Query Expansion and Snippets Generation

Abhay Kasturia
Rashmi Dwaraka

Table of Contents

INTRODUCTION

The project is a search engine which uses query expansion techniques and snippet generation to improve user's experience with the search engine.

Query expansion is the process of reformulating a user query to improve retrieval performance in information retrieval operations. It helps to improve user experience as it increases the likelihood of retrieving relevant documents.

Snippet is a short summary of the document, which is designed so as to allow the user to decide its relevance. The snippet in our project consists of a few lines containing a term or all terms from the query highlighted. This helps the user to find a more relevant document according to his requirements.

The project uses multiple query expansion techniques .The performance before query expansion and after it, as well as the performance among different query expansion techniques are measured and compared using various parameters such as Avg. Precision, Recall, F1 score, MRR, MAP and NDCG.

The project uses BM25 ranking model to retrieve top results for an expanded query. The top results are then displayed with snippets to increase users' understanding of the document and possibly increase their click through ratio and enhance their interaction with the search engine.

METHODOLOGY

1. EXPAND QUERY BY WORD VARIANTS IN THE SAME STEM CLASS

Brief: The user query is expanded by word variants in the same stem class. This is same as the query-based stemming.

How: In this method, we expand the original queries by adding all the word variants in one same stem class for each term in the query. That expanded query is then run on Lucene BM25 Model to compute the top relevant documents.

2. GLOBAL ANALYSIS

Brief: The user query is expanded by checking word association/co-occurrence of the query terms with the vocabulary of the entire data collection. Word association is measured under different metrics and in certain text window. In our project, we consider the text window to be each the whole individual document. Below are the association metrics we use for finding the co-occurrence of each query term with the entire vocabulary:

2.1 DICE'S COEFFICIENT (DICE)

How: Term association measures are an important part of many approaches to query expansion and DICE's co-efficient is one of the metric to calculate term association measure. The formula to calculate DICE's co-efficient between two terms "a" and "b" is:

$$\frac{2 \cdot n_{ab}}{n_a + n_b} \overset{rank}{=} \frac{n_{ab}}{n_a + n_b}$$

Where “nab” is the frequency where both “a” and “b” occur in a single text window, in our case in the same document. And similarly “na” is the frequency of “a” occurring in a document and “nb” is the frequency of “b” occurring in a document. Based on this formula we calculate the association measure between a query term and a word from the vocab. We take the top 15 words with the highest scores and consider them as most associated with the query term. The query is then expanded by calculating and including top 15 most strongly associated words for each query term. That expanded query is then run on Lucene BM25 Model to compute the top relevant documents.

2.2 MUTUAL INFORMATION MEASURE(MIM)

How: MIM is one of the metric to calculate term association measure. The formula to calculate MIM co-efficient between two terms “a” and “b” is:

$$\log \frac{P(a,b)}{P(a)P(b)} = \log N.\frac{n_{ab}}{n_a.n_b} \overset{rank}{=} \frac{n_{ab}}{n_a.n_b}$$

Where, “nab”, “na” and “nb” carry the same representation as in DICE’s. And N is the total size of the text window, which in our case is a single document. Based on this formula we calculate the association measure between a query term and a word from the vocab. We take the top 15 words with the highest scores and consider them as most associated with the query term. The query is then expanded by calculating and including top 15 most strongly associated words for each query term. That expanded query is then run on Lucene BM25 Model to compute the top relevant documents.

3. LOCAL ANALYSIS: QUERY EXPANSION BY PSEUDO RELEVANCE FEEDBACK

Pseudo relevance feedback is a local-based automatic query expansion technique, in which the Top N returned documents for the original query will be considered as relevant to the query, and terms to expand the original query will be generated by using this Top N documents. Based on different retrieval models, we have different methods to implement pseudo relevance feedback.

3.1. VECTOR SPACE MODEL BASED PSEUDO RELEVANCE FEEDBACK:

How: The Rocchio algorithm can be implemented as follows:

Step1: Represent each query and document by a vector of size |V|, where V is the vocabulary in the data corpus. Each entry in the vector is the tf.idf score of the corresponding term.

