WSM Proj2

111356032 Chu Yun Wang MIS

December 6, 2022

1 Abstract

This report uses Indri-5.18 to retrieve relevant documents from WT2G collection using 50 Trec queries and evaluate the returned ranked result. The implemented ranking functions are as follows: Vector Space Model raw TF, OKAPI, Language Model with different smoothing methods, and Relevance Model.

The following experiment will be divided into four parts: Firstly, we will discuss the effect of different indexing method, various query types, parameter setting for ranking functions. Besides, we will compare the three ranking functions - OKAPI, language model with laplace smoothing(LM) and language model with Jelinek-Mercer smoothing(JM). Moreover, since LM doesn't do well compared to another two ranking functions, we improve LM's performance by adding pseudo relevance feedback. Lastly, after comparing the performance of OKAPI TF and raw TF, we find out that there is no significant difference between them.

2 Environmental Setup and indexing method

This section will briefly introduces the environmental settings and discusses the effect of different indexing method: with or without stopwords filtering and stemming.

2.1 Environmental Setup

This experiment uses indri-5.18 under ubuntu @20.04 and gcc @4.8. First and foremost, we need to configure the file to produce make file for the indri source code. Afterwards, we run make and make install to make indri executable.

2.2 Indexing Methods Comparison

2.2.1 The Effect of Stopwords Filtering

To see whether stopwords filtering can affect retrieval performance significantly, we run 50 TREC queries' title as query against WT2G collection with Okapi, Language Model with Dirichlet smoothing(LM) and Langauge Model with Jelinek-Mercer smoothing function(JM) respectively, using both stopwords filtering indexing and stopwords indexing.

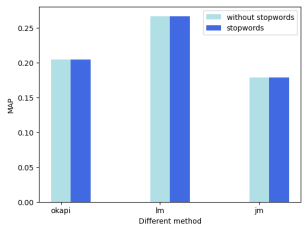
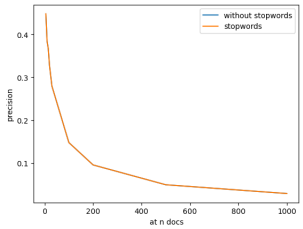
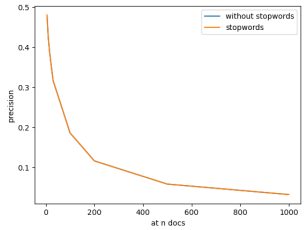


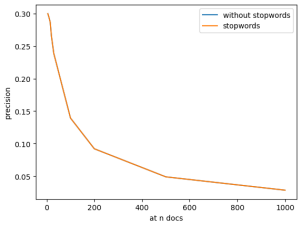
Figure 1: MAP of OKAPI, LM, JM



(a) OKAPI



(b) LM



(c) JM

Figure 2: Precision at n documents

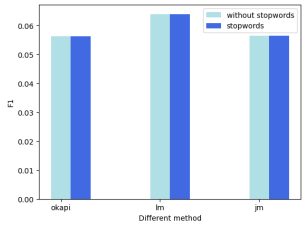


Figure 3: F1 of OKAPI, LM, JM

In both of Figure 1, 2, and 3, the results of without and with stopwords filtering have no significant difference in MAP, precision curve, and F1 value.

In my opinion, there are two reason for the above stated phenomena: lack of stopword in query (title tag) and the effect of IDF. After inspecting the whole query file, we found that almost all the queries do not contain stopwords. Therefore, stopwords filtering is not helpful in this case. In addition, IDF makes the frequent words in a document decrease its weight, which makes the stopwords filtering much more ineffective.

2.2.2 The Effect of Stemming

In this section, experiment was done to see whether stemming has its effect on the performance of document retrieval.

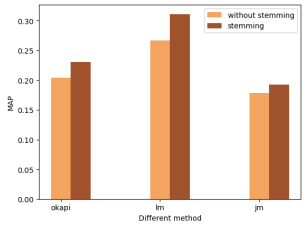
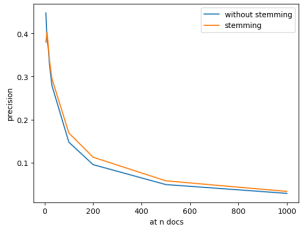
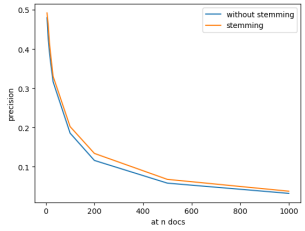


