The report of Text Processing of Assignment 1:

1. Description of the implementation achieved:

My code implement retrieval under alternative term weighting schemes, i.e. under binary, term frequency and TFIDF schemes. For different schemes, it can successfully return 10 best-ranking documents for each query and print result to a result file. Beyond that, my code is very efficient and can return results very quickly. You can see that from next pictures.

The scheme of binary:

```
[dyn116192:Document_Retrieval_Assignment_Files 3 vvvvv$ python ir_engine.py -s -p]
 -w binary -o binary_sp.txt
TIME (retrieval): 0.10
[dyn116192:Document_Retrieval_Assignment_Files 3 vvvvv$ python ir_engine.py -s -w]
 binary -o binary_s.txt
TIME (retrieval): 0.05
[dyn116192:Document_Retrieval_Assignment_Files 3 vvvvv$ python ir_engine.py -p -w]
 binary -o binary_p.txt
TIME (retrieval): 0.27
[dyn116192:Document_Retrieval_Assignment_Files 3 vvvvv$ python ir_engine.py -w bi]
nary -o binary.txt
TIME (retrieval): 0.24
```

The scheme of term frequency:

```
[dyn116192:Document_Retrieval_Assignment_Files 3 vvvvv$ python ir_engine.py -s -p
 -w tf -o tf_sp.txt
TIME (retrieval): 0.93
[dyn116192:Document_Retrieval_Assignment_Files 3 vvvvv$ python ir_engine.py -s -w
 tf -o tf_s.txt
TIME (retrieval): 0.58
[dyn116192:Document_Retrieval_Assignment_Files 3 vvvvv$ python ir_engine.py -p -w
 tf -o tf_p.txt
TIME (retrieval): 2.58
[dyn116192:Document_Retrieval_Assignment_Files 3 vvvvv$ python ir_engine.py -w tf
 -o tf.txt
TIME (retrieval): 2.58
```

The scheme of TFIDF:

```
[dyn116192:Document_Retrieval_Assignment_Files 3 vvvvv$ python ir_engine.py -s -p]
 -w tfidf -o tfidf_sp.txt
TIME (retrieval): 0.71
[dyn116192:Document_Retrieval_Assignment_Files 3 vvvvv$ python ir_engine.py -s -w]
 tfidf -o tfidf_s.txt
TIME (retrieval): 0.44
[dyn116192:Document_Retrieval_Assignment_Files 3 vvvvv$ python ir_engine.py -p -w]
 tfidf -o tfidf_p.txt
TIME (retrieval): 2.02
[dyn116192:Document_Retrieval_Assignment_Files 3 vvvvv$ python ir_engine.py -w tf]
idf -o tfidf.txt
TIME (retrieval): 1.98
```

2. Evaluation of result:

Evaluation of result

	Retrieved Documents	Relevant Documents	Relevant Retrieved	Precision	Recall	F-measure
Binary(index and queries with stop list with stemming)	640	796	103	0.16	0.13	0.14
Binary(index and queries with stop list but not with stemming)	640	796	84	0.13	0.11	0.12
Binary(index and queries not with stop list but with stemming)	640	796	59	0.09	0.07	0.08
Binary(index and queries not with stop list not with stemming)	640	796	44	0.07	0.06	0.06
TF(index and queries with stop list with stemming)	640	796	123	0.19	0.15	0.17
TF(index and queries with stop list but not with stemming)	640	796	106	0.17	0.13	0.15
TF(index and queries not with stop list but with stemming)	640	796	72	0.11	0.09	0.10
TF(index and queries not with stop list not with stemming)	640	796	49	0.08	0.06	0.07
TFIDF(index and queries with stop list with stemming)	640	796	172	0.27	0.22	0.24
TFIDF(index and queries with stop list but not with stemming)	640	796	140	0.22	0.18	0.19
TFIDF(index and queries not with stop list but with stemming)	640	796	166	0.26	0.21	0.23
TFIDF(index and queries not with stop list not with stemming)	640	796	132	0.21	0.17	0.18

3. Conclusion:

I can draw a conclusion from above table that the best scheme is TFIDF, no matter whether the index and queries with stop list and stemming. And the TF is a little better than binary. For each scheme, the best result is always when index and queries with stop list and stemming. The second best for TF and binary is when index and queries with stop list but not with stemming, but for TFIDF is when index and queries not with stop list but with stemming. And the worst result is always when index and queries not with stop list and stemming.

So I can infer that a good IR system should not only consider the item's frequency and whether the item appear, but also consider the value of TFIDF of every item. Besides that, the stop list and stemming is important for a good IR system to improve the accuracy.