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

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

Similarly, we calculate tfi.df for each query as follows:

$$okapi\_tf_{w,q} = \frac{tf_{w,q}}{|q|}$$

Where, |q| is the length of the query.

Step2: Run BM25 retrieval models over the original query.

Step3: Regard the Top N returned documents (we set N=50 here in the experiments) as relevant documents, and all others as non-relevant documents.

Step4: Retrieve the extended query whose vector can be computed by the following equation:

$$q_j' = \alpha \cdot q_j + \beta \cdot \frac{1}{|Rel|} \sum_{d_i \in Rel} d_{ij} - \gamma \frac{1}{|Nonrel|} \sum_{di \in Nonrel} d_{ij}$$

Where, qj is the jth entry in the vector representing q. qj' is the updated entry for the jth entry in expanded query qj'. Rel indicates the set of relevant documents, and Nonrel indicates the set of non-relevant documents. |Rel| is the size of relevant document set, and the same for |Nonrel| for non-relevant document set. dij is the jth entry in vector for document di.

Step5: Re-run BM25 retrieval model to get the ranking results for expanded query. Determined by the entry tf.idf score, we consider Top 50 terms to form the expanded query. We set α=8, β=16, and γ=4 in our experiment.

3.2. LANGUAGE MODEL BASED PSEUDO RELEVANCE FEEDBACK:

Relevance language models provide a formal retrieval model for pseudo relevance feedback and query expansion. The procedure can be described as follows.

Step1: Rank documents using the query likelihood score for query q.

To estimate P(q|D), we use unigram query likelihood model to estimate P(q|D), which can be computed as:

$$\log P(q \mid D) = \sum_{i=1}^{n} \log(P(q_i \mid D)$$

Where, qi is every term in query q. To solve the zero probability problem, we use Dirichlet smoothing for computing log(P(w|D)).
Query Likelihood language model can be represented as below:

$$\log P(q_i \mid D) = \log(\frac{f_{qi,D} + \mu \frac{c_{qi}}{|C|}}{|D| + \mu})$$

And we set μ = 2000 for our project.

Step2: Select top N documents to be the set of C.

Step3: Calculate the relevance model probabilities P(w|R) using the estimate for P(w, q1 … qn) where R represents the true distribution of the relevance model for query. P(w|R) can be computed as:

$$P(w \mid R) \approx \frac{P(w, q_1 \ldots q_n)}{P(q_1 \ldots q_n)}$$

$$P(q_1 \ldots q_n) = \sum_{w \in V} P(w, q_1 \ldots q_n)$$

$$P(w, q_1 \ldots q_n) = \sum_{D \in C} P(D) P(w \mid D) \prod_{i=1}^{n} P(q_i \mid D)$$

P(D) is regarded as uniform and can be ignored. C is the set of relevant documents. We apply Jelinek-Mercer smoothing here for P(qi|D). But, we do not apply smoothing technique for P(w|D) calculation.

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

Where, cqi is the number of times a query term occurs in the collection of documents, and |C| is the total number of word occurrences in the collection. We set λ = 0.5 for our calculations in this project.

Step4: Rank documents again using the KL-divergence score as follows:

$$\sum_{w} P(w \mid R) \log P(w \mid D)$$

Where, w here is the Top N (N=15) words in terms of their P(w|R) value.

4. Experimental Evaluation

One of the primary distinctions made in the evaluation of search engines is between effectiveness and efficiency.
Effectiveness - measures the ability of the search engine to find the right information and Efficiency - measures how quickly this is done.
For a given query, and a specific definition of relevance, we can more precisely define effectiveness as a measure of how well the ranking produced by the search engine corresponds to a ranking based on user relevance judgments. Efficiency is defined in terms of the time and space requirements for the algorithm that produces the ranking.