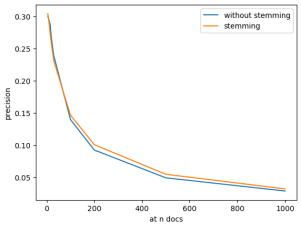
Figure 4: MAP of OKAPI, LM, JM



(a) OKAPI



(b) LM



(c) JM

Figure 5: Precision at n documents

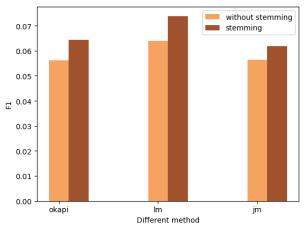


Figure 6: F1 of OKAPI, LM, JM

In figure 4, 5, and 6, we can see that brown sticks are significantly higher than the yellow sticks, and the yellow curves are equal or higher than the blue curve at any level. These results signify that stemming can effectively increase the precision, precision at k, and F1.

The reason is that stemming can reduce the disturbing effect of tense e.g. play and played, and plural forms e.g. bottle and bottles, and effectively find the matched words which originally may be in different forms without stemming. Thus, stemming can effectively increase the performance of document retrieval.

3 Query Types Comparison

After determining the indexing methods, we will discuss the retrieval results under different query types. This experiment uses TREC Queries, which have three kinds of query tags - title, description, and narrative, and we will observe the differences of queries from title tag, and queries from description tag.

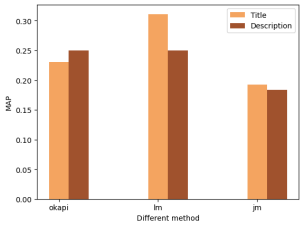


Figure 7: MAP comparison

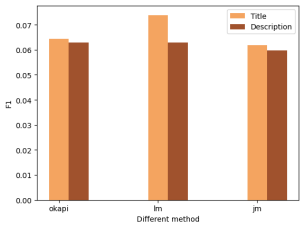
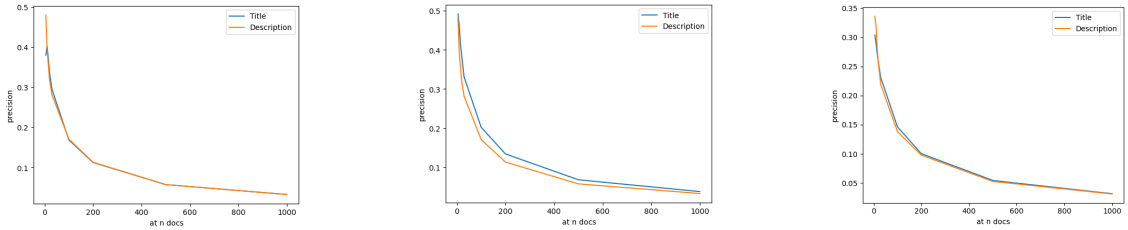


Figure 8: F1 comparison



(a) OKAPI (b) LM (c) JM

Figure 9: Query type comparison under different ranking functions

Surprisingly, The results from figure 7, 8, 9 show that queries from title tag outperforms the queries from description tag almost in the three ranking functions except for OKAPI's MAP. In our intuition, we might think that queries with more detailed explanation may help us find documents that are much more relevant. However, the results turn out to be the opposite.

The reason is that, after observing the different of the two types of queries, we find that queries from description tag are written in natural language, whereas queries from title are extracted to only few core words of the search intention. Under this circumstance, queries from description tag contains much more stopwords or general words, which may decrease the performance of retrieval, contributing to the above results.

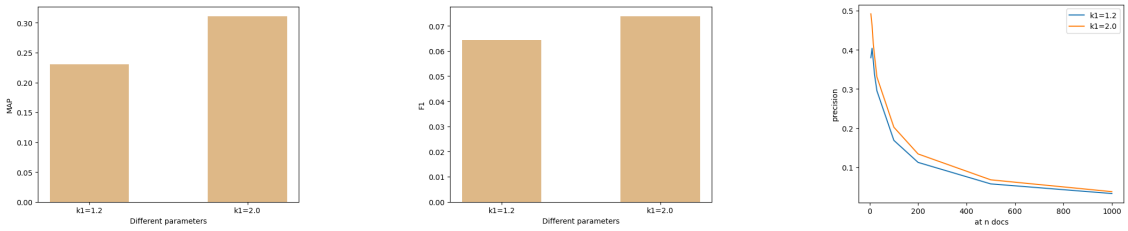
4 Parameter Setting of Ranking Functions

In the following experiments, we are going to see how different settings impacts the performance of information retrieval(MAP, F1, Precision at k, interpolated precision).

