Your Name: Mengyu Yang

Your Andrew ID: mengyuy

Homework 2

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)? It is not necessary to mention software provided by the instructor.

   No. I used the framework provided in course website (QryEval) and implemented different models by myself.

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

   Yes.

Your Name: Mengyu Yang

Your Andrew ID: mengyuy

Homework 2

1. Experiment 1: Baselines

	Ranked Boolean	BM25 BOW	Indri BOW
P@10	0.1500	0.3000	0.2300
P@20	0.1800	0.2950	0.2800
P@30	0.1667	0.2967	0.2900
MAP	0.0566	0.1304	0.1277

2. Experiment 2: Queries with Synonyms and Phrases

2.1. Queries

10:#NEAR/2(#SYN(cheap inexpensive) #SYN(internet networks network))
12:#SYN(djs dj DiscJockey DiscJockeys)
26:#NEAR/2(#SYN(lower decrease low) #NEAR/1(heart #SYN(rate rates)))
29:#NEAR/1(#SYN(ps playstation) 2 games)
33:#NEAR/1(#SYN(elliptical ellipticals) #SYN(trainer trainers machine machines))
52:#SYN(avp #NEAR/1(association of volleyball professionals))
71:#NEAR/1(#SYN(living live life) in india)
102:#NEAR/1(fickle creek farm)
149:#SYN(uplift earthquake) at yellowstone #NEAR/1(national park)
190:#NEAR/1(brooks brothers) #SYN(clearance sale)

2.2. Query descriptions

For each query, provide a brief (1-2 sentences) description that identifies which strategy was used for that query, any important deviations from your default strategies, and your intent, i.e., why you thought that particular structure was a good choice.

My query set is structured based on the following strategies:

a. Use #SYN operator when a query term is satisfied one or more than one conditions as follows: **1**. If the term is an abbreviation, the full word will be added as a parameter of #SYN operator. **2.** If the

term has synonyms that have very similar meanings, add the synonyms to SYN operator. **3.** If the term has replacements that are more commonly used or with more specific meaning, then the replacements will be added as parameters of #SYN operator. **4.** If a word has different forms such as "do doing done did" or "machine machines", add the forms that make sense in the query to #SYN operator. This strategy makes the search result more intelligent since the search engine not only retrieves document based on term but also based on the term of similar meanings.

b. Use #NEAR operator when several terms has a specific meaning as a phrase, so that search engine matches those terms as a whole and the retrieval documents are more likely to match the meaning of the query.

10:#NEAR/2(#SYN(cheap inexpensive) #SYN(internet networks network))
This query uses the two strategies mentioned above. Cheap and inexpensive has the same meaning, internet is also referred as network(s) in some situation, so #SYN is used to support the synonyms. Apart from that, I used #NEAR operator to make sure that cheap is very closed to internet, which increase the probability that the meaning of this phrase is retained.

12:#SYN(djs dj DiscJockey DiscJockeys)
This query utilized the first strategy mentioned above. Since dj is an abbreviation, I added its full word Disc Jockey and their plural forms in order to retrieve documents contains this terms and its equivalent terms.

26:#NEAR/2(#SYN(lower decrease low) #NEAR/1(heart #SYN(rate rates)))
Two strategies are applied in this query. "lower" "decrease" "low" have similar meaning. "rates" is the plural form of rate". "heart rate(s)" is a common phrase that should be appear together to express a meaning, and "lower" should be very closed to "heart rate" so that the document retrieval is more likely to discuss about "lower heart rate" instead of other issues about "heart rate".

29:#NEAR/1(#SYN(ps playstation) 2 #SYN(game games))
The intent of this query is to identify 'ps' and 'playstation', which are commonly used one for the other. SYN operator is used to for "ps" and "playstation" to ensure this. Also, 'game' is the singular term of 'games', so 'game' is also added to this query using #SYN. As the four terms should be a phrase that matched as a whole, a near operator with distance 1 is used.

