1.Performacne Testing

Termweight	Stoplist	Stemming	Runtime	Precision	Recall	F-measure
Binary	No	No	1.25	0.07	0.06	0.05
Binary	Yes	No	0.80	0.13	0.11	0.12
Binary	No	Yes	1.25	0.09	0.07	0.08
Binary	Yes	Yes	0.79	0.16	0.13	0.14
Tf	No	No	1.30	0.08	0.06	0.07
Tf	Yes	No	0.89	0.17	0.13	0.15
Tf	No	Yes	1.33	0.11	0.09	0.10
Tf	Yes	Yes	0.88	0.19	0.15	0.17
Tf.idf	No	No	1.45	0.21	0.17	0.19
Tf.idf	Yes	No	0.86	0.22	0.17	0.19
Tf.idf	No	Yes	1.56	0.26	0.21	0.23
Tf.idf	Yes	Yes	0.86	0.27	0.21	0.24

2. Observation

a. The stemming and stoplist method will improve the runtime performace. See more effectives on stoplist.
b. The precision : td.idf > tf > binary .
c. The python dict is a hashmap, its worst case is therefore O(n) if the hash function is bad and results in a lot of collisions. The average time complexity is of course O(1). So that a dictionary is a better way to store data.

3. Process

a. While init the data, we have the index. Due to the fact that all the |D|, df, tf are all the same in data, we can do the counting in the init. And count the d which is the ranking value. Then create a candidate data which contain all the doc to count. After create the candidate document sequence, we can create a candidate_term dictionary which contain all the term and frequency for each document(Make it a two level will enable us to do it easier). Last, depends of each method(binary, tf, tf.idf),we do the different process to data. After having the rank, we can sort our result before we output it.