CS 6200 Project

2018 Fall

Group member: Wandong Wu, Peijie Hao, Sicheng Wang
Instructor: Nada Naji, Rukmini Vijaykumar

1. Introduction

In this project, we designed and built an information retrieval system, evaluated and compared its performance in terms of retrieval effectiveness between different approaches. Dataset from CACM test-collection is used. Documents are parsed, ranked using multiple strategies. Snippets of [a selected result] are generated. Multiple evaluation matrixes are used to compare their performance. Advanced searching methods are provided, including exact match, best match and ordered best match with proximity.

2. Effort Division

BM25 Retrieval Model	Peijie Hao
JM Smoothed Query Likelihood Model	Peijie Hao
TF-IDF Retrieval Model	Wandong Wu
Lucene Retrieval Model	Wandong Wu
Pseudo-Relevance Feedback Model	Sicheng Wang
Snippet Generation and Query Highlighting	Sicheng Wang
Evaluation	All
Report	All

3. Techniques used

3.1 Document processing

First of all, we removed all the number at the end of each document. Then, we applied punctuation removing and case folding on all documents. We used unigram retrieval models. Documents are tokenized are tokenized into individual terms. Inverted indexes are generated for BM25 and JM-smoothed Query Likelihood model, term frequencies

and document frequencies are stored for TF-IDF model.

3.2 Query processing

Queries are processed in the same way as documents and tokenized into unigram tokens.

3.3 BM25

BM25 is a bag-of-words retrieval function that ranks a set of documents based on the query terms appearing in each document, regardless of the inter-relationship between the query terms within a document (e.g., their relative proximity). There are some variations of the scoring function for BM25, but the most common form is:

$$\sum_{i \in Q} \log \frac{(r_i + 0.5)/(R - r_i + 0.5)}{(n_i - r_i + 0.5)/(N - n_i - R + r_i + 0.5)} \cdot \frac{(k_1 + 1) f_i}{K + f_i} \cdot \frac{(k_2 + 1) q f_i}{k_2 + q f_i}$$

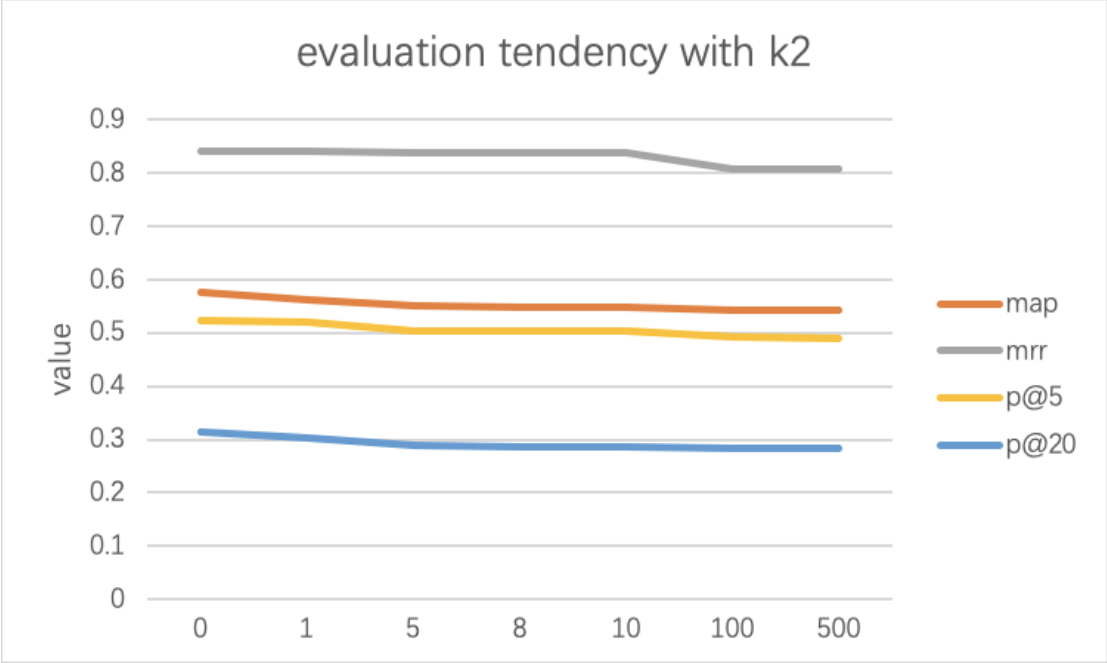
k1, k2, and K are parameters whose values are set empirically.