33:#NEAR/1(#SYN(elliptical ellipticals) #SYN(trainer trainers machine machines))
#SYN is used to match the synonym "elliptical machine". Considering the plural forms, ellipticals, trainers, machines are also included in #SYN operators. Elliptical trainer is noun as a whole, so NEAR operator with distance one is applied.

52:#SYN(avp #NEAR/1(association of volleyball professionals))
This query utilizes two strategies above. first, avp is the abbreviation of "association of volleyball professionals" so that latter is added as a parameter os #SYN. Second, "association of volleyball professionals" is the name of an organization, so it is treated as a phrase by using #NEAR operator.

71:#NEAR/1(#SYN(living live life) in india)
This query uses both strategies. "living live life" are different forms of a word, and all of these three words make sense in this query, so that #SYN operator is used to support different forms. "living in india" express meanings as a whole, so NEAR operator with distance one is used to make sure that they are matched together.

102:#NEAR/1(fickle creek farm)
Strategy b is used in this query. fickle creek farm is a phrase and has specific meaning as a whole, so NEAR operator with distance is used to ensure that search engine match three words together.

149:#SYN(uplift earthquake) at yellowstone #NEAR/1(national park)
This query is super interesting. Adding "earthquake" as a synonyms of "uplift" increase MAP from 0.06 to 0.1903, and increase P30 from 0.1 to 0.3. The reason behind it might be that earthquake has more specific meaning than uplift, and earthquake is a more common word used in daily life and in document as well. "national park" is matched as a phrase by using NEAR with distance one as it has meaning as a whole.

190:#NEAR/1(brooks brothers) #SYN(clearance sale)
Again, both two strategies are utilized here. brooks brothers is the name of the company, so NEAR operator is used to match the whole phrase. Second, sale is a commonly used synonym of clearance so it is added using #SYN operator so that document contains sale but not clearance can also be matched.

2.3. Experimental Results

	Ranked Boolean	BM25 BOW	Indri BOW	Ranked Boolean Syn/Phr	BM25 Syn/Phr	Indri Syn/Phr
P@10	0.1500	0.3000	0.2300	0.2500	0.3100	0.3000
P@20	0.1800	0.2950	0.2800	0.2900	0.3650	0.3250
P@30	0.1667	0.2967	0.2900	0.3033	0.3500	0.3233
MAP	0.0566	0.1304	0.1277	0.1173	0.1717	0.1552

2.4. Discussion

For all three models, structured query set performs better than unstructured query set. There is a great improvements on P@10, P@20, P@30 and MAP after using the strategies mentioned above. The use of synonyms and phrases improve the result as expected. The use of #NEAR operator helps to match a phrase instead of just a word when some terms in the query has specific meaning as a whole. And #SYN helps match synonyms of the query term, so a document without the query term can also be matched if it contains the words of similar meaning. This strategy makes the result more intelligent since the search engine is not only matching the term, but kind of like matching the meaning of the word.

From the chart we can observe, BM25 and Indri performs better than Ranked Boolean for both unstructured and structured query set. One of the main reason is in Ranked Boolean scored is calculated by only considering tf, while in BM25/Indri, more factors such as idf, doclen, etc are used, which provides more insights of corpus, document and query.

3. Experiment 3: BM25 Parameter Adjustment

3.1. k_1

	k_1							
	1.2	0	0.5	1.0	2.4	5	10	100
P@10	0.3000	0.04	0.28	0.3000	0.2900	0.22	0.25	0.2
P@20	0.2950	0.02	0.31	0.2950	0.2900	0.28	0.255	0.205
P@30	0.2967	0.0433	0.3067	0.3000	0.2967	0.28	0.25	0.2033
MAP	0.1304	0.0142	0.1280	0.1303	0.1286	0.125	0.1186	0.1007

.

