**Your Name: Jiachen Li**

**Your Andrew ID: jiachenl**

# Homework 1

## 1 Statement of Assurance

I certify that all of the materials I submit are original works that were done by myself.

## 2 Structured query set

### 2.1 Summary of query structuring strategies

To create a good structured query, I think the key point is to identify the information need behind the query terms. Based on that, my strategies can be summarized as follows:

**Strategy 1:** I'd like to first check whether the terms in original query are adequate for a good result. If not, I may add some other term based on my experience. Of course, I may also revise the term that is not suitable in the query.

**Strategy 2:** I'll then evaluate the relationship between each term. To be specific, I want to find whether there are some terms that are usually used together for special meanings or as fixed expressions. If so, I could apply NEAR/n operator for them.

**Strategy 3:** To make searching more efficient, I will try to get an idea about where each term is more likely to appear. Thus, I could specify the searching for each term in a right field.

**Strategy 4:** At last, based on the overall relationship between each part of the terms, I'll make a final decision about which operator (i.e., AND or OR) should be used to combine the terms for a better result

### 2.2 Structured queries

My structured queries as well as their descriptions are listed as follows.

**Query 10:#AND(cheap.title internet.keywords)**
Strategy 3 and 4 are used since I believe cheap is more likely to appear in title to attract eyes and internet, which indicates the content of the document, should be a keyword. The results should be combined with AND to obtain the document that contains both terms.

**Query 12:#AND(djs.url djs.title)**
Strategy 1, 3 and 4 are used. The original query contains only one term, which I think is not adequate at all. Since I don't have an idea about which term to add, so I simply duplicate this term but let them in different fields, then the result is combined with AND for higher precision.

**Query 26:#AND(#NEAR/2(lower heart rate))**
Strategy 2 is used. These three terms together have specific meaning, so just search them within NEAR.

**Query 29:#AND(#OR(#NEAR/2(ps 2) ps2.url) game)**
All the strategies are used. First, since "ps 2" stands for Sony's PlayStation, I apply NEAR operator. Besides, I think "ps2" could also lead to similar result, so I add this term, specify it in "url" field and combine the results of these two with OR. After that, I filter the result by applying AND operator with "game".

**Query 33:#AND(#NEAR/2(elliptical trainer))**
Strategy 2 is used. These two terms together have specific meaning.

**Query 52:#OR(avp.url avp.title)**
Strategy 1, 3 and 4 are used. This query also contains only one term at first, so I take the same trick as in query 12. However, here I combine the terms by OR since it give me better result. (I suppose the reason is that there isn't much overlap in the inverted list produced by these two fields of avp.)

**Query 71:#AND(living india.keywords)**
Strategy 3 and 4 are used. India may be contained in lots of article, so I limit the searching area in the keywords field.

**Query 102:#AND(#NEAR/2(fickle creek farm))**
Strategy 2 is used. I think these 3 words stand for a particular farm (not quite sure), so I pick NEAR.

**Query 149: #AND(uplift #OR(#NEAR/3(yellowstone national park) #NEAR/3(yellowstone park)))**
Strategy 1, 2 and 4 are used. Similar as Query 29, since "yellowstone national park" has specific meaning, and "yellowstone park" may have the same meaning, so I apply NEAR for each of them and then combine the result by OR. Finally, I use AND operator to find results also including "uplift".

**Query 190:#AND(#NEAR/2(brooks.title brothers.title) clearance)**
Strategy 2, 3 and 4 are used. "Brooks brothers" should stand for a brand, and this kind of information is more likely to appear in the title, so I use the NEAR operator for title field. Besides, I used AND to filter the result from "brooks brothers" that contains the "clearance".