4.1 Document Collection Analysis

Data Collection Analysis	
Total number of documents	84660
size of collection vocabulary	101701
total number of word occurrences in the collection	21478667
average document length	253.71
average query length (first 10 queries)	16.2

4.2 Evaluation

4.2.1 Original Query Terms

Original_Query.txt

4.2.2 Query Output For Expanded Queries

4.2.2.1 Stem Classes

4.2.2.2 EXPANDED QUERY USING DICES MEASURE

4.2.2.3 EXPANDED QUERY USING MIM

4.2.2.4 EXPANDED QUERY USING ROCCHIO ALGORITHM

4.2.2.5 EXPANDED QUERY USING LANGUAGE MODEL

In our project, we evaluate the relevance of retrieved documents based on below measures:

1) **Precision:**
   Precision (P) is the fraction of retrieved documents that are relevant.

$$Precision = \frac{|A \cap B|}{|B|}$$

2) **Recall:**

Recall (R) is the fraction of relevant documents that are retrieved

$$Recall = \frac{|A \cap B|}{|A|}$$

	Relevant	Non-Relevant
Retrieved	$A \cap B$	$\overline{A} \cap B$
Not Retrieved	$A \cap \overline{B}$	$\overline{A} \cap \overline{B}$

3) F1 Score:

The F measure is an effectiveness measure based on recall and precision that is used for evaluating classification performance and also in some search applications. It has the advantage of summarizing effectiveness in a single number. It is defined as the harmonic mean of recall and precision, which is

$$F = \frac{1}{\frac{1}{2}(\frac{1}{R} + \frac{1}{P})} = \frac{2RP}{(R+P)}$$

4) MRR:

The reciprocal rank measure has been used for applications where there is typically a single relevant document. It is defined as the reciprocal of the rank at which the first relevant document is retrieved. The mean reciprocal rank (MRR) is the average of the reciprocal ranks over a set of queries.

5) Precision at N:

Since the total number of relevant documents for a query has a strong influence on precision at k, we calculate Precision values for subset N of retrieved documents at N=5, 10, 20, 30, 50, 70, 100. The

Average Precision at N is calculated over all the queries

6) MAP:

Arithmetic mean of average precision values can be calculated using the below formula:

$$\mathrm{MAP}(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} \mathrm{Precision}(R_{jk})$$

7) NDCG at N (Normalized Discounted Cumulative gain)

The gain is accumulated starting at the top of the ranking and may be reduced, or discounted, at lower ranks. The DCG is the total gain accumulated at a particular rank p.

$$DCG_p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2 i}$$

Where "reli" is the graded relevance level of the document retrieved at rank "i". Normalizing the actual DCG values by dividing by the ideal values gives us the normalized discounted cumulative gain (NDCG) values

$$NDCG_p = \frac{DCG_p}{IDCG_p}$$

Where IDCG is the ideal DCG value for that query.

Stem Class	Original Query	Expanded Queries
Avg. Precision across all queries	0.005006982668675	0.003070860252357
Avg. Recall across all queries	0.949564305047500	0.965308694787500
Avg. F1 score across all queries	0.009950998215771	0.006115577777811
Avg. MRR	0.950000000000000	0.900000000000000
MAP	0.410564625850340	0.168414965986394
Avg. Precision @5	0.48	0.22
Avg. Precision @10	0.46	0.19
Avg. Precision @20	0.465	0.16
Avg. Precision @30	0.436666667	0.153333333
Avg. Precision @50	0.394	0.16
Avg. Precision @70	0.344285714	0.148571429
Avg. Precision @100	0.294	0.147
Avg. NDCG @5	0.468246638	0.219661162
Avg. NDCG @10	0.458735546	0.200979257
Avg. NDCG @20	0.46168388	0.176872806
Avg. NDCG @30	0.444145083	0.169638993
Avg. NDCG @50	0.413363013	0.169840611
Avg. NDCG @70	0.384073489	0.165723487
Avg. NDCG @100	0.374551676	0.181194173