K is a more complicated parameter that normalizes the TF component by document length. Specifically

$$K = k_1((1 - b) + b \cdot \frac{dl}{avdl})$$

The k2 parameter selection for BM25 requires a lot of attempts. With the help of method evaluation, we generated the following table and curve:

K2	0	1	5	8	10	100	500
map	0.5764	0.5633	0.5507	0.5483	0.5471	0.5416	0.5412
mrr	0.8418	0.8393	0.8381	0.8378	0.8378	0.8080	0.8080
p@5	0.5230	0.5192	0.5038	0.50384	0.5038	0.4923	0.4884
p@20	0.3134	0.3028	0.2894	0.2875	0.2865	0.2836	0.2826



3.4 Jelinek-Mercer Smoothing

Smoothing is a technique for avoiding this estimation problem and overcoming data sparsity, which means that we typically do not have large amounts of text to use for the language model probability estimates.

Jelinek-Mercer method is known to be a simple smoothing technique, by setting the coefficient controlling the probability assigned to unseen words to λ.

$$\log P(Q|D) = \sum_{i=1}^{n} \log((1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|})$$

where fqi,D is the number of times word qi occurs in document D, and |D| is the number of words in D. cqi is the number of times a query word occurs in the collection of documents, and |C| is the total number of word occurrences in the collection.

3.5 TF-IDF

I use tf-idf to calculate the document term weight in the Vector Space model.

For term frequency(TF):

$$tf_{ik} = \frac{f_{ik}}{\sum_{j=1}^{t} f_{ij}}$$

where tfik is the term frequency weight of term k in document Di, and fik is the number of occurrences of term k in the document.

For inverse document frequency component (IDF):

$$idf_k = \log \frac{N}{n_k}$$

where idfk is the inverse document frequency weight for term k, N is the number of documents in the collection, and nk is the number of documents in which term k occurs. The effects of these two weights are combined by multiplying them (hence the name tf.idf). Given this, the typical form of document term weighting in the vector space model is:

$$d_{ik} = \frac{(\log(f_{ik}) + 1) \cdot \log(N/n_k)}{\sqrt{\sum_{k=1}^{t} [(\log(f_{ik}) + 1.0) \cdot \log(N/n_k)]^2}}$$

In this model, documents and queries are assumed to be part of a t-dimensional vector space,

$$D_i = (d_{i1}, d_{i2}, \ldots, d_{it}),$$

$$Q = (q_1, q_2, \ldots, q_t),$$

When the vectors are normalized so that all documents and queries are represented by vectors of equal length. The cosine measure is defined as:

$$Cosine(D_i, Q) = \frac{\sum_{j=1}^{t} d_{ij} \cdot q_j}{\sqrt{\sum_{j=1}^{t} {d_{ij}}^2 \cdot \sum_{j=1}^{t} {q_j}^2}}$$

