CS6200 Information Retrieval: Final project

Akshay Sadarangani, Ankita Nallana, Sam Wagenvoorde

December 2017

Fall Semester

Professor Nada Naji

1 Introduction

In our final project, we have implemented three retrieval models and assessed their performance. We have used the CACM corpus, its stemmed version, its stopped version (a stoplist was provided) and evaluated each model's performance. Additionally, we also generated document rankings for queries using Lucene.

For all the retrieval models that we implemented, we evaluated them using Mean Average Precision, Mean Reciprocal Rank and generating their Precision and Recall Tables. We used the relevance judgements provided in *cacm_rel.txt*.

We have provided them in their respective folders (by retrieval models).

2 Effort Division

Task	**Who**
Generated corpora (regular, stopped, stemmed)	Ankita Nallana
Lucene Retrieval Model	Sam Wagenvoorde
BM25 Retrieval Model	Akshay Sadarangani
TF-IDF Retrieval Model	Ankita Nallana
Smoothed Query Likelihood Model	Akshay Sadarangani
Pseudo Relevance Feedback	Sam Wagenvoorde
Snippet Generation and Query Highlighting	Ankita Nallana
Evaluation framework	Ankita Nallana
Documentation	Everyone

Table 1: Effort Division

3 Techniques used:

3.1 Query Refinement

For query refinement, we used text parsing and transformation to narrow down the query. We tokenized the query into query terms and removed all punctuation from them. Additionally, for the baseline runs with stopping, we removed stopped words from the query as well. This facilitated better document matching and scoring.

3.2 Snippet Generation

For snippet generation we employed a very simple technique.

After retrieving the ranked documents(per query), we matched the terms in each sentence (in every document) against the terms in the query. We created a window where five words before the earliest matched term (in a sentence) and five words after the last matched term were included (and the terms in between these two). This way we could provide context and meaning to our results as sentences were retrieved and not just the query terms.

4 Project Implementation Details:

4.1 Generating the corpora and other data

We read the entire corpus and stored all content in pickle files. We removed the column of numbers that were present at the end of each document, stripped it of all punctuation and converted all text to lowercase. Term Frequencies, Document Frequencies/Inverted Indexes, IDF scores were pre-computed and stored in pickle files. Pickle files were also generated for frequencies of query terms in the corpus and documents for Smoothed Query Likelihood Model.

4.2 Parsing/Processing the corpora and the queries

Both the queries and the corpora were processed the same way - punctuation marks were removed and sentences were tokenised into individual terms. As and when required by the retrieval models, every term's frequency - be it across the corpus, document or query - was fetched(/computed in the case of queries) and used for scoring the documents. Once the documents are scored, the top 100 scoring documents are fetched. We assigned query ids in an incremental fashion while processing them as a batch.

The NLTK library was used to generate tokens while parsing sentences/corpora

4.3 Retrieval Models

Task 1 is concerned with building our own retrieval systems. The four retrieval models are:

- TF-IDF
- BM25
- Smoothed Query Likelihood Model
- Lucene's Default Model (Lucene is an indexing and search library)

4.3.1 TF-IDF

Perhaps one of the most popular weighting/scoring schemes, TF-IDF stands for Term Frequency - Inverted Document Frequency. Term Frequency has been computed as the number of times a term occurs in a document divided by the number of terms in that document. Inverse Document Frequency is the logarithm(to the base e) of total number of documents in a collection divided by the number of documents containing a term. The final TF-IDF score for a term is generated by multiplying both these values.

4.3.1 Query by Query analysis - For stemmed corpus

- Query ID = 1, *portabl oper system*
  The first two documents have a very small difference in their scores which makes them quite similar - i.e. similar term frequencies and idf scores of both documents(two and three of the query terms occur in the two documents respectively). However, the third document has a significantly lower score which tells us that perhaps the query term(s) do not appear as frequently - just one query term *oper* and only once - in that document (and the following documents).
- Query ID = 3, *parallel algorithm*

  The second and third document retrieved have equal scores meaning they are both equally likely to be retrieved and that in their documents the terms are equally spread(The term *parallel* occurs once and their document lengths are the same).

- Query ID = 7, Query = *parallel processor in inform retriev*
  The difference between the scores of the documents is quite large. This tells us that the terms are not too closely distributed across the corpus (i.e. documents) and that their frequencies and document lengths vary greatly.

4.3.2 BM-25

"In information retrieval, Okapi BM25 (BM stands for Best Matching) is a ranking function used by search engines to rank matching documents according to their relevance to a given search query. It is based on the probabilistic retrieval framework developed in the 1970s and 1980s by Stephen E. Robertson, Karen Spärck Jones, and others." - Wikipedia

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

BM25 Formula

Query-by-Query Analysis - For stemmed corpus

- Query ID = 1, Query = *portabl oper system* This query has a difference of BM25 score between rank 1 and rank 2 documents of 8.76. This shows us that there is a large gap between the first two ranks which implies that there are very few documents in the collection which have all terms of the query present in a single document.