Fig 4.1 Stem Class Query Expansion Evaluation

DICE's	Original Query	Expanded Queries
Avg. Precision across all queries	0.005006982668675	0.001838437465187
Avg. Recall across all queries	0.949564305047500	0.999401197604800
Avg. F1 score across all queries	0.009950998215771	0.003667274130625
MRR	0.950000000000000	1.000000000000000
MAP	0.410564625850340	0.291360544217687
Avg. Precision @5	0.48	0.4
Avg. Precision @10	0.46	0.37
Avg. Precision @20	0.465	0.33
Avg. Precision @30	0.436666667	0.286666667
Avg. Precision @50	0.394	0.236
Avg. Precision @70	0.344285714	0.222857143
Avg. Precision @100	0.294	0.194
Avg. NDCG @5	0.468246638	0.454424704
Avg. NDCG @10	0.458735546	0.419863807
Avg. NDCG @20	0.46168388	0.378237501
Avg. NDCG @30	0.444145083	0.339751602
Avg. NDCG @50	0.413363013	0.290246971
Avg. NDCG @70	0.384073489	0.273271426
Avg. NDCG @100	0.374551676	0.267189394

Fig 4.2 DICE's Query Expansion Evaluation

MIM	Original Query	Expanded Queries
Avg. Precision across all queries	0.005006982668675	0.005006928721901
Avg. Recall across all queries	0.949564305047500	0.949564305047500
Avg. F1 score across all queries	0.009950998215771	0.009950892150505
Avg. MRR	0.950000000000000	0.954545454545454
MAP	0.410564625850340	0.087767470624613
Avg. Precision @5	0.48	0.12
Avg. Precision @10	0.46	0.1
Avg. Precision @20	0.465	0.07
Avg. Precision @30	0.436666667	0.07
Avg. Precision @50	0.394	0.066
Avg. Precision @70	0.344285714	0.072857143
Avg. Precision @100	0.294	0.086
Avg. NDCG @5	0.468246638	0.133699772
Avg. NDCG @10	0.458735546	0.1157886
Avg. NDCG @20	0.46168388	0.091325613
Avg. NDCG @30	0.444145083	0.086663245
Avg. NDCG @50	0.413363013	0.079438258
Avg. NDCG @70	0.384073489	0.087757153
Avg. NDCG @100	0.374551676	0.104343388