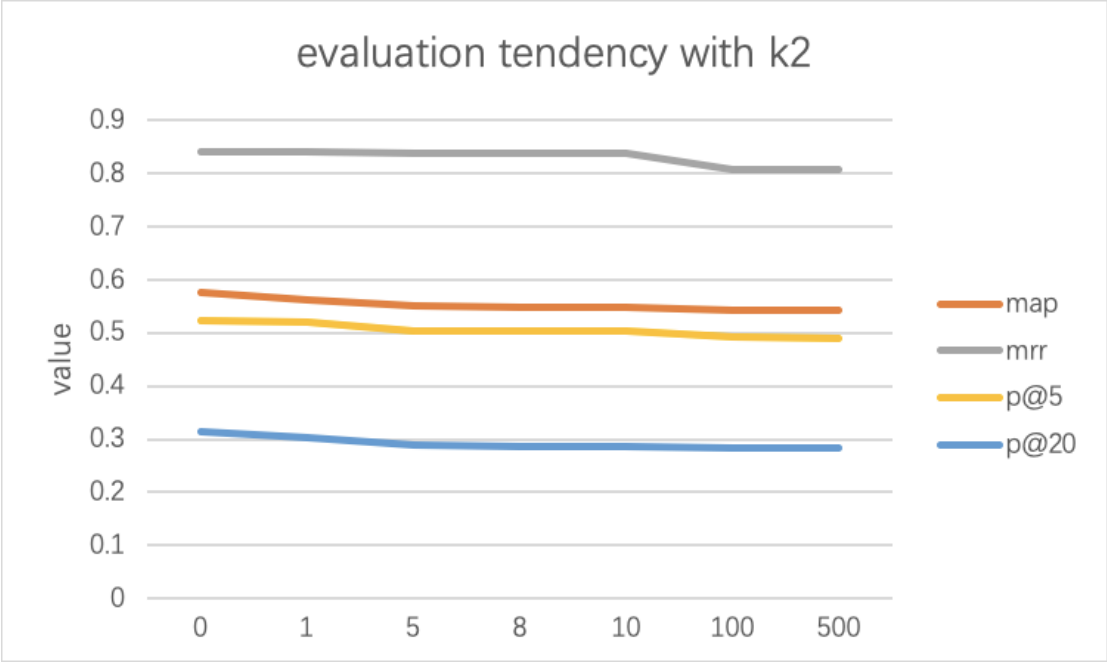
If a document has a higher Cosine value, it will have a higher rank.

3.6 Pseudo-Relevance Feedback

In this, we performed query expansion using Pseudo-Relevance Feedback technique. The assumed relevance document would be the top 10 documents that are retrieved from the original query. Ten expansion terms are generated after using stopword list to filter the top frequent terms that occur in the relevant documents.

We used BM25 ranking model to perform the first retrieval with the parameter k1 = 1.2, k2 = 0.0, b = 0.75. Stop words for most frequent word filtering is selected from the common_words.txt file and expanded by experience from multiple runs. The second retrieval is also performed using BM25 model. With the help of evaluation methods, we are able to obtain the following table and graph:

K2:2	0	1.5	3	5	10	100	500
MAP	0.5579	0.5721	0.5729	0.5683	0.5622	0.51586	0.51575
MRR	0.8699	0.8800	0.8768	0.8689	0.8485	0.77724	0.77724
P@5	0.5461	0.5346	0.5346	0.5423	0.5269	0.49615	0.49230
P@20	0.3413	0.3384	0.3326	0.3307	0.3265	0.28846	0.2875



Thus, the final selection for the k2 parameter of the second run of pseudo-relevance feedback is 2.0.

3.7 Stopping

The steps of stopping are as follows:

First, we generate a stop word list from the given file common_words.txt.

Second, after splitting the de-puncuationed document content, we traverse the content list and append the terms which are not in the stop word list into the final document content list.

Finally, the stopped documents are sent to the document ranking models to perform retrieval.

3.8 Stemming

From given files, we generated the query list and document list.

Results analysis:

Stemmed Query [portabl oper system]

Analysis:

After checking the top-ranked documents generated from the three queries, we discovered that the ranking score of top-ranked documents drops much faster than that without stemming. The most important term in the query should be “portable” and we are pleased to find that the documents which contain this term are ranked highest. The top 1 document contains the only exact match of the query. Thus the ranking models are efficient.

