Your Name: Xiaoyu He

Your Andrew ID: xiaoyuh1

Homework 1

Collaboration and Originality

Your report must include answers to the following questions:

1. Did you receive help of any kind from anyone in developing your software for this assignment (Yes or No)? It is not necessary to describe discussions with the instructor or TAs.

   Yes, I asked Shiqing Jing whether we should use QryIop or QrySop for Near operator. I asked Yupin Huang should we output 100 line when we upload files to trec_val.

2. Did you give help of any kind to anyone in developing their software for this assignment (Yes or No)?

   Yes, I pointed to Yupin Huang that the docGetMatchResult of Near operator will not produce correct result, she has to be careful with the method overloading of QryIop operator which might lead to unexpected result.

3. Are you the author of every line of source code submitted for this assignment (Yes or No)? It is not necessary to mention software provided by the instructor.
   Yes

4. Are you the author of every word of your report (Yes or No)?
   Yes

Your Name: Xiaoyu He

Your Andrew ID: xiaoyuh1

Homework 1

Instructions

1 Structured query set

1.1 Summary of query structuring strategies

Briefly describe your strategies for creating structured queries. These should be general strategies, i.e., not specific to any particular query.

1. Add keywords or title or URL or body if there is only one word in the query
2. Add Near to consecutive words
3. Use AND if the word can appear separately in the document.
4. Put title or keywords in the most import words
5. Use URL field if the word in likely to appear in the URL

1.2 Structured queries

List your structured queries. For each query, provide a brief (1-2 sentences) discussion of:

1. ***which strategy (from Question 1.1) was used for that query,***
2. ***any important deviations from your default strategies, and***
3. ***your intent, i.e., why you thought that particular structure was a good choice.***

6:#OR(kcs.keywords kcs.title kcs.url kcs)

- I used strategy 1 since this query only contains single words. I only get results that are "highly related" to kcs, without getting the document that only have 'kcs' in the body. I think this can improve the accuracy.

71:#NEAR/2(living in india)

- I used stategy 2, I only get documents that have "living XX india" without getting documents that "living" and "india" are seperated, thus improve accuracy.

79:#OR(voyager.keywords voyager.title voyager.url)

- I used strategy 1 since this query only contains single words. I hope to only get results that are "highly related" to voyager.

84:#NEAR/2(continental plates)

- I used stategy 2, I only get documents that have “continental plates” without getting documents that “continental” and “plates” are separated.

88:#NEAR/1(forearm pain)

- I used stategy 2, I only get documents that have “forearm pain” without getting documents that “forearm” and “pain” are separated.

96:#OR(rice.title rice.keywords)

- I used strategy 1 since this query only contains single words. I only get results that are “highly related” to rice, without getting documents that just mentioned “rice” in the body.

104:#AND(indiana child support)

- I used strategy 3, because “indiana child support” should all appear in the document but not need to appear together. For example, “support child in indiana” is also acceptable. This can have higher recall rate compare to near.

182:#AND(#NEAR/1(quit smoking) #OR(smoking.title smoking.url somking.keywords))

- I used strategy 1, 2. I want to get documents that are highly related to smoking, so I used #OR(smoking.title smoking.url somking.keywords). I only want to get document that contains “quit somking” together, so I used NEAR operator.

194:#NEAR/2(designer dog breed)

- I used stategy 2, I only get documents that have “designer dog breed” or “designer style dog breed”without getting documents that “designer dog breed” are separated.

196:#OR(#NEAR/1(sore throat) #NEAR/2(throat sore))

- I used stategy 2. Using NEAR, I only get documents that have “sore throat” or “sore throat”.

2 Experimental results

Present the complete set of experimental results. Include the precision and running time results described above. Present these in a tabular form (see below) so that it is easy to compare the results for each algorithm.

2.1 Unranked Boolean

	BOW #OR	**BOW #AND**	**Structured**
P@10	0.0000	0.0100	0.3400

P@20	0.0050	0.0250	0.3000
P@30	0.0100	0.0433	0.2367
MAP	0.0004	0.0033	0.0773
Running Time	9311.922 ms	3036.388 ms	3738.377 ms

2.2 Ranked Boolean

	BOW #OR	**BOW #AND**	**Structured**
P@10	0.2000	0.4300	0.36
P@20	0.2650	0.4800	0.35
P@30	0.2833	0.4833	0.3667
MAP	0.1015	0.2439	0.1205
Running Time	8206.997 ms	3113.430 ms	3566.685 ms

3 Analysis of results: Queries and ranking algorithms

For unranked Boolean result, the structured queries have much higher result than #OR and #AND. The retrieval performance of #OR is bad because the result of #OR contains too much unrelated documents. #AND also retrieves a lot of irrelevant documents. #AND will retrieve documents that contains "a lot of people from India wants to lives in US", however this is not relevant to "living in India". In the structure queried, I used lots of NEAR operator and putted more constrains, the precision of retrieved document is much more accurate.

One interesting observation is that from P@10 to P@30, the precision in unranked Boolean fluctuates, but in ranked Boolean, the precision is almost continuously increasing. I guess that is because 30 is still a small number, using p@50 will produce more reliable and stable result, as suggested on the class.

Another interesting observation is that ranked Boolean #AND achieved highest MAP and best precision, even higher than structured query. For now, I don't quit understand why.

The performance of ranked Boolean is much higher than unranked in #OR and #AND. Generally speaking, if a query term appeared many many times in a document, they are much more likely to be relevant. Taken this into account, the ranked Boolean have much better result than unranked Boolean.

However, in structured queried, ranked Boolean and unranked Boolean performs similar, I think that is because unranked or ranked doesn't make much much difference because structured queries are already doing the best it can to achieve highest accuracy.

The running time of OR query is much longer than AND and structured query. The reason is that OR will return too many result, which slows down the system. My structured queries contain a lot of #NEAR, so on the one hand, the result returned by structured queries should be much less than #AND, this should shorten the running. But on the other hand, #NEAR is much more complicated and time consuming than

#AND, this will make the time longer time. Thus, balancing this two factor, the running time of structured query and #AND do not have a significant difference.

4 Analysis of results: Query operators and fields

In unranked Boolean model, structured query using #NEAR operator improves the precision a lot. Thus #NEAR and fields is very powerful and helpful in unranked Boolean.

However, it's not the same in ranked Boolean model. It's a bit of upsetting to find that my sophisticated structured query doesn't even perform as good as simple #AND operator. The performance of structured query greatly depends on my own "subjective" judgment, that some terms should near each other.

In ranked Boolean model, my "subjective" structured queries don't perform as good as "objective" #AND queries. So, instead of using my "subjective" belief to construct structured queries, it would be even better to use "object" #AND. Maybe a person who is more experience in constructing structured queries than me can produce better result, but this is still hard.

Finally, I think the effectiveness of field is limited, and it depends on the context. For example, some document may have missing title and keywords. In the structured queries that I designed, I heavily relies on the assumption that title, keyword, URL are not missing. If my assumption is wrong, my query result quality can be damaged.