COM6115: Document Retrieval Assignment Report

1 Description of the implementation

1.1 introduction

All three weighting schemes are implemented. The weighting score is ranked from high to low to show 10 most relevant documents to the given query. Some calculations are completed before the iterations to speed up the code. A `time.time()` function is used temporarily in the `IR-engine.py` only for the use of recording time, it is completely independent with the implementation and could be removed in practice without affecting any results.

1.2 binary

The score is computed using the function $\frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} d_i^2}}$. In the Binary mode, $q_i = 0$ if the term is not in the query, and $q_i = 1$ for the term in the query regardless of the frequency and similar for d_i. The term $q_i d_i$ then represents the number of common terms in the query and the document and $\sum_{i=1}^{n} d_i^2$ is correlated to the length of documents.

1.3 TF mode

The score is computed using the function $\frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} d_i^2}}$. In the TF mode, q_i is the frequency of the term that occurs in the query and d_i is the frequency of the same term in the document. The score is higher if the term is more "important"(i.e. have higher frequency) in both query and document.

1.4 TFIDF mode

The score is computed using the function $\frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} d_i^2}} \cdot idf_i$, where $idf_i = \log_{10} \frac{|D|}{d_i}$. In the code, the idf_i is calculated in advance and multiplied to the result of TF mode directly. The difference between TF mode and the TFIDF mode is that the less common words weights more in the TFIDF mode.

2 Performance results

No stoplist, No stemming			
	Binary	TF	TFIDF
Rel_Retr	44	49	67
Precision	0.07	0.08	0.10
Recall	0.06	0.06	0.08
F-measure	0.06	0.07	0.09
Time (sec.)	0.20	0.25	0.305

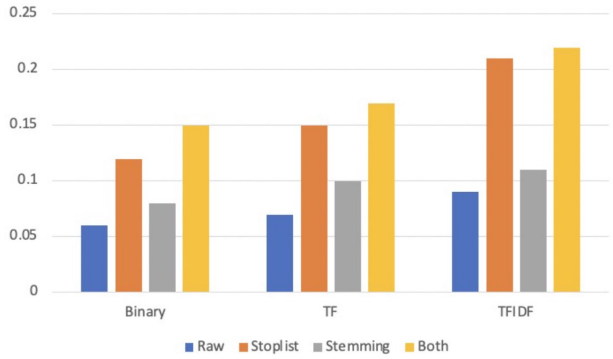
With stoplist, No stemming			
	Binary	TF	TFIDF
Rel_Retr	83	107	148
Precision	0.13	0.17	0.23
Recall	0.10	0.13	0.19
F-measure	0.12	0.15	0.21
Time (sec.)	0.09	0.11	0.12

No stoplist, With stemming			
	Binary	TF	TFIDF
Rel_Retr	61	73	82
Precision	0.10	0.11	0.13
Recall	0.08	0.09	0.10
F-measure	0.08	0.10	0.11
Time (sec.)	0.21	0.27	0.33

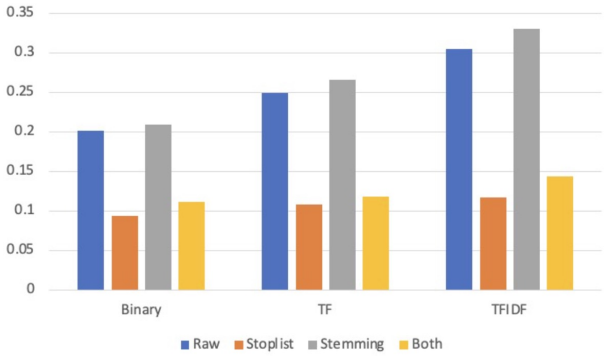
With stoplist, With stemming			
	Binary	TF	TFIDF
Rel_Retr	105	122	160
Precision	0.16	0.19	0.25
Recall	0.13	0.15	0.20
F-measure	0.15	0.17	0.22
Time (sec.)	0.11	0.12	0.14

Tabel 1: Data for applying different configurations and different retrieval methods

3 Discussion



Graph 1: F-measure for applying different configurations



Graph 2: Time for applying different configurations

Graph 1 is created to reflect the change of F-measure(i.e. the accuracy of the programme) on different retrieval methods. From the graph we notice that the application of stoplist has more distinctive improvement for all three retrieval methods comparing with stemming. The reason could be that there is a large number of "non-content" words in the queries and documents, which could also explain why stoplist has the most distinctive improvement on TFIDF, as the TFIDF method focuses on less common terms, and the recognisation of less common terms could be affected by "frequent but non-content" terms.

The effect of applying stemming is less obvious on all three methods. The reason could be that terms in queries and documents do not have too many morphological variants. The method is still effective as it reduces the affect of inflection, although it leads to a less obvious improvement.

Another figure that is worth to discuss is the time(**Graph 2**). TFIDF method takes the longest time to compute because of extra calculations for the idf_i value. Another observation is that applying the stoplist reduces the time taken to run the code while applying stemming increases that time. The stoplist method reduces the number of common terms in queries and documents by removing a list of "non-content" words, and the stemming adds common terms in queries and documents by changing words to their "stemmer". A larger number of common terms requires more iterations and longer time to process and vice versa.

4 Conclusion

The TFIDF retrieval method and stoplist manipulation method is suggested for this case. Both of TFIDF and stoplist have a good improvement on accuracy with reasonable time cost. Stemming manipulation method could also be considered as it improves the accuracy, only when time efficiency is not a main factor.