INFORMATION RETRIEVAL Fall'16

PROJECT REPORT

Group Member:

Jiaxin Yang

Mingchao Wu

Deepak Surana

Instructor:

Prof. Nada Naji

Introduction

In task 1, four different retrieval models were implemented, which were BM25, tf-idf, Cosine Similarity and Lucene. In task 2, we chose pseudo relevance feedback as our expansion technique, and used BM25 in this run. In task 3, indexes were generated for the stopping and stemmed corpus, then BM25 was used in these two runs.

In evaluation phase, the seventh run is generated by using BM25 and combining query expansion technique with stopping. MAP, MRR, P@K and Precision & Recall were evaluated for all seven runs. They have been compared on the model parameters that compares two models in overall (like, MAP and MRR) as well as on parameters that compares two models for each Query (like, P@K, Precision and Recall). Details of the result is in the 'Result' section of this report.

Jiaxin Yang implemented three retrieval models in Task 1, stopping and stemming in Task 3 and wrote the corresponding part in report. Mingchao Wu implemented the pseudo relevance feedback in Task 2 and wrote the corresponding part in report. Deepak Surana implemented the Lucene base run in Task 1, Phase 2: Evaluation, snippet generation and wrote the corresponding part in report.

Literature and resources

Implementation in this project was mainly based on the book [1], and corresponding slides. Implementation of query expansion was based on chapter six, the pseudo relevance feedback. Refer to the 'Bibliography' section for a full list of resources used. Lucene as a third part tool was used for one of the base runs in task one.

Implementation and discussion

Phase 1: Indexing and Retrieval

Task1

Four search engines were implemented in this task. Lucene's default setting is used.

For tf-idf model, document scores were calculated as sum of tf * idf. Both tf and idf are logarithmically scaled.

For cosine similarity model, the implementation was based on this video [2]. The query vector was calculated as normalized logarithmically scaled idf. The document vector was calculated as normalized logarithmically scaled tf. For those query terms that don't appear in the document, that term component is zero. Document score was calculated as dot product of the two vector, as the formula below.

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

For BM25, the implementation was based on the formula below, as what we learned in class. For those queries that don't have relevance judgement, R and ri were set to zero. We chose k1 = 1.2, k2 = 100, b = 0.75.

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

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

For all three retrieval models, term-at-a-time was used. And a score accumulator was used for recording score of each document.

Task2

Pseudo Relevance Feedback was chose as our query expanding technique. It performs two round retrieval. The top 5 retrieved documents in the first round were assumed as relevant documents. Term frequency were counted in the 5 documents. Then a term was added to the query if it meets the following three conditions: 1. it is not a common word. 2. Its term frequency is as least 6. 3. It is not in the original query. Different settings for minimum term frequency were tried. And 6 was chose for producing appropriate amount of expanding terms. BM25 was used in this run.

Task3

For stopping, a new corpus was generated and indexed with all stopping words removed. Stopping words were also removed in all queries. BM25 was used in this run.

For stemming, index was generated for the stemmed corpus. BM25 was used for seven stemmed queries in this run.

Phase 2: Evaluation

Firstly, added one more base-run to our set of different IR models by combining both Query Expansion and stopping with BM25.

Basically, evaluation can be done by comparing overall model/base-runs performance or by judging them on result of each query that they gives. Parameters that we have considered here to evaluate whole models for a given set of Queries are MAP (Mean Average Precision) and MRR (Mean Reciprocal Rank). Whereas, to evaluate Query-by-Query results for different base-runs parameters considered are P@K (Precision at Kth rank document in the result for a query), Precision (how well it is doing at rejecting

non-relevant documents) and Recall (how well the search engine is doing at finding all the relevant documents for a query).

1. *Parameters to evaluate query-by-query:*

$$Precision = \frac{Number\ of\ Relevant\ documnets\ Retrieved}{Total\ Documents\ Retrieved}$$

$$\text{Recall } = \frac{Number\ of\ Relevant\ documnets\ Retrieved}{Total\ Relevant\ documents}$$

$$\text{Average Precision } = \frac{Sum\ of\ Precision\ at\ Relevant\ documents}{Total\ Relevant\ documents}$$

$$\text{Reciprocal Rank } = \frac{1}{Rank\ of\ first\ Relevant\ document}$$

$$Precision\ @\ K\ = Precision\ for\ Kth\ rank\ document\ in\ the\ result\ of\ a\ query$$

2. *Parameters to evaluate overall performance of a retrieval model:*

$$Mean \text{ Average Precision (MAP) } = \frac{Sum\ of\ Average\ Precision\ for\ each\ query}{Total\ number\ of\ queries\ executed}$$

$$Mean\ Reciprocal\ Rank \text{ (MRR) } = \frac{Sum\ of\ Reciprocal\ Rank\ for\ each\ query}{Total\ number\ of\ queries\ executed}$$

From, implementation point of view the resultant list of results for each query has been gathered from different base-runs. Taking a single base-run at a time, for each query result the query-by-query evaluation measures are calculated and stored by appending the two columns Precision and Recall corresponding to each document in the result (using the above mentioned mathematical formulae). This way, the intermediate values of Precision and Recall has been stored for each document in the ranked result of a query. P@K is simply the value of Precision for the Kth rank document.