Stemmed Query [parallel algorithm]

Analysis:

After looking at the whole corpus, we found out that term “parallel” occurs a lot less than term ”algorithm”. “parallel” occurs in less than 100 documents while “algorithm” appears in more than 1000 documents. Looking at the top-ranked documents that are retrieved by the three models, most of them contain the term “parallel” and the topics look to be relevant to the query. However, the ranking score is very low for this query and there are even documents that are ranked 0, which indicates that the term “algorithm” is nearly ignored by the model.

Stemmed Query [perform evalu and model of comput system]

Analysis:

This query contains “and” and “of”. Because we do not use stopping list when performing stemmed query, these two terms will be calculated as the sub-score. Considering almost every document will contain these two terms, it will have no effect on the ranking. However, it will increase the score of almost every document. After checking the retrieval results, we could find the document score of this query will be significantly higher than those of other queries, which is consistent with our analysis

3.9 Dynamic Snippet generation

Having chosen or ranked the documents matching a query, we wish to present a results list that will be informative to the user. The standard way of doing this is to provide a snippet, a short summary of the document, which is designed so as to allow the user to decide its relevance. The two basic kinds of summaries are static and dynamic. Dynamic summaries display one or more ``windows'' on the document, aiming to present the pieces that have the most utility to the user in evaluating the document with respect to their information need.

Snippets are generated using dynamic snippet generating technique. The significance factor is determined by Luhn’s algorithm, substituting significant words by query terms. The significance factor for a text window is computed by dividing the square of the number of query terms in the span by the size of this window.

All query words are highlighted with “” tag during generation.

3.10 Evaluation Techniques

In this part, we use Precision, Recall, MAP, MRR, and P@K to evaluate the performance of our system.

Precision and Recall:

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

MAP (Mean Average Precision): MAP for a set of queries is the mean of the average precision scores for each query.

$$\text{MAP} = \frac{\sum_{q=1}^{Q} \text{AveP(q)}}{Q}$$

where Q is the number of queries.

MRR (Mean Reciprocal Rank): MRR is the average of the reciprocal ranks over a set of queries, and reciprocal of the rank at which the first relevant document is retrieved.

P@K: Precision at K

	MAP	MRR	P@5	P@20
BM25	0.576	0.842	0.523	0.313
JM Smoothing	0.464	0.863	0.446	0.233
Lucene	0.386	0.649	0.342	0.208
TF-IDF	0.065	0.094	0.023	0.013
PRF	0.571	0.883	0.535	0.337
BM25 Stopping	0.630	0.939	0.573	0.323
JM Stopping	0.464	0.863	0.446	0.232
TF-IDF Stopping	0.128	0.196	0.042	0.027

