Your Name: Jiangtian Qian

Your Andrew ID: jiangtiq

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.
   No.

   If you answered Yes, provide the name(s) of anyone who provided help, and describe the type of help that you received.

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

   If you answered Yes, provide the name(s) of anyone that you helped, and describe the type of help that you provided.

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.

   If you answered No:
   a. identify the software that you did not write,
   b. explain where it came from, and
   c. explain why you used it.

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

   If you answered No:
   a. identify the text that you did not write,
   b. explain where it came from, and
   c. explain why you used it.

Your Name: Jiangtian Qian

Your Andrew ID: jiangtiq

Homework 1

Instructions

1 Structured query set

1.1 Summary of query structuring strategies

1. For professional term, name, location name, phrase and so on, use NEAR/1 to combine them together.

2. Keep the mandatory term, and for the similar adjective which define the nouns, and select one of them.

3. Divide the long query to several two-pair query, like biword method.

4. For term which may have some relationship, use NEAR/n to limit their distance.

5. For hot topic, just use the only nouns to query.

6. Use field, like url, keywords, title to find the match documents.

1.2 Structured queries

708:#OR(#AND(Decorative slate sources) #AND(slate sources) #AND(Decorative slate) #AND(slate.url sources.url)) 1.Strategy3,6; 2. Divided the three terms query to (slate source) and (decorative slate), not include (decorative slate), because (decorative slate) have little relationship with the query. 3. The recall for this decorative is low, so I add OR operator to increase it. It is can return right documents when just search for slate sources and Decorative slate instead of Decorative slate sources. By doing this, we can get more related documents back.
710:#AND(#OR(#NEAR/1(Prostate cancer) #NEAR/2(cancer Prostate) #NEAR/1(Prostate.title cancer.title) #NEAR/1(Prostate.keywords cancer.keywords)) treatments) 1. Stratgy 1, 3,4,6 2. The term is given as Prostate cancer treatments, I use NEAR/2 and change the order of the term in case of the different combination of the phrase. 3. The prostate cancer is a professional term, so I combine them together, and try use different field to search it. The precision is high enough, we reduce it a little bit by making “treatments” a choice not a mandatory to get higher recall.
721:#OR(#AND(Census application) #AND(Census data) #AND(Census data application)) 1. Strategy 2, 3 2. Nothing else 3. Census is a professional term, the aim seems like to find the application of Census, so Census is mandatory, keep it and combine it with other words. By doing this, we can increase the recall and reduce little of the precision.

```
726:#AND(#OR(#NEAR/3(Hubble telescope) #NEAR/200(telescope Hubble) #NEAR/200(telescope.title Hubble.title)) repairs)
```

1. Strategy 1,4,6
2. Invert the order of the term when use strategy 4.
3. Limit the distance of telescope and Hubble will help to increase the precision and add field to query terms wil help return more related documents, which helps increase the recall.

```
728:#OR(#NEAR/1(save endangered whales) #NEAR/1(endangered whales) #AND(whales save) #AND(whales endangered) #AND(whales.title endangered.title) whales.url whales.title)
```

1. Strategy 1, 3, 4, 5, 6
2. Invert the order of the term when use strategy 4.
3. Strategy 1 and 3 helps increase the precision and strategy 5,6 helps increase the recall. The MAP is better than before, because it will has more related documents in the related top n documents.

```
730:#OR(#NEAR/200(#NEAR/1(Gastric bypass) complications) #NEAR/200(complications #NEAR/1(Gastric bypass)))
```

1. Strategy 1, 3, 4
2. Nothing else
3. Because Gastric is a professional term, so is the bypass. I combine them together to increase the precision. And add the limit, like NEAR/200, here will increase the precision.

```
733:#OR(#NEAR/10(Airline.title overbooking.title) #NEAR/10(Airline.url overbooking.url) NEAR/200(Airline.keywords overbooking.keywords)  #AND(Airline overbooking))
```

1. Strategy 4, 6
2. Nothing else
3. Add field to help get more document back, which helps increase the recall.

```
742:#AND(#NEAR/1(hedge funds) fraud protection)
```

1. Strategy 1
2. Nothing else
3. Hedge funds is a professional term, which can be treated as a phrase. So use NEAR/1 will help increase the precision and decrease little of recall for top n documents.

```
749:#OR(#NEAR/1(Puerto Rico) #NEAR/1000(#NEAR/1(Puerto Rico) state) #NEAR/1(Puerto.title Rico.title))
```

1. Strategy 1,2,4
2. Nothing else

3 Puerto Rico is a location name, which can be treated as a phrase. So use NEAR/1 will help increase the precision and decrease little of recall for top n documents. Add field to increase the recall.

```
751:#OR(#AND(Scrabble Players) #NEAR/1(Scrabble.title Players.title) #NEAR/1(Scrabble.keywords Players.keywords))
```

1. Strategy 1,4
2. Nothing else

3 Scrabble Players can be treat as a professional term or not, so we deal with it in two situations. Firstly, NEAR/1 will help increase the precision if it is a phrase or use AND if it is not. This increase the precision and decrease little of recall for top n documents. Add field to increase the recall.