3.2. b

	b							
	0.75	0	1	0.5	0.25	0.1	0.65	0.95
P@10	0.3000	0.24	0.22	0.25	0.25	0.26	0.26	0.24
P@20	0.2950	0.28	0.295	0.295	0.305	0.29	0.295	0.305
P@30	0.2967	0.2733	0.3067	0.3033	0.3033	0.2967	0.3167	0.32
MAP	0.1304	0.1067	0.1205	0.1295	0.1302	0.1243	0.1287	0.1256

3.3. Discussion

Seven values between 0 to 100 is tested for k1 and seven values between 0 to 1 is tested for b. I tried to get performance data resulted from a large range so that I can analyze the trend of performance with different parameters values. Edge values for these two parameters (0 for k1, 0 and 1 for b) is checked so that the extreme case can be analyzed.

From the k_1 chart we can observe: the performance of the search engine is worst at k_1 = 0, and then the performance goes better with k_1 increasing. At the point k_1 = 1.2, the performance reaches the peak, after which, the performance decrease with k_1 keep increasing. The reason of the bad performance when k_1 = 0 is that document term frequency has no effect on the score at this point, and rare words and repeated query terms dominate. The search engine performs best with k_1 value around 0.5~5. When k_1 goes to very large, the performance is also bad. This is because that the tf weight goes to 0 with k_1 increasing to infinity. When k_1 is very large and tf << k_1, the tf weight is closed to 0 no matter what tf is, in other words, the difference of term frequency does not effect the final score much. This is why the search engine doesn't performs well when k_1 is large.

From the parameter b chart we can observe: The search engine perform worst at b = 0, and then behaves better with b increasing. When b in the range of 0.25~0.75, the performance is best and MAP is around 0.13. When b is larger than 0.75, search engine behaves worse with b increasing. Overall, the effect of changing b is minor. Expect for b = 0, all the values of b gives MAP about 1.20~1.31. The reason of bad performance at b = 0 is because that document length is ignored and long documents are always more likely to be retrieved.

4. Indri Parameter Adjustment

4.1. μ

	μ							
	2500	0	500	1000	2000	5000	8000	1000000
P@10	0.2300	0.2600	0.3200	0.2700	0.22	0.2200	0.19	0.1900
P@20	0.2800	0.3000	0.3100	0.3300	0.3050	0.2700	0.305	0.2600
P@30	0.2900	0.3100	0.3167	0.3167	0.2900	0.2933	0.3033	0.2367
MAP	0.1277	0.1254	0.1346	0.1316	0.1304	0.121	0.1178	0.0943

4.2. λ

	λ							
	0.4	0	0.2	0.6	0.8	1	0.01	0.95
P@10	0.2300	0.27	0.25	0.20	0.18	0.03	0.27	0.15
P@20	0.2800	0.3000	0.295	0.275	0.25	0.015	0.3000	0.205
P@30	0.2900	0.3133	0.3000	0.27	0.26	0.01	0.3100	0.2333
MAP	0.1277	0.1346	0.1318	0.1241	0.1174	0.0042	0.1340	0.1073

4.3. Discussion

I chose seven values from 0 to 1000000 for mu and seven values from 0 to1 for lambda in order to get data resulted from a large range and analyze the trend of the performance with input parameters. Edge cases (0 for mu, 0 and 1 for lambda) are tested so that extreme cases can be checked.

From the mu chart we can observe that the performance of the search engine is going up from mu = 0 to mu = 500, and behaves best when mu is in the range of 500 ~ 2000 with MAP around 0.13. After that, the performance drops with mu increasing and MAP is lowered to 0.094 when mu reaches 1000000. In Indri model, smoothing decreases as mu goes to 0. When mu = 0, there is maximum likelihood only and no smoothing. The choice of the value for mu is depends on the length of the document. Probabilities are more granular in short documents, so larger mu is better for short documents. By contrast, in long

documents, probabilities are more smooth, so large mu is less important. In this dataset, 500~2000 is most suitable with the document length and gives best performance.

From the lambda chart we can find that the search engine performs best when lambda = 0 and the performance is worse with lambda increasing to 1. In Indri model, smoothing decreases as lambda goes to 0. In this query set, the average length of query is short. Short queries contains few query terms, so every query term must match. “idf weighting” is less important for short queries, so small lambda is best. This is why the performance peaked when lambda = 0 and decrease with lambda increasing in this experiment.