Project

Title: Query Expansion and Snippets Generation

Aim:

- Get you familiar with multiple query expansion techniques;
- Provide you hands-on experience on query expansion techniques implementation;
- Conduct efficient evaluation to measure and compare the performance before query expansion and after it, as well as the performance among different quern expansion techniques;
- Provide hands-on experience to generate snippets for retrieval results [Extra Credits].

Organization:

This is a team-work based project, so that you can either work individually or form a team of 2-3 people and work together. If you are working as team, you should specify in much detail how you share the responsibility and contribution. In terms of submitted programming codes, you should add notes on the source codes indicating which parts is written by which specific person.

You are expected to hand in the programming codes you used to fulfill the project tasks, a short report on analyzing the evaluation performance, and a short note on your contribution on the projects if you work as a team.

The full credit for this project is 120', where the query expansion part is 100', and snippets generation part is 20', even though it is optional, you are highly recommended to do.

The project will be posted on Marth 30th, 2016, and due on April 21st, 2016, 11.59pm.

Data Collection

We will work on the TREC provided AP89-collection for both tasks. You can download IR_data/AP89_DATA.zip by logging into http://www.ccs.neu.edu/home/vip/teach/IRcourse/html/index.html. Go to "data resources". One representative from each team, please contact me for the password.

Files in the AP89 corpus are in a standard format used by TREC, where each file contains multiple documents. The format is similar to XML, but standard XML and HTML parsers will not work correctly. Instead, read the file one line at a time with the following rules:

- Each document begins with a line containing <DOC> and ends with a line containing </DOC>
- The first several lines of a document's record contain various metadata. You should read the <DOCNO> field and use it as the ID of that document.
- The document contents are between lines containing <TEXT> and </TEXT>.
- All other file contents can be ignored.

You can also find the following files useful for your tasks:

- stoplist: you can use it to remove the stopping-words.
- stem-classes: list word variants for stem classes.
- query_desc.51-100.short: contains 25 queries
- qrels.adhoc.51-100.AP89: contains the relevance judgments for queries.

We choose to use this dataset because it has relevance judgments, and is larger than CACM data collection.

Data Processing

You can process the data either by yourself or by using Lucene.

Method1: by yourself.

If you choose to process the data by yourself, you will do the followings:

1) Parsing and tokenizing: get the content for each document between <TEXT>and</TEXT> (as described above), and parse the document to retrieve possible tokens by first removing the punctuation marks {“.”, “,”, “?”, “!”, “;”, “:”} at the end of the entire document or each sentence and then separating possible tokens by space. You can then apply the following rules to get tokens. It is the same as introduced in Assignment 1-B EC1 for tokenizing:

A token is a contiguous sequence of characters which matches the regular expression \w+(\.?\w+) – that is, any number of letters and numbers, possibly separated by single periods in the middle. For instance, bob and 376 and 98.6 and 192.160.0.1 are all tokens. 123,456 and aunt's are not tokens (each of these examples is two tokens). All alphabetic characters should be converted to lowercase during tokenization, so bob and Bob and BOB are all tokenized into bob.*

2) No stemming
3) Remove the stop-words in the provided *stoplist.*
4) Indexing: building an inverted index in the form of word→(docid1, tf1), (docid2, tf2), …, (docidN, tfN)
5) Remove stop-words for each query, and convert all terms in the query into lowercase.

Method 2: using Lucene

You can use Lucene to help you do tokenizing and building Index. It is all right that in Method1 and Method2, different tokenizing or stop-word list are used for processing. Please make sure that you know how to use Lucence API to get some useful information like the term frequency and document frequency of each term, and more. Please apply the tokenizing (as mentioned above) for the query, removing stop-words in the query by using the provided in *stoplist*, and convert all terms in the query into lowercase.

Task 1 Query Expansion

The problem of word mismatch is fundamental to information retrieval. Simply stated, it means that people often use different words to describe concepts in their queries than authors use to describe the same concepts in their documents. The severity of the problem tends to decrease as queries get longer, since there is more chance of some important words co-occurring in the query and the relevant documents. In many applications, however, the queries are very short. An obvious approach to solving this problem is query expansion. We will explore several query expansion techniques in this project.

A. Expand query by word variants in the same stem class

This resembles query-based stemming, which from some perspective is a kind of query expansion technique. In this method, you can expand the original queries by adding all the word variants in one same stem class for each term in the query.

In this method, you can choose to use any retrieval model you have created in previous assignments (Assignment 1-B Task 2. A/B/C, or Lucene-based TF-IDF/BM25) to get the ranking results for the original query and expanded query.

B. **Global analysis: expand query by checking word association/co-occurrence.**

We generate possible expansion words by analyzing the co-occurrence/association between words in the entire data collection. Word association can be measured under different metrics and in certain text window. In this project, **we fix the text window to be each individual document**, and would like to apply the following two metrics for word association measurement.

- *Dice's Coefficient (Dice)*
- *Mutual Information (MIM)*

Please retrieve the Top 15 most strongly associated terms for each term in the query to generate the expanded query.

Please use the same retrieval model you used for Task1.A to get the ranking results for the original query and expanded query.

C. 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.

C.1 Vector space model based pseudo relevance feedback: the Rocchio algorithm.