2 Experimental results

2.1 Unranked Boolean

	BOW #OR	BOW #AND	Structured
P@10	0.0000	0.1800	0.2000
P@20	0.0050	0.1800	0.1950

P@30	0.0100	0.2000	0.2100
MAP	0.0002	0.0351	0.0483
Running Time	12356.403	3327.666	5212.748

2.2 Ranked Boolean

	BOW #OR	**BOW #AND**	**Structured**
P@10	0.0900	0.2500	0.3500
P@20	0.1300	0.3200	0.4300
P@30	0.1267	0.3267	0.4100
MAP	0.0104	0.0715	0.1067
Running Time	12175.560	3427.666	5347.068

3 Analysis of results: Queries and ranking algorithms

Why we get different results under different combination of operator?
Because the information need is ambiguous for search engine. AND operator is exact-match, so when we include all the query terms and use the #AND operator. We will have all the related documents which include all the terms return back. At this time, we will have a high precision. As we can see above, the top n documents' precision of BOW#AND is really high and much larger than #OR at the same level. What about the recall? Obviously, #AND operator will return less documents than the #OR operator, since #AND operator just return documents contains all the terms but #OR return all the documents which contains at least one of the term. So the BOW#AND's recall is smaller than #OR. For MAP, we can see from the table above, BOW #AND has higher MAP than BOW #OR. This makes sense, because documents returned by BOW #OR may include unrelated document, which will reduce the MAP. But document returned by BOW #AND is much more relevant than BOW #OR, so it is reasonable to have higher MAP than BOW #OR.

But why BOW #AND is not the highest in precision and MAP?
Because we have Structured query here. The structured query is combined with #AND, #OR, #NEAR and query terms. Since there is hybrid of these three operators, there must have a tradeoff of precision and recall. Balanced these two tradeoff and get a good MAP here.
Though we #AND operator include the all the query terms, it doesn't mean it will return the matched documents. In other word, query terms combination is not equals to information need. For example, enter #AND(search engine course), we want to search the related search engine course, but may get some documents about the plating course just because its content have these three words, like "we have **search** a lot in the internet and find the machine **engine** which is… and learn the **course**…". At this time time, we treat search engine course as a whole will increase the precision largely, this is strategy 1 I mentioned above-use NEAR/1 for the phrase. Also, I use strategy 3, which helps increase the precision by divided long query to two or three parts. This can be used to deal with hybrid of phrase query and normal query. This is the same idea as **biword** method. By doing this, I got a higher precision and recall than BOW #AND and BOW #OR. And there is another problem. When we use #AND operator, it will include all the

terms to do the query, but there my be other description about the terms. For example, #AND(Decorative slate source), it just return documents contain these three term. But what if there is a document which content is related to this query without "Decorative". So, I keep the mandatory terms and combine it with other defined terms. For this case, we can get document which content is blue slate source without "Decorative" return back. This strategy can help increase the recall with little decrease in the precision. Also, we can use #NEAR/N operator to limit the distance between two related terms. If these two terms have relationship, it can not be far away from each other in the content. At last, we add field to help return more related documents and fast the program processing. The more correct document we get at top N levels, the better MAP we will have. As we can see, by using these strategy, we get a much better performance than just use #AND.

For the running time, #AND operator has the smallest running time, because it has less match documents and #OR has the highest for it has the largest matched documents. The Structured is in the middle.

For different Boolean retrieval models, we can see that unranked Boolean retrieval model's precision is much smaller than the ranked Boolean retrieval. For BOW #OR, it is reasonable because the unranked Boolean retrieval model may return unordered mix of unrelated documents and related documents back. We have huge possibility to have unrelated documents at the top level. For BOW #AND, it is the same too. But the difference between ranked Boolean retrieval is smaller than BOW #OR, because #AND operator is more likely to return much more related documents back and the possibility to have related documents in top level is larger than #OR. For structured query, it is the same reason to have ranked Boolean retrieval to performance better than unranked Boolean retrieval model. As we can see, ranked Boolean works very well when the searcher knows what he wants. The run time is similar to the unranked Boolean retrieval model; it seems like ranked Boolean retrieval is very efficient.

4 Analysis of results: Query operators and fields

The #AND operator can achieve a good performance at the most time. #AND operator will retrieval the documents contains all the terms, which ensure the high precision of the returned back results. It can help return a high precision results which scarified a little recall percentage. For it will return less documents than any other operators so it runs quick in the query. But the #AND operator is not the perfect one, which can still return unrelated documents back. We can use the hybrid operator to optimize query.

The #OR can help retrieval more related documents back than any other operators. Because it will return the documents back which contains at least one of the terms. It can help return a high recall results scarified a little precision percentage. But it more likely to return documents which doesn't match the information need. And it is very slow in the query.

The #NEAR/n operator can help increase the precision of phrase query and limit the distance of related terms as we mentioned three examples above. But it not that flexible to use, for example, can not inverted terms order when do the NEAR query.

The title, url and keywords field helps increase the number of document retrieval back(recall) but a little. The title is the most helpful when used with the near operator.

I think the #OR operator is little slow than I though. At first, I expect it run faster than #AND operator.