4. Results

4.1 Retrieval Models

Result for BM25 Model: task1/BM25.txt

Result for JM Smoothed Query Likelihood Model: task1/JelineKM_.txt

Result for tf-idf vector space Model: task1/TF_IDF.txt

Result for Lucene Search Engine: task1/LUCENE_.txt

4.2 Query Expansion

Result for Pseudo-relevance feedback: task2/PRF BM25_0 + BM25_2.txt

4.3 Stopping and Stemming

Result for stemmed BM25: task3/BM25stemmed.txt

Result for stemmed JelineKM: task3/JelineKMstemmed .txt

Result for stemmed tf-idf: task3/TF_IDFstemmed.txt

Result for stopped BM25: task3/BM25stopped.txt

Result for stopped JelineKM: task3/JelineKMstopped .txt

Result for stopped tf-idf: task3/TF_IDFstopped.txt

4.4 Snippet Generation and Query Highlighting

Result for snippet generation: task4/snippet_BM25.txt

4.5 Evaluation

Result for BM25 evaluation: task5/BM25_evaluation.txt

Result for JelineKM evaluation: task 5/JelineKM_evaluation.txt

Result for tf-idf evaluation: task 5/tfidf_evaluation.txt

Result for Lucene default evaluation: task 5/lucene_default_evaluation.txt

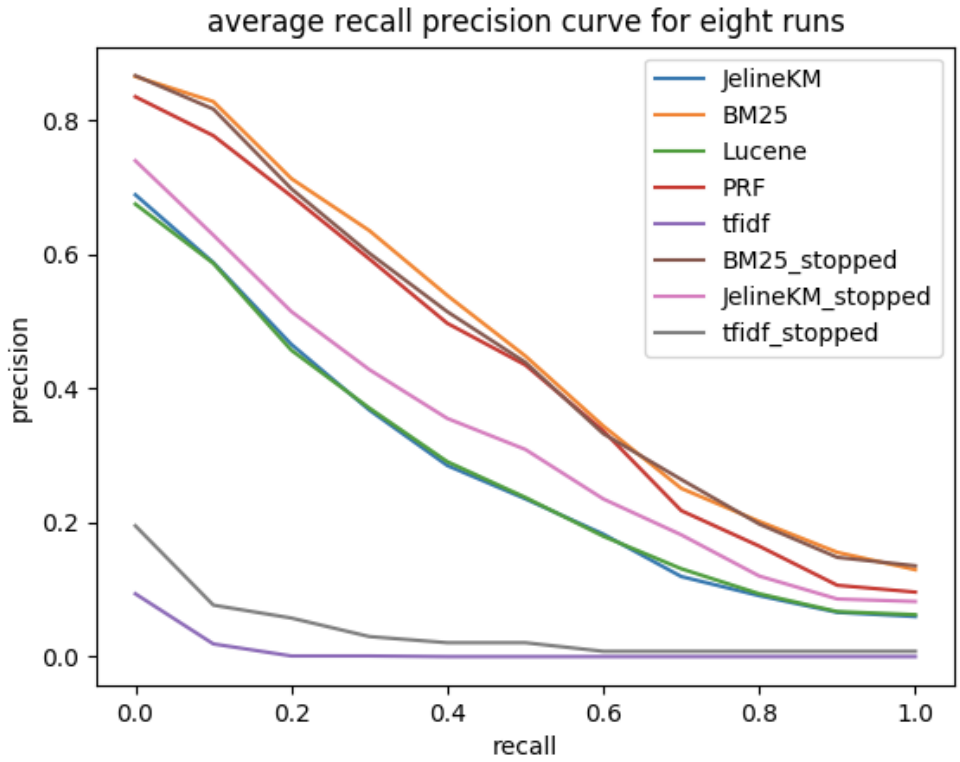
Result for stopped tf-idf evaluation: task 5/stopped_tfidf_evaluation.txt

Result for stopped BM25 evaluation: task 5/stopped_BM25_evaluation.txt

Result for stopped JelineKM evaluation: task 5/stopped_JelineKM_evaluation.txt

Result for pseudo relevance foodback evaluation: task 5/PRF_evaluation.txt

Result for recall-precision curve:



5. Conclusions and outlook

Conclusion:

Comparing the scores for ranking models, BM25 model performs the best, followed by pseudo-relevance feedback model. Lucene default model and Jelinek-M perform equally well while tf-idf model performs badly. Also, applying the stopping method could improve the retrieval system performance largely. Generally, the stemming procedure could bring improvement to retrieval results by expanding the query.

Outlook

In the future, pseudo-relevance feedback can be implemented with discriminative models and ML algorithms (e.g. topic models, LDA). Snippet generating could be upgraded to support stemming.

6. Bibliography

[1] Croft, W. B., Metzler, D., & Strohman, T. (2010). Search engines: Information retrieval in practice (Vol. 283). Reading: Addison-Wesley.

[2] Jesse Anderton (2018 July) Language Models. Retrieved from http://www.ccs.neu.edu/home/vip/teach/IRcourse/1_retrieval_models/slides/language_models.pdf

[3] Wikipedia (2016 Jan 30) Query likelihood model. Retrieved from https://en.wikipedia.org/wiki/Query_likelihood_model

[4] Wikipedia (2018 Oct 30). Tf-idf Retrieved from https://en.wikipedia.org/wiki/Tf%E2%80%93idf

[5] Schütze, H., Manning, C. D., & Raghavan, P. (2008). Introduction to information retrieval (Vol. 39). Cambridge University Press.