# 3 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.

## 3.1 Unranked Boolean

| | BOW #OR | BOW #AND | Structured |
|---|---|---|---|
| **P@10** | 0.0100 | 0.0400 | 0.2700 |
| **P@20** | 0.0050 | 0.0200 | 0.2650 |
| **P@30** | 0.0033 | 0.0433 | 0.2567 |
| **MAP** | 0.0010 | 0.0142 | 0.1220 |
| **Running Time** | 00:08 | 00:01 | 00:01 |

## 3.2 Ranked Boolean

| | BOW #OR | BOW #AND | Structured |
|---|---|---|---|
| **P@10** | 0.1500 | 0.2500 | 0.4100 |
| **P@20** | 0.1800 | 0.2600 | 0.4300 |
| **P@30** | 0.1667 | 0.2767 | 0.4100 |
| **MAP** | 0.0566 | 0.0980 | 0.1823 |
| **Running Time** | 00:09 | 00:01 | 00:01 |

# 4 Analysis of results

### BOW #OR vs BOW #AND vs Structured

Generally speaking, the #OR and #AND approaches, which focus on high recall and high precision respectively, are both quite simple strategies to form queries. They do not take the relationship between each term into consideration, and they also leave the other information such as the fields behind. Compared with them, the structured way is a bit more sophisticated. By bring my own knowledge and experiences, the terms can be formed in a way that better represents the information need, so the query process can be much more efficient thus improving the result. Of course, in the meantime, the structured query will look more complicated. According to the experimental results on precision, the #AND approach outperforms the #OR approach while the structured approach outperforms both of them, which matches our intuition and discussion above. For the running time, there is no doubt that the #OR approach will take much longer than #AND, since it has to combine the score list of each term. However, though in my experiment the structured approach runs as fast as #AND, I do not think that the structured approach must be faster than #OR, because it really depends on how complicated the structured queries are.

### Ranked vs Unranked

It can be seen from the results that the ranked model outperforms unranked model for all query forming strategies. This is not surprising since the ranked model use the information of term frequency as the score to help rank the result, which makes sense because the document with higher term frequency has more possibility to be relevant. For the running time, the ranked model should be longer as, on the one hand, it calculates the score information during the retrieval. On the other hand, it need to sort the results by score after the retrieval.

### Query Operators and Fields

In this assignment, three query operators are studied and implemented. Among them, I think the #AND operator is the most wildly used one as it can filter the results to satisfy the information need within every single term, which in most cases is what we want. Besides, the #AND operator should also be the most efficient one, as it just need to loop over the shortest score list of its terms. However, the #AND operator doesn't take the relationship of the terms and it sometimes may lead to a bad result due to its focus on precision. At this time, in my point of view, the #OR and #NEAR operators will come to rescue.

The #NEAR operator can be used to deal with adjacent terms that have specific meaning. For example, in query 29, term "ps" and "2" together point to Sony's 2nd generation PlayStation. If we simply search with #AND(ps 2), as you can imagine, the precision could be very low since the meaning for single "ps" is ambiguous. And there could also be quite a lot documents contain the term "2", so many irrelevant documents will be retrieved. On the contrary, if we use #NEAR/2(ps 2), the meaning would be much clear, so as the retrieval result. The #NEAR operator should be added very carefully. If you make

mistakes about the position relationship between every adjacent terms, the search engine will miss many relevant results. For the running time, in my opinion, applying #NEAR operator is similar as applying #AND for each term's score list and the relevant document's position list one after another. Therefore, the running time for #NEAR operator should be longer than #AND, and will vary according to the length of position list.

The #OR operator can be used when the recall of the result is low. In the same example as shown above, though searching with #NEAR/2(ps 2) can have pretty good precision, only one relevant document is retrieved, i.e. the recall is very low. To improve the performance, I added another term "ps2.url" and use #OR to combine them like #OR(#NEAR/2(ps 2) ps2.url), since "ps 2" and "ps2" have exactly the same meaning here, the #OR combines their score list and gets a better result (i.e., 4 relevant documents are retrieved).

The fields can tell search engine where to retrieve the information from, thus it can make searching more efficient. However, the biggest challenge is probably the usage of fields really depends on your own judgment about where the term is more likely to appear. Actually, sometimes you have to do experiments to find out which field works better. A perfect use of field might improve both precision and recall while a bad usage could ruin your result immediately, for example, once I specified a wrong field to a term, it turned out the MAP dropped directly to 0.