In this section, we are going to compare the result of different parameter setting for OKAPI, Language model with different smoothing methods.

4.1 OKAPI

In the following experiment, we are going to see whether different parameter settings have different effect on document retrieval.



(a) MAP (b) F1 (c) Precision at K

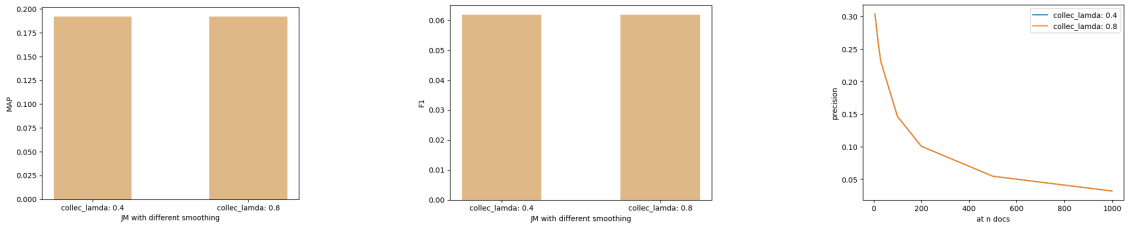
Figure 10: OKAPI performance with different parameter settings

In figure 10, it is observable that larger k1(k1=2.0) outperforms smaller k1(k1=1.2) since MAP, F1, and precision curve are all higher when k1=2.0.

The reason why larger k1 is performing better in this case is owing to the role of k1 as a parameter to scale document term frequency. Therefore, if a query contains terms with higher tf, then the OKAPI tf will be higher, so the more relevant documents will be likely to be found.

4.2 Language Model with Jelinek-Mercer Smoothing

This part will discuss the effect of different parameter setting of language model with Jelinek-Mercer smoothing (JM) on document retrieval performance.



(a) MAP (b) F1 (c) Precision at K

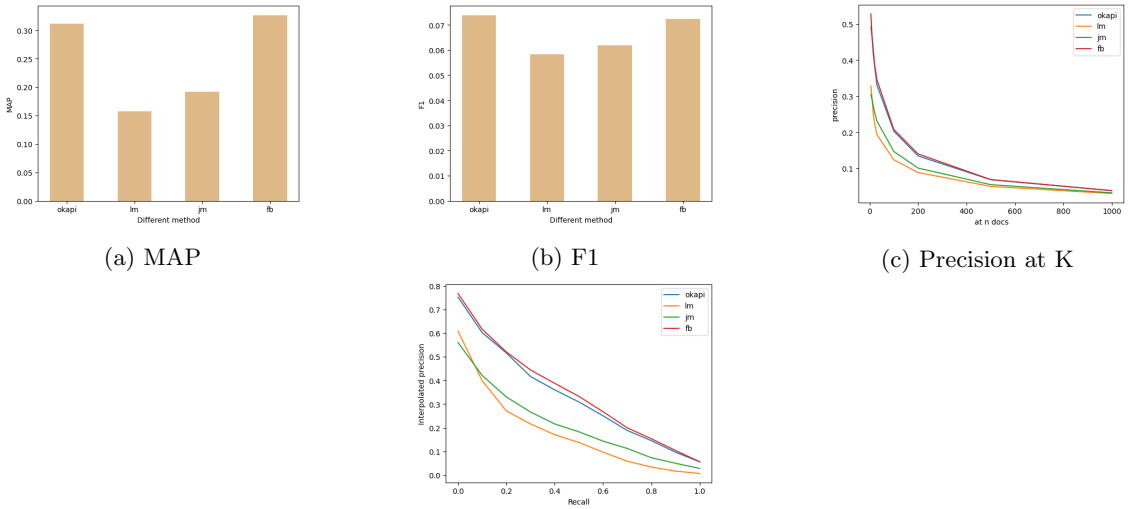
Figure 11: JM performance with different parameter settings

In figure 11, we can see that the performance of language model with JM smoothing has no significant change after increasing corpus lambda from 0.4 to 0.8.