- Query ID = 3, Query = *parallel algorithm* This query has a difference of BM25 score between rank 1 and rank 2 documents of 0.25. This shows that the two documents are equally likely to be retrieved and the query terms are evenly spread out in the corpus. It has a gradual fall in score vs query 1 which had a drastic fall.

- Query ID = 5, Query = *appli stochast process* This query has a difference of BM25 score between rank 1 and rank 2 documents of 0.9. This is not a very huge difference considering the range and spread of scores. This type of distribution of scores points towards uniformly spready query terms in the corpus. This can indicate that the query is not very relevant to the corpus as singular terms are being matched in most cases.

4.3.3 Smoothed Query Likelihood Model - for stemmed corpora

"The query likelihood model is a language model used in information retrieval. A language model is constructed for each document in the collection. It is then possible to rank each document by the probability of specific documents given a query. This is interpreted as being the likelihood of a document being relevant given a query." - Wikipedia

$$\log \mathbb{P}(Q|D) = \sum_{i=1}^{n} \log((1 - \lambda)\frac{fq_i, D}{|D|} + \lambda \frac{cq_i}{|C|})$$

Smoothed QLM Formula

Query-by-Query Analysis - For stemmed corpus

- Query ID = 1, Query = *portabl oper system*
  This query has a score of -2.03 (sum of very small logs is negative) which is very small. This means that the frequency of the terms in the corpus is very low.
- Query ID = 3, Query = *parallel algorithm*
  This query has a score of -5.95. It is bigger (higher rank is a lower number in QLM) than -2.03 but it is still low. This means that the terms in this query have a higher frequency than query 1 but still not very high.
- Query ID = 7, Query = *parallel processor in inform retriev*
  This query has a high score of -18.78. This is probably because of the high frequency of the word "in" in the corpus. There might be other terms in the query which may have contributed to this high score as well but "in" gives the highest score.

5 Pseudo Relevance Feedback

Task 2 is concerned with Pseudo Relevance Feedback (PRF), a form of Automated Query Expansion. The expansion terms are based on the top retrieved documents for initial queries, the method assumes that the top k retrieved documents are relevant. (1) provides a clear visual explanation of the process.