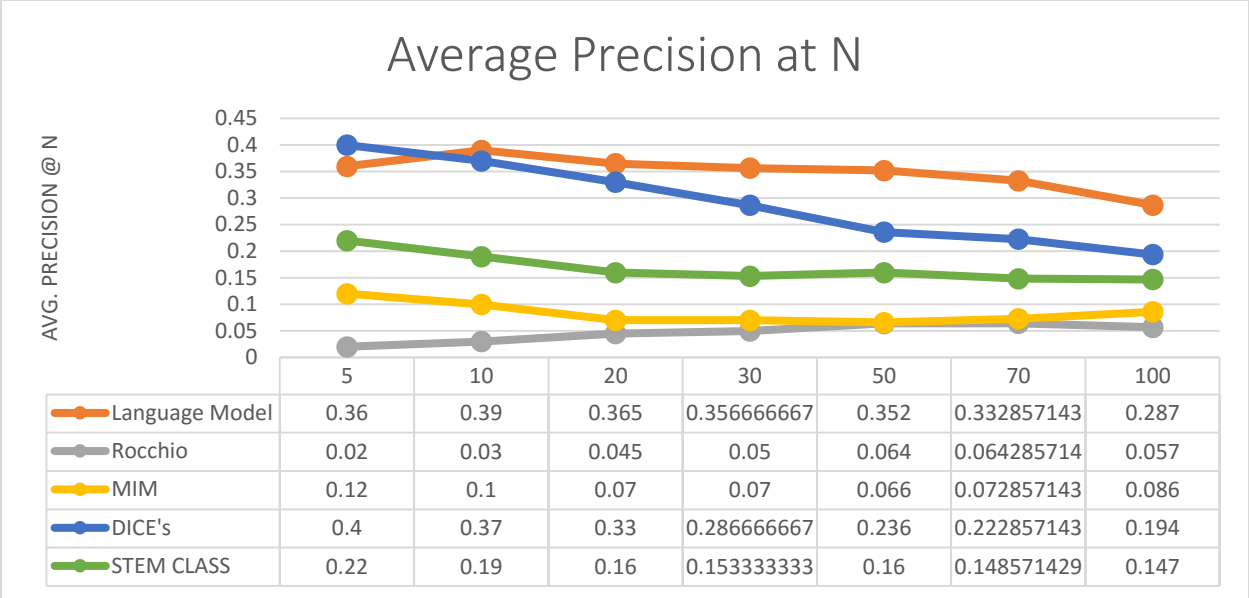
Fig 4.3 MIM Query Expansion Evaluation

ROCCHIO	Original Query	Expanded Queries
Avg. Precision across all queries	0.005006982668675	0.002243696400993
Avg. Recall across all queries	0.949564305047500	0.991409754109956
Avg. F1 score across all queries	0.009950998215771	0.004474580764109
Avg. MRR	0.950000000000000	1.000000000000000
MAP	0.410564625850340	0.047183673469388
Avg. Precision @5	0.48	0.02
Avg. Precision @10	0.46	0.03
Avg. Precision @20	0.465	0.045
Avg. Precision @30	0.436666667	0.05
Avg. Precision @50	0.394	0.064
Avg. Precision @70	0.344285714	0.064285714
Avg. Precision @100	0.294	0.057
Avg. NDCG @5	0.468246638	0.014038609
Avg. NDCG @10	0.458735546	0.022606985
Avg. NDCG @20	0.46168388	0.034911369
Avg. NDCG @30	0.444145083	0.040127443
Avg. NDCG @50	0.413363013	0.052395834
Avg. NDCG @70	0.384073489	0.054988174
Avg. NDCG @100	0.374551676	0.051843144

Fig 4.4 ROCCHIO Query Expansion Evaluation

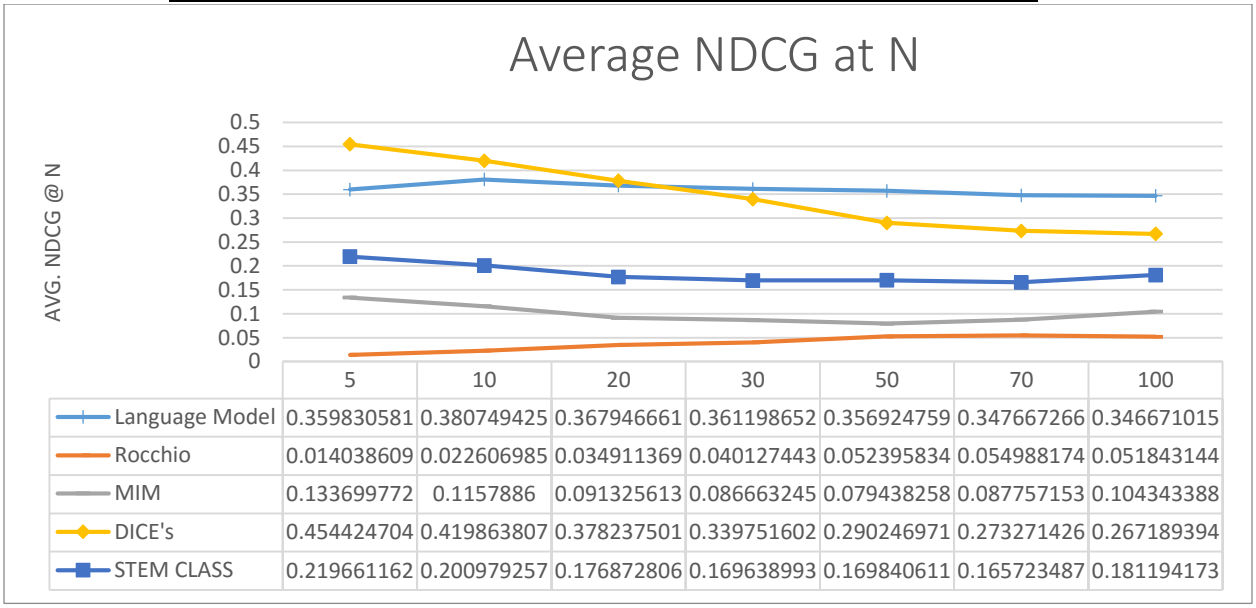
Language Model	Original Query	Expanded Queries
Avg. Precision across all queries	0.005006983	0.0020479
Avg. Recall across all queries	0.949564305	0.996925409
Avg. F1 score across all queries	0.009950998	0.004083953
MRR	0.95	1
MAP	0.410564626	0.34907483
Avg. Precision @5	0.48	0.36
Avg. Precision @10	0.46	0.39
Avg. Precision @20	0.465	0.365
Avg. Precision @30	0.436666667	0.356666667
Avg. Precision @50	0.394	0.352
Avg. Precision @70	0.344285714	0.332857143
Avg. Precision @100	0.294	0.287
Avg. NDCG @5	0.468246638	0.359830581
Avg. NDCG @10	0.458735546	0.380749425
Avg. NDCG @20	0.46168388	0.367946661
Avg. NDCG @30	0.444145083	0.361198652
Avg. NDCG @50	0.413363013	0.356924759
Avg. NDCG @70	0.384073489	0.347667266
Avg. NDCG @100	0.374551676	0.346671015

