Document Retrieval Report

Student Name: Jian Cui
UCard No: 001718979

Introduction

The purpose of this project is to perform document retrieval based on the vector space model. We have two important basic files available for the implementation of this system. The provided query.txt file contains a set of IR queries that can be used to retrieve relevant documents, and the documents.txt file is a document collection of all documents. These two files are pre-processed(whether a stoplist is used, whether terms are stemmed) based on the inverted index to effectively identify documents related to the query.

The implementation of this document retrieval system under three different term weighting schemes: binary, tf and tfidf. After the system is implemented, the performance of the system is tested and evaluated under different preprocessing options and available term weighting schemes. Specifically, how does different preprocessing affect the accuracy of the retrieval, and which term weighting scheme performs better, etc.

Implementation of Method

The implementation of the system is mainly divided into two steps. The first step is to view documents as vectors of term weighting. The second step is to rank the candidate documents according to the calculation of similarity between a document and a query.

For the calculation of term weighting, we describe this three schemes separately.

1) **Binary**
   If the term present in the document collection, the weight is 1, otherwise it is 0
2) **TF**
   The number of times a term appears in a specific document.
3) **TFIDF(TF multiply IDF)**

$$idf_{w,D} = \log\frac{|D|}{df_w}$$

   For this formula, $|D|$ is total number of documents in collection, df_w is number of documents containing term w.

With respect to the second step, the similarity calculation formula as follow:

$$sim(\vec{q},\vec{d}) = \cos(\vec{q},\vec{d}) = \frac{\sum_{i=1}^{n} q_i\, d_i}{\sqrt{\sum_{i=1}^{n} {q_i}^2}\,\sqrt{\sum_{i=1}^{n} {d_i}^2}}$$

When we have calculated the similarity of each candidate document with query, we rank this documents according to the magnitude of the similarity value, that is, the larger the similarity value, the higher the relevance of the document to the query.

Result

For this project, I have done all the necessary work. The test results of the system under different configurations are as follows.

		No stoplist No stemming	No stoplist With stemming	With stoplist No stemming	With stoplist With stemming
binary	Precision	**0.07**	**0.09**	**0.13**	**0.16**
	Recall	0.06	0.07	0.10	0.13
	F-measure	0.06	0.08	0.11	0.15
	Time(second)	1.26	0.91	0.67	0.56
tf	Precision	**0.08**	**0.11**	**0.17**	**0.19**
	Recall	0.06	0.09	0.13	0.15
	F-measure	0.07	0.10	0.15	0.17
	Time(second)	1.39	1.27	0.61	0.64
tfidf	Precision	**0.21**	**0.26**	**0.22**	**0.27**
	Recall	0.17	0.21	0.18	0.22
	F-measure	0.18	0.23	0.19	0.24
	Time(second)	3.54	3.01	2.36	1.82

Conclusion

From the above table, we can see that for document retrieval, the scheme of TFIDF has the highest accuracy rate, up to 27%(withstoplist_withstemming), while the precision of schemes of TF and Binary are 19% and 16%, respectively. The same for Recall, the performance of TFIDF>TF>Binary.

F-measure is also a good evaluation standard, it combines the results of Precision and Recall, also its performance for Binary, TF and TFIDF has same trace with Precision.

From the scheme of Binary, we can see that the precision without using stoplist and stemming is the lowest, which is 7%, and the precision of using stemming and stoplist alone has improved, which is 9% and 13%, respectively. So we could infer that using stoplist to improve the system performance is better than stemming under the Binary scheme. With both using stemming and stoplist, it is clear that the system performs best at 16%. The TF and TFIDF schemes are similar, but the only difference is that for TFIDF scheme, using stoplist to improve the system performance is less than using stemming(22% and 26%).