Your Name: Sz-Rung Shiang

Your Andrew ID: sshiang

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

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

3. Are you the author of every line of source code submitted for this assignment (Yes or No)?
   Yes. (I took the same course last semester but I dropped in the end. I used the same code.)

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

Your Name: Sz-Rung Shiang

Your Andrew ID: sshiang

Homework 1

1 Structured query set

1.1 Summary of query structuring strategies

The following are some rules to generate my structured queries:

1. If the query contains the phrase, #NEAR/1 operator is used to make sure these words appear together.
2. If the query contains the phrase that may be frequently used but there may be some adjectives or adverb between it, then #NEAR/3 is used.
3. If there is only a single term in the query, then my system will search for this term in keywords, title or body part.
4. If the query is a single term and the query is a general term, then the system will search in keywords or title only.
5. If the query contains a set of terms but they are not necessary to be a phrase, then the system will use AND for each terms which is not a stop word.
6. If the query contains an important word, then the term has to appear in the title.

1.2 Structured queries

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

Strategy (3): *kcs* is a single term and it's with ambiguity; therefore I use strategy (1) to make the documents with this term in keywords/title/body as relevant documents.

71:#AND(#AND(living in india) india.title)

Strategy (5)(6): *living in india* is the query with multiple terms and it's not a phrase; therefore docuemnts with these terms in body may be the relevant documents. In addition, *india* is the most important concept in the query and it might appear in the body but it is not relevant to the article (for example, the website with choose language or location); as a result, I make a constraint that *india* must be in the title.

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

Strategy (4): *voyager* is the single and general term in the query; therefore, document with this term in keywords and title may be the relevant term.

84:#NEAR/1(continental plates)

Strategy (1): *continental plates* is a phrase and it's less likely that there are other terms between them; therefore, I used strategy (1) to make it as hard constraint as a phrase in body.

88:#AND(forearm pain)

Strategy (5): the two terms *forearm pain* is not necessary as a phrase and it's less likely to appear in the same document but the document is not relevant to the query.

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

Strategy (4): *Rice* is a general term and therefore the system only consider the documents with this term in title and keywords as relevant documents. If the documents with rice in the body, it's possible that *rice* is not the main topic in the documents.

104:#AND(indiana #NEAR/1(child support))

Strategy (1)(5): *Child support* is a phrase to represent the specific meaning and *Indiana* and *child support* is not necessary to be a phrase. As a result, strategy (1) is used for *child support* and strategy (5) is used for *Indiana* and *child support*.

182:#NEAR3(quit smoking)

Strategy (2): *Quit smoking* is a phrase and there might be terms between it; therefore I used strategy (2).

194:#NEAR/3(designer #NEAR/1(dog breeds))

Strategy (1)(2): *Dog breeds* is a phrase to represent a specific meaning and *designer dog breeds* is a phrase and there might be terms between it.

196:#NEAR/1(sore throat)

Strategy (1): *Sore throat* is the phrase and therefore I used the strategy (1) to form up the query.

2 Experimental results

2.1 Unranked Boolean

	BOW #OR	**BOW #AND**	**Structured**
P@10	0.0000	0.0100	0.1400
P@20	0.0050	0.0250	0.1800
P@30	0.0100	0.0433	0.1733
MAP	0.0004	0.0033	0.0398
Running Time	6810.72ms	2288.26ms	2461.05ms

2.2 Ranked Boolean

	BOW #OR	**BOW #AND**	**Structured**
P@10	0.2000	0.4300	0.3800
P@20	0.2650	0.4800	0.3950
P@30	0.2833	0.4833	0.4133
MAP	0.1015	0.2439	0.1601
Running Time	7068.05ms	2346.08ms	2492.83ms

3 Analysis of results: Queries and ranking algorithms

In ranked case I used term frequency as scores while in unranked case I used 0/1 scores no matter how many times they are matched. In section 2-1, #OR operator only achieves 0.01 in P@30 and 0.0004 in MAP, #AND operator achieves 0.0433 in P@30 and 0.0033 in MAP, while structure query performs significantly better, which is 0.1733 in P@30 and 0.0398 in MAP. It's because the number of retrieved documents is large but the relevant ones are hard to be distinguished from irrelevant ones because the order of retrieved documents is by sorting the document ID, which is not related to any relevance judgment. For ranked case, all the queries perform better than those in unranked case. It's because the more the matching of terms (more terms are matched), the more relevant the documents should be (Although it might not always be true.).