Fig 4.5 Language Model Query Expansion Evaluation



	5	10	20	30	50	70	100
Language Model	0.36	0.39	0.365	0.356666667	0.352	0.332857143	0.287
Rocchio	0.02	0.03	0.045	0.05	0.064	0.064285714	0.057
MIM	0.12	0.1	0.07	0.07	0.066	0.072857143	0.086
DICE's	0.4	0.37	0.33	0.286666667	0.236	0.222857143	0.194
STEM CLASS	0.22	0.19	0.16	0.153333333	0.16	0.148571429	0.147

Fig 4.6 Average Precision at N across Query Expansion techniques



	5	10	20	30	50	70	100
Language Model	0.359830581	0.380749425	0.367946661	0.361198652	0.356924759	0.347667266	0.346671015
Rocchio	0.014038609	0.022606985	0.034911369	0.040127443	0.052395834	0.054988174	0.051843144
MIM	0.133699772	0.1157886	0.091325613	0.086663245	0.079438258	0.087757153	0.104343388
DICE's	0.454424704	0.419863807	0.378237501	0.339751602	0.290246971	0.273271426	0.267189394
STEM CLASS	0.219661162	0.200979257	0.176872806	0.169638993	0.169840611	0.165723487	0.181194173

Fig 4.7 Average NDGC at N across Query Expansion techniques

5. SNIPPET GENERATION USING LUCENE

Once results are retrieved and returned for a given query, a good text summary (the snippet) created for each returning document will improve users' understanding of the document, possibly increase their click through ratio, and enhance their interaction with the search engine. Snippets are generated using Lucene in this project.
For snippet generation, we are using Lucene Highlighter class. Here, we store the documents' text in the index and set Index options using "Field.Store.YES" for below parameter in the field "ncontent" with the following parameter set to save the position and offset of all the terms:
FieldInfo.IndexOptions.DOCS_AND_FREQS_AND_POSITIONS_AND_OFFSETS

Using the Highlighter class, we create string fragments containing the best fragments matching the query on "ncontent" field in the index and taking the top 4 fragments to be displayed with the query terms highlighted as bold.

Lucene Snippet for query 85:
"Document discuss allegations measures taken corrupt public officials governmental jurisdiction worldwide"


Snippet Generated
for Query 85.txt

6. BIBLIOGRAPHY

- http://www.search-engines-book.com/
- http://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf
- https://wiki.apache.org/lucene-java/LuceneFAQ
- https://lucene.apache.org/core/4_7_2/highlighter/org/apache/lucene/search/highlight/Highlighter.html
- https://lucene.apache.org/core/4_7_2/highlighter/org/apache/lucene/search/highlight/Highlighter.html#field_summary