Report for Assignment2

1. Introduction

This report will show the results based on the *Retrieve* class which aims to built an IR systems. In the *Retrieve* class, three term weighting schemes are applied successful. The cosine of two vectors will be used to enlarge their similarity which will be contained in *give result* function.

2. Methods

In this part,'-s' means stoplist will be used, '-p' means stemming will be used. All results will be showed in the table below. The highest F-measures can achieve 0.28 when apply TDIDF weight, stoplist and stemming. Before giving out the results, the number of same words between query and document has been calculated and multiplied to the cosine of two vectors.

3. Performance

3.1 Binary Results

Binary weights: whether or not term is present in document.

Weighting Schemes	None	stoplist	stemming	stoplist & stemming
Rel-Retr	75	108	95	132
Precison	0.12	0.17	0.15	0.21
Recall	0.09	0.14	0.12	0.17
F-measure	0.10	0.15	0.13	0.18
Time(s)	1.166	0.355	1.269	0.503

3.2 TF Results

TF weights: number of times the word occurs in document. The calculation process will be contained in *compute TF* function.

Weighting Schemes	None	stoplist	stemming	stoplist & stemming
Rel-Retr	76	129	102	149
Precison	0.12	0.20	0.16	0.23
Recall	0.10	0.16	0.13	0.19
F-measure	0.11	0.18	0.14	0.21
Time(s)	1.730	0.487	1.767	0.724

3.3 TFIDF Results

IDF weights: inverse document frequency.

TFIDF = TF * IDF (contained in *compute_tfidf function*)

Weighting Schemes	None	stoplist	stemming	Stoplist & stemming

Rel-Retr	140	150	168	182
Precison	0.22	0.23	0.26	0.28
Recall	0.18	0.19	0.21	0.23
F-measure	0.19	0.21	0.23	0.25
Time(s)	1.729	0.602	1.880	0.796

4. Discussion

When applying both stoplist and stemming, the system can achieve the fastest speed no matter which term weight scheme be chosen.

TFIDF can achieve better peroformance but it will also cost more time to give the results.

When the dataset is not big enough, try to calculate the number of same words between query and document then apply it to the function eval() to enlarge their similarity can improve the performance of each model.