The reason for the little effect of parameter change is due to large amount of documents in the corpus, so we are unlikely to get tf=0 in many documents. Thus, it is not helpful to increase retrieval performance by merely increasing corpus lambda.

5 Ranking Functions Comparison and Adding Pseudo Relevance Feedback

After discussing our adoption of stopwords or stemming or not, query type selection, and parameter setting for each ranking functions, we are going to compare the effect of the three ranking functions - OKAPI, Language Model with Laplace Smoothing (LM) and Language Model with Jelinek-Mercer Smoothing(JM), and several methods to improve their performance.



(a) MAP (b) F1 (c) Precision at K

(d) Interpolated Precision

Figure 12: Ranking Functions Comparison

5.1 Comparing OKAPI, LM and JM

From MAP, F1, Precision at K, and Interpolated Precision in figure 12, it is observable that OKAPI(blue curve) performs better than the other two functions according to MAP, F1, and precision at different levels of k. This is probably because the parameters for language model with laplace smoothing and jm smoothing haven't been optimized.

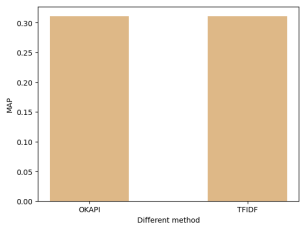
In addition, in comparison to LM(orange curve), JM(green curve) performs slightly better overall. However, based on precision at K and interpolated precision, it is obvious that LM does better at the first few searches. Therefore, if we care more about retrieving relevant documents at the first few searches, it is better to choose LM over JM.

5.2 Adding Pseudo Relevance Feedback on LM

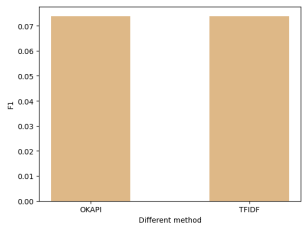
After figuring out LM doesn't perform better than other ranking functions overall, we try to come up with several methods to improve the performance of LM. Based on precision at k and interpolated precision curve in figure 12, it is noticeable that LM' s performance(orange curve) has significantly improved after adding pseudo relevance feedback(red curve).

5.3 Comparing OKAPI TF and Raw TF

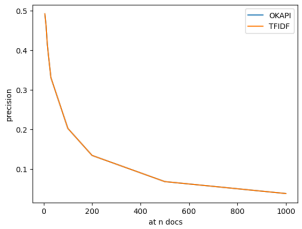
In the above experiment, we observe that OKAPI does better than other two functions. So, in this part, we want to see whether there is any difference in the performance of OKAPI TF and raw TF since they are both vector space model.



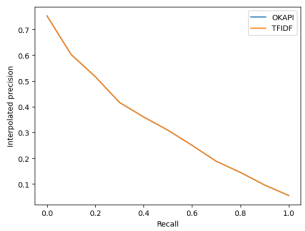
(a) MAP



(b) F1



(c) Precision at K



(d) Interpolated Precision

Figure 13: Comparing Vector Space Model - OKAPI TF and raw TF

According to figure 13, it is obvious that there is no significant difference of MAP, F1, Precision at k, and interpolated precision between OKAPI and raw TF. The reason is that the concepts contained in the two kinds of vector space model and parameter settings are almost the same. They all include the concepts of term frequency, document length normalization, and inverse document frequency (if we further multiply tf by idf). Additionally, the parameter settings are same for the two functions (k1=2.0, b=0.75) in order to compare on the same basis. Therefore, owing to the same function concepts and parameter settings, the performance of OKAPI TF and raw TF has no significant distinction.

6 Conclusion

Clearly, in the above experiment, we have demonstrated how stopwords filtering, stemming or not, query types, and parameter settings impact the performance of OKAPI, language model, and language model with jm smoothing in chapter 1 to 4. In chapter 5, we compare the performance among the three main models, finding that LM only does well at the first few searches. Thus, we try to improve LM's performance by introducing relevance feedback using the first 10 retrieved documents. And the results of adding feedback show that it is quite helpful to increase LM's performance in all aspects. Afterwards, we also see that OKAPI does well among the three main models, so we try to compare raw TF and OKAPI TF since they are very similar in their concepts. Consistently, according to figure 13, OKAPI TF indeed performs as good as raw TF.

In conclusion, after comparing OKAPI, LM, LM with JM smoothing, it is better to choose OKAPI TF, raw TF or LM with relevance feedback to gain better retrieval results.