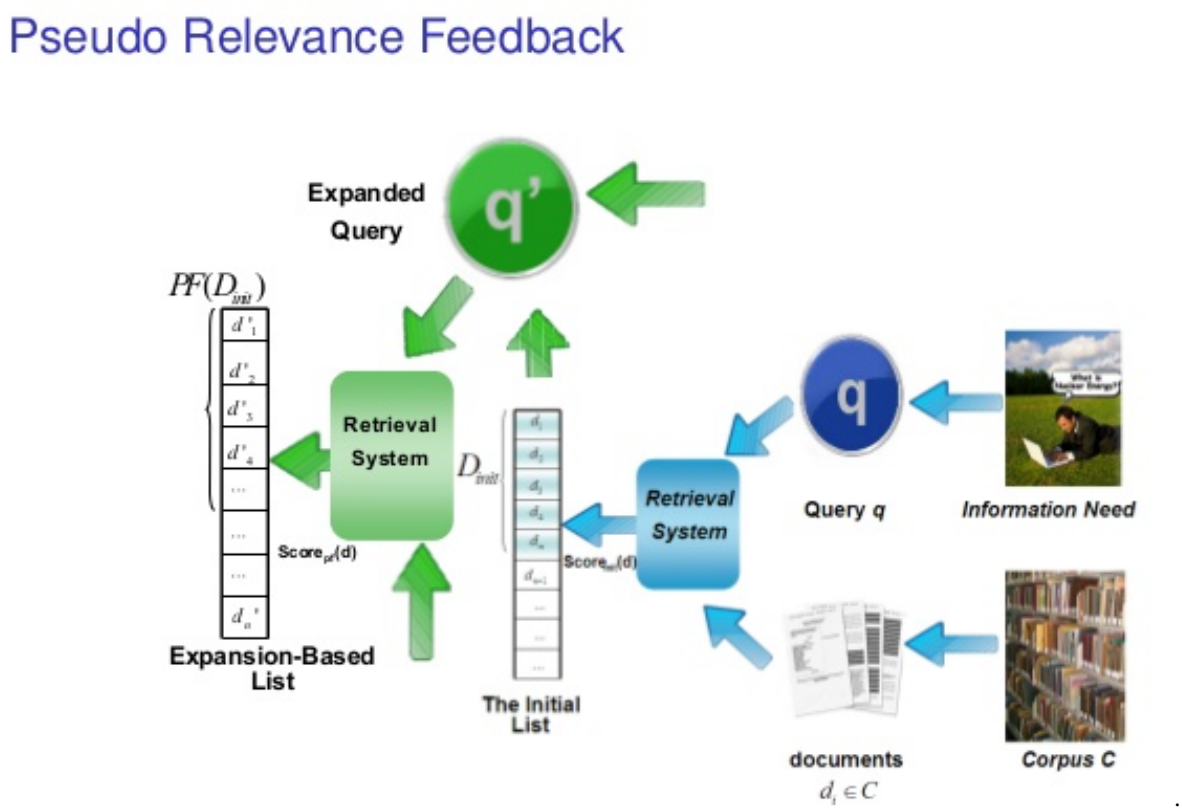

Figure 1: Pseudo relevance feedback query expansion explained

A general rule of thumb is not to choose a large set of terms - that could lead to very bad performance as a lot of noisy, non-topical terms may be chosen (2). PRF has been thoroughly researched and seems to be an effective way of improving ranking however, it is rarely implemented in practice, most likely due to its unpredictable nature (3). Also, the assumption that the top k documents are relevant does not hold in reality - many expansion terms identified in traditional approaches are indeed unrelated to the query and harmful to the retrieval (4). We performed a PRF on the results of the TF-IDF base run. The results from this query expansion are similar to the results without the PRF, this can be due to multiple reasons:

- PRF selects unrelated terms for query expansion
- The assumption that the top k documents are relevant does not hold
- The documents are too short

The latter requires some explanation, if the documents are short, then the query expansion terms are likely to appear only appear in the top k documents (and not in the others). Thus retrieving with the expanded query will recover the same document. In order to see more effect with PRF it should be used on lengthier documents, so that more relevant, not in the query, terms can be found.

6 Results

MAP scores, MRR scores, P@K ($K = 5$ and $K = 20$) have been provided per model per query. Additionally, Precision and Recall tables have also been provided per model per query.

Let us consider the MAP and MRR scores for a few queries - Query 1, 8, 17, 24, 57

Query 1 : *What articles exist which deal with TSS (Time Sharing System), an operating system for IBM computers?*

Retrieval Model	MAP score	MRR score
TF-IDF	0.006805367	0.027027027
BM25	0.008232976	0.2
Smoothed QLM	0.0107069386	0.5

Query 8 : *Addressing schemes for resources in networks; resource addressing in network operating systems*

Retrieval Model	MAP score	MRR score
TF-IDF	0.0064980	0.2
BM25	0.0075231495	1.0
Smoothed QLM	0.0089773929	1.0

Query 17 : *Optimization of intermediate and machine code*

Retrieval Model	MAP score	MRR score
TF-IDF	0.01948801	0.1428571
BM25	0.0225083990	1.0
Smoothed QLM	0.0228621228	1.0

Query 24 : *Applied stochastic processes*

Retrieval Model	MAP score	MRR score
TF-IDF	0.039289678	0.5
BM25	0.0467166925	1.0
Smoothed QLM	0.0406759827	0.2

Query 57 : *Abstracts of articles: J. Backus, "Can programming be liberated from the Von Neumann style? A functional style and its algebra of programs", CACM 21 Re Millo, R. Lipton, A. Perlis, letter to ACM Forum, CACM 22 (1979), 629-630 Backus, J. De Millo, R. Lipton, R. Perlis, A.*

Retrieval Model	**MAP score**	**MRR score**
TF-IDF	0.0004089	0.0011
BM25	0.0006364309	0.0018050541
Smoothed QLM	0.0003084890	0.0009132420

7 Conclusion

From the scores above, we can see that **QLM** has the higher MAP scores and high MRR scores (even reaching 1 in some cases) in most cases. However, it is worth noting that **BM25** also shows very promising results and finishes a close second. The difference of 0.5 at times in their MRRs seems to suggest a rank difference of only 1. TF-IDF in comparison does not seem to do a great job.

References

[1] L. Zighelnic, "Robust query expansion based on query drift prevention," 2009.

[2] K. Raman, *Improving Pseudo-Relevance Feedback: "Multiple Paths, Same Goal*. Indian Institute of Tecnology, Bombay, May 2010.

[3] W. B. Croft, D. Metzler, and T. Strohman, *Search engines: Information retrieval in practice*, vol. 283. Addison-Wesley Reading, 2010.

[4] G. Cao, J.-Y. Nie, J. Gao, and S. Robertson, "Selecting good expansion terms for pseudo-relevance feedback," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 243–250, ACM, 2008.