Question 1 – Running queries

map: 0.16925271708009806

Rprec: 0.19803030303030303

recip_rank: 0.37175925925925923

P_5: 0.19999999999999998

P_10: 0.14666666666666667

P_15: 0.09777777777777778

Question 2 – TF-IDF Similarity

map: 0.28087837129801413

Rprec: 0.29668109668109666

recip_rank: 0.5202777777777777

P_5: 0.30000000000000004

P_10: 0.21666666666666665

P_15: 0.14444444444444446

Question 3 – Evaluation measures

From the evaluation results of TF and TFIDF, we can see that the evaluation measures of the TFIDF system are higher than the TF system. Therefore, I think TF_IDF performs better in terms of similarity. For the five evaluation measures, I focus more on MAP and P_10, because the precision only considers the number of relevant documents in the returned results, and does not consider the rank between documents. For gov corpus and search engines, the returned results must be in order, and the more relevant documents should are ranked higher. MAP is a single-value indicator reflecting the performance of the system on all relevant documents. A higher MAP value means a higher ranking of relevant documents retrieved. Secondly, since most users are unwilling to turn the page to see the back documents, therefore need to pay more attention to the precision of the first few documents. The value of p_10 indicates the precision of the top 10 documents.

Question 4 – Improving the pre-processor

Setting 1：change PorterStemmer() to LancasterStemmer()

Result

map: 0.26236514378478665

Rprec: 0.30001443001443007

recip_rank: 0.48055555555555546

P_5: 0.30000000000000004

P_10: 0.21

P_15: 0.14000000000000004

Analysis

By comparing with the evaluation results of TFIDF, we can see that in addition to a slight increase in Rprec, other aspects are reduced. Probably because compared to PorterStemmer(), LancasterStemmer() lead to many shorter words will become totally obfuscated, although the speed is faster, the precision is reduced.

Setting 2：change PorterStemmer() to SnowballStemmer()

Result

map: 0.28767069934034223

Rprec: 0.3176334776334776

recip_rank: 0.5527777777777777

P_5: 0.3266666666666666

P_10: 0.21999999999999995

P_15: 0.14666666666666667

Analysis

By comparing with the evaluation results of TFIDF, we can see that every aspect has a slight improvement, snowballstemmer() is an improved algorithm of PorterStemmer(), faster and more precise than PorterStemmer() , so I choose snowballstemmer() as stemmer.

Setting 3: change nltk.tokenize.WhitespaceTokenizer() to nltk.regexp_tokenize()

Result

map: 0.2947449506032455

Rprec: 0.32113531629660663

recip_rank: 0.5592293906810035

P_5: 0.3096774193548387

P_10: 0.22903225806451613

P_15: 0.15268817204301077

Analysis

By comparison, we can see that in addition to a small reduction in p_5, all other aspects have improved. This is because nltk.regexp_tokenize() uses regular expressions to separate words, which is more accurate.

Setting 4: change nltk.tokenize.WhitespaceTokenizer() to nltk.word_tokenize()

Result

map: 0.3073246295151057

Rprec: 0.326497113997114

recip_rank: 0.5953174603174602

P_5: 0.31333333333333335

P_10: 0.2366666666666667

P_15: 0.15777777777777785

Analysis

Compared with nltk.regexp_tokenize(), there are good improvements in all aspects, so I choose to use nltk.word_tokenize() as the tokenizer.

Setting 4: Add stop word list

Result

map: 0.3043939737511166

Rprec: 0.33483044733044737

recip_rank: 0.5959259259259259

P_5: 0.31333333333333335

P_10: 0.2366666666666667

P_15: 0.15777777777777785

Analysis

Compared with the previous results, adding stop word did not significantly improve. It may be because there are not many stop words in the document, so the effect is not obvious. In addition, adding stop word will lead to longer running time, so I give up stop word.

Question 5 – Further Modification

In order to improve the performance of the system, I used the BM25 algorithm to calculate the relevance between each document and the query.

Result

map: 0.4340686198721913

Rprec: 0.4332359307359307

recip_rank: 0.5964285714285714

P_5: 0.48000000000000004

P_10: 0.31666666666666665

P_15: 0.21111111111111114

Analysis

Compared to TFIDF, the evaluation results of BM25 have improved a lot. It is because BM25 adds several adjustable parameters on the basis of traditional TF-IDF, making it more flexible and powerful in application, and has higher practicability.