and add this value to the document’s element in the self.doc\_l array. is calculated as above, except we take the product of in the document and .

**TD-IDF Weighting:** To compute the same logic is applied as above, but this time our nested structure iterates over the TF-IDF dictionary instead of the dictionary. For each third-level dictionary value the iterator points to, we square this TF-IDF weight and add it to the current document’s element in self.doc\_l. After the loop terminates we have for the chosen weighting scheme, so a list comprehension is used to root each sum of squares, giving the Euclidian length of each document. is calculated as above except out term weights instead come from the TF-IDF dictionary. We must also compute the TF-IDF for intersecting terms in the query (non-intersecting terms have value zero as IDF for what word is zero).

**Results**

In each of the nine experiments which used pre-processing techniques, precision, recall and accuracy were improved.

**Impact of Stop-Listing (SL)**

Stop-listing was most effective when used alongside term-frequency (TF) weighting. This is because TF rewards words the more they appear in the corpus, therefore high frequency, unimportant words like “a” or “the” are given a large weighting. Applying SL filters out this kind of word, so we just consider words which are high frequency in the text but not necessarily across the language (corpus dependent) as “important” instead. For this reason, stop-listing also improved performance of when used alongside the binary weighting scheme. SL marginally improved performance when used with TF-IDF. This is because TFIDF gives a lower weight to words which are high frequency across the corpus- this is somewhat functionally equivalent to the stop-list, so the effect of the SL is dampened.

**Impact of Stemming (ST)**

When used alongside a Binary or TF weighting scheme, the impact of stemming is minimal when compared to that of stop-listing. When used alongside a TFIDF weighting scheme, stemming results in significant improvements to both precision and accuracy over base TFIDF.

**Combining Stemming and Stop-listing**

For all three weighting schemes a combination of stemming and stop-listing produces the best F-measure, giving the best trade-off between precision and recall. The average increase in precision when pre-processing is applied is 0.093, while the average increase to recall is 0.073. This suggests these pre-processing is more effective at increasing precision than recall although only slightly.

**Conclusion**

Overall, a TD-IDF configuration with SL and ST pre-processing gives the best F-measure (0.23), meaning the best trade off between precision and recall. In all pre-processing configurations, TF-IDF performs better than its TF and Binary counterparts. Without pre-processing, the difference in performance between binary and TF weighting is marginal, with TF offering higher precision but identical recall. TF-IDF is the best performing weighting scheme and also benefits greatly from stemming over the two more primitive weighting schemes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Binary** | | | |
| No PP | SL | ST | SL and ST |
| **Precision** | 0.07 | 0.13 | 0.09 | 0.16 |
| **Recall** | 0.06 | 0.11 | 0.07 | 0.13 |
| **F-measure** | 0.06 | 0.12 | 0.08 | 0.14 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **TF-IDF** | | | |
| No PP | SL | ST | SL and ST |
| Precision | 0.17 | 0.19 | 0.24 | 0.25 |
| Recall | 0.14 | 0.15 | 0.19 | 0.2 |
| **F-measure** | 0.15 | 0.17 | 0.21 | 0.23 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Term Frequency** | | | |
| No PP | SL | ST | SL and ST |
| **Precision** | 0.08 | 0.16 | 0.11 | 0.19 |
| **Recall** | 0.06 | 0.13 | 0.09 | 0.15 |
| **F-measure** | 0.07 | 0.15 | 0.10 | 0.17 |