The Rocchio algorithm can be implemented as follows:

- 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. Please refer to Assignment 1-B Task 2.A on how to compute for TF and IDF score. Please make sure that when getting the TF score for term in the query, you follow the equation as:

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

  where |q| is the length of the query.
- Run your retrieval models (the same as in Task 1.A and Task 1.B) over the original query.
- Regard the Top N returned documents (we set N=50 here in the experiments) as relevant documents, and all others as non-relevant documents.
- 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, q_j is the jth entry in the vector representing q. q_j' is the updated entry for the jth entry in expanded query q_j'. *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. d_{ij} is the jth entry in vector for document d_i.

- Re-run your retrieval model to get the ranking results for expanded query.

Determined by the entry *tf.idf* score, please return the Top 50 terms to form the expanded query.

We set α=8, β=16, and γ=4 in this experiment.

C.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.
- 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).
- 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. Please apply Dirichlet smoothing for computing log(P(w|D)), which will be described later.

When using relevance language model for pseudo relevance feedback, you have to retrieve the information as term frequency, document frequency, collection-based term frequency, and more from your built Index or from Lucene API.

C.2.1 Query likelihood Model to estimate P(q|D)

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

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

q_i is every term in query q. To solve the zero probability problem, we can introduce the smoothing technique. We consider two smoothing techniques in this project.

C2.1.1 Jelinek-Mercer Smoothing

With Jelinek-Mercer smoothing, we have:

$$\log P(q_i \mid D) = \log((1-\lambda)\frac{f_{qi,D}}{|D|} + \lambda \frac{C_{qi}}{|C|})$$

where, c_{qi} 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 $\lambda = 0.5$ in this experiment.

C2.1.2 Dirichlet Smoothing

Under Dicirhlet smoothing, we can represent the query likelihood language model as:

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

and we set $\mu = 2000$ in this experiment.

C2.2 Estimate P(w|R)

In Step 3 and 4, we need to estimate P(w|R), where R represents the true distribution of the relevance model for query. P(w|R) can be computed as:

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

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

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

where, P(D) is regarded as uniform and can be ignored. *C* is the set of relevant documents. We do not apply smoothing here for P(*w*|D) and $P(q_i|D)$.

C2.3 Parameters to tune

Two parameters will have great impact on the query expansion retrieval results, i.e. the number of Top N documents to form the relevant set; and the Top M words with highest probabilities to expand the query. Please set N=10, and M=15 in this method.

D. Evaluation

Please consider the following measures for performance evaluation among multiple different methods. We will use the first 10 queries in file *query_desc.51-100.short* to do experiments in the project.

1) Averaged Precision across all queries
2) Averaged Recall across all queries
3) Averaged F_1 score across all queries
4) MRR
5) MAP
6) Averaged Precision@N (where N=5, 10, 20, 30, 50, 70, 100)
7) Averaged NDCG@N (where N=5, 10, 20, 30, 50, 70, 100).

Please evaluate the performance and make comparisons among the following methods:

1) Comparing retrieval performance for original queries vs. expanded queries using method A. Generate a comparison Table 1 for metrics 1)-5) in Task 1.D.
2) Comparing retrieval performance for original queries vs. expanded queries using method B with Top N (N=15) terms with two different term association measures. Generate a comparison Table 2 for metrics 1)-5).
3) Comparing retrieval performance for original queries vs. expanded queries using method C1 with Top N (N=50) terms. Generate a comparison Table 3 for metrics 1)-5).
4) Comparing retrieval performance for original queries (using JM-smoothing and Dirichlet smoothing separately) vs. expanded queries using method C2 with Top N terms (N=15). Generate a comparison Table 4 for metrics 1)-5)
5) Draw two plots comparing Precision@N and NDCG@N (N=5, 10, 20, 30, 50, 70, 100) after query expansion using the above four different methods. (You should have 6 curves in each plot, since Method B uses two term association measures; and in Method C2, we have two smoothing techniques).

Extra Credits: Task 2 Snippets Generation

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. We will generate our own snippets in this project.

A. **Luhn's approach**

Luhn's approach is to rank each sentence in a document using a significant factor and to select the top sentences for the summary. The significant factor for a sentence is calculated based on the occurrence of significant words. Please refer to Section 6.3.1 in the Bruce Croft's textbook for more details on how to compute the significant factor for each sentence in a document.

Please return the Top 2 sentences as the snippet for Top 5 documents for the first query (queryID=85, the original query) in file *query_desc.51-100.short.*

B. Using Lucene.

You can also use Lucence to generate the snippets. Highlighter class may need to be used.

What to hand in:

1.Your codes on implementing four query expansion techniques, and snippets generation.

2. A README file on how to run your codes.

3. A note on how each team member shares the contribution for the project.

4. You are expected to submit a short report (as *.docx or .pdf) for this project. The report should contain three sections.

Section 1: Introduction. Please briefly introduce what are the main goals in this project; what is query expansion; why do we need to do query expansion; What is snippet, and why generating good snippets is important.

Section 2: Methodology. Please introduce the different techniques for query expansion and snippets generation.

Section 3: Experimental Evaluation

- Section 3.1 Data collection analysis. Please provide the following statistics for the data collection: the total number of documents, the size of the collection vocabulary, the total number of word occurrences in the collection, the average document length, and the average query length (the first 10 queries).
- Section 3.2: Evaluation.

a) Please first output the original query and the expanded query you can generate using four different methods for the first query (queryID=85) in file *query_desc.51-100.short*.
b) The four comparison tables and two plots introduced in Task 1.D.
c) and the observations and conclusions you can achieve when analyzing the evaluation results.

- Section 3.3: Snippets generation. The generated snippets.