The run to calculate these parameters for individual query also gives the average Precision value for the whole query result. Collecting all those Average Precision values for a baseline-run with multiple queries, Mean Average Precision has been calculated accordingly.

For each baseline-run, evaluation of every single query result returns the Reciprocal Rank for that query-result. By having all these reciprocal ranks for a set of queries, Mean Reciprocal Rank has been calculated for the whole model by averaging

them.

Snippet Generation

Snippet Generation is a form of Text Summarization. The approach we have taken here was described by H.P. Luhn.
We are considering BM25 has the search model which will generate 100 top result documents (same as in Task 1).

1. To generate snippets of the resultant documents, we have considered Top 10 documents.
2. For each document, the procedure is to parse the html file for the document, get the title and the body text.
3. Splitting the body text into sentences on a period (.).
4. For each sentence, calculating the significant factor as described by Luhn
5. In each sentence, categorizing words as either Significant or not.
6. Taking a text span from the sentence which has at most 4 non-significant words.
7. Creating an object, which will have the array of words in the sentence, its significance factor, start index of text span and end index of text span.
8. After getting the object for each sentence in a particular document, taking the 2 sentences with highest significant factor.
9. As an snippet, showing the DocID at top (for reference), then the document title and the two most significant sentences in the next two lines

For highlighting, the query text we have looked into snippet and covered the query words with square brackets “[ ]” .

Query-by-query analysis

The 13rd query is "code optimization for space efficiency". Document 2491 was ranked 17 in BM25 model, but ranked 6 in BM25 with stemming. The stemmed query is "code optim for space effici ". The observation is that in the original document, the word appears as "optimizations", thus the term "optimization" in the query doesn't match the document. But in stemmed document, the word appears as "optim", thus match the query. And "optimization" has relatively high idf, which boost the score remarkably.

The 23rd query is "Distributed computing structures and algorithms". Document 3148 was ranked 1 in BM25 model, but ranked 8 in BM25 with stemming. The stemmed query is "distribut comput structur and algorithm". The observation is that in the original document, the two words appear many times exactly as "distributed" and "computing". In stemmed corpus, many words such as "distribute", "computer" are stemmed as "distribut" and "comput". Thus there are more documents (higher df) contain the two terms. While in original corpus, "distributed" and "computing" has much lower df. The two words become less significant in stemmed corpus. So the score of document 3148 got weakened.

Evaluation

Snippet Generation

Results

Retrieval Model	Mean Average Precision
TFIDF	0.315785
Cosine_Similarity	0.29712
Lucene_Search	0.345313
BM25	0.455223
BM25_Expanding	0.437943
BM25_Stopping	0.44102
BM25_Stemming	0.46441
BM25_Stopping_Expanding	0.419205

Table 1 Mean Average Precision

Retrieval Model	Mean Reciprocal Rank
TFIDF	0.520306
Cosine_Similarity	0.495747
Lucene_Search	0.580845
BM25	0.675569
BM25_Expanding	0.651999
BM25_Stopping	0.652853
BM25_Stemming	0.761905
BM25_Stopping_Expanding	0.631276

Table 2 Mean Reciprocal Rank

Table 1 and 2 shows the mean average precision (MAP) and Mean Reciprocal Rank (MRR) of the seven runs plus the stemming run. For Precision&Recall and P@K, the tables of all queries in all runs are in the 'evaluation result' folder. The stemming run cannot be compared with others. Because it only has seven queries.

Conclusions and outlook

As shown in the table, BM25 has the highest MAP and MRR, then Lucene, tf-idf and Cosine Similarity. BM25 has the best performance as expected. It is the most desired retrieval model when relevance judgement is available. Lucene, as a mature open source retrieval system, has the second best performance in this corpus. Cosine Similarity performs worst in this corpus. For query expanding, MAP and MRR are a little lower than the base run. Perhaps it is because for many queries, there are not many

relevant documents in the corpus. The assumed relevant top 5 documents may not actually be relevant in some occasions. Pseudo Relevance Feedback works better if there are many relevant documents in the corpus, for example 100, then the assumed relevant documents have bigger chance to be actually relevant. For stopping, the performance is slightly worse than the base run, which means for this corpus, stopping doesn't make much difference. Probably because in BM25, those common words don't get remarkable scores, and most documents contain some common words, which gave them similar score boost for common words. BM25 with Pseudo Relevance Feedback and Stopping has worse performance, which means combining two worse techniques makes an even worse performance.

Outlook

Pseudo Relevance Feedback doesn't work well in this corpus, other query expansion technique can be implemented like Thesauri and Inflectional variants. Also for cosine model, there are many variation of implementation. The Lucene version used in this project is based on cosine model, and has better performance than our cosine model. So improvements can be made in our cosine similarity model.

Bibliography

[1] Croft, W. B., Metzler, D., & Strohmann, T. (2010). *Search engines*. Pearson Education.

[2] 19 - 7 - Calculating TF-IDF Cosine Scores-Stanford NLP-Professor Dan Jurafsky & Chris Manning https://www.youtube.com/watch?v=E3shpvJUZ84