In addition, #AND queries outperform structured queries in both Precision and MAP in ranked case. The reason might be that structured queries are more strict because #NEAR operators are used; however, the document set is small and thus for some queries the retrieved documents are less than 100, which will deteriorate the recall rate and even the precision when P is large. When k (P@k) is 5,10,20 the difference in performance is 2%, 5% and 9.5% respectively. It shows that the structured query has the ability to distinguish relevant documents and rank them higher; however, it doesn't have the ability to retrieve a lot of relevant documents because it filters out too many documents.

The following is the discussion for running time. In section 2-1 and 2-2, the running time of #OR operator spends roughly 3 times than other operators do. It's because doing the #OR (union), we increase the numbers of documents in the inverted list. In addition, the results show that there is no obvious difference between ranked and unranked cases. The only part that will increase the time of ranked one is that we can to call getTf() function to get the term frequency.

4 Analysis of results: Query operators and fields

Operator #OR, #AND, #NEAR/1 and #Near/3 operators are used in the system. Among them, #NEAR/1 is the most rigorous operator that it only allows the case that terms are consecutive, and the overall ranking of the most strict of operator is #NEAR/1 > #NEAR/3 > #AND > #OR. However, the results of precision and MAP evaluations are not in this fashion. The results of structured query operation is even lower than #AND operation. The reason is that there is too few documents are retrieved if we use this strict operation, therefore reducing the number of retrieved documents at the same time. #NEAR/3 operator is a good balance between strict query and enough number of retrieved documents. For operation #OR, there are too many documents matching the pattern, so the distinction of relevance is not obvious. For the document with "search engine", the one with "search" and "engine" in it, the one with only "engine", they all get the same scores (given the term frequency is the same.) in unranked case and the score of the count of matching terms in ranked case. That is to say, for document A with "search" appearing 20 times and "engine" appearing 0 times and document B with both "search" and "engine" appearing 5 times, A can get higher score, which doesn't make sense in this query case. Overall, the stricter the query operator is, the more relevant but less documents may be retrieved. The performance of relevance may also depends on how many relevant documents in the set and the parameter in the evaluation metrics.

Different fields may have different advantages according to the queries. Field *body* has most of the content and therefore much more terms in it. Using field *body* will increase the number of retrieved document, and most of the time if the queries match the terms in the *body* field, the documents should be relevant. On the other hands, fields *title* and *keywords* have less content in it but the terms in it are all important and relevant to the topic of the documents. There is tradeoff using these fields – relevance and recall: if the queries terms are very unique and the document frequency is low, then using *body* fields may be the best choice, retrieving enough number of documents and they are relevant; while if the queries are stop words or words with high frequency, then *title* or *keywords* fields may be better because there might be too many retrieved documents using *body* field and thus the precision and MAP will be low. The weakness is that there might be less documents to be retrieved and then recall may be low. Take our queries for example, query "rice" (common term) gets 0.0218 mAP using *body* field and 0.0541 mAP using *keywords* and *title* fields; while query "kcs" (less frequency term) gets 0.2892 in mAP using *body* field and 0.1368 in mAP using *keywords* and *title* fields. Other fields *url* and *inlink* have the same intuition as *title* and *keywords*. In conclusion, if the query terms are distinguishable between relevant and irrelevant documents set, using body field is enough and effective, while if the query terms are not distinguishable (ambiguity or common words), then using strict fields such as keywords or title is more effective.