Your Name: Jiachen Li

Your Andrew ID: jiachenl

Homework 2

Statement of Assurance

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

1 Experiment 1: Baselines

	Ranked Boolean	BM25 BOW	Indri BOW
P@10	0.1500	0.2900	0.2400
P@20	0.1800	0.3050	0.2750
P@30	0.1667	0.3267	0.2967
MAP	0.0566	0.1325	0.1275
Time	00:09	00:09	00:09

2 Experiment 2: Queries with Synonyms and Phrases

2.1 Queries

10:#NEAR/3(#SYN(cheap inexpensive) internet)

12:#SYN(djs #NEAR/2(disc jockey)) djs.url

26:#NEAR/2(#SYN(lower slower) heart rate)

29:#SYN(#NEAR/2(ps 2) #NEAR/2(playstation 2)) games

33:#NEAR/2(elliptical #SYN(trainer train))

52:#SYN(avp #NEAR/3(association volleyball professional)) avp.url

71:#SYN(living life) in india

102:fickle creek farm #NEAR/2(fickle creek farm)

149:uplift at #SYN(#NEAR/2(yellowstone national park) #NEAR/2(yellowstone park))

190:#NEAR/2(brooks.title brothers.title) #NEAR/2(brooks brothers) clearance

2.2 Query descriptions

Query 10: #NEAR/3(#SYN(cheap inexpensive) internet)
Given that "inexpensive" has the similar meaning as "cheap" here, I combined them with #SYN operator. Besides, since "cheap internet" can be further interpreted as a phrase, then I applied the #NEAR operator. The parameter of #NEAR is chosen as 3 because there may be some other word in the middle of "cheap" and "internet", for example, "cheap high speed internet".

Query 12: #SYN(djs #NEAR/2(disc jockey)) djs.url
According to my guess, the "dj" here should refer to "disc jockey", so I added this term in the #SYN operator. Moreover, I duplicated "djs" with url field because I think "djs" is an abbreviation and then very likely to appear in the such field.

Query 26: #NEAR/2(#SYN(lower slower) heart rate)
These three terms "lower heart rate" together have specific meaning, so just search them within NEAR. Moreover, "slower", as a synonym of "lower", is added to improve the recall.

Query 29: #SYN(#NEAR/2(ps 2) #NEAR/2(playstation 2)) games
Since "ps 2" stands for Sony's PlayStation, I first apply NEAR operator to make them a phrase. Besides, I think their full name "playstation 2" might lead to similar or even better result, so I add this term, and combine the results of these two with #SYN.

Query 33: #NEAR/2(elliptical #SYN(trainer train))
These two terms together can form a phrase, so I used #NEAR. Besides, to improve the searching result, I added a term "train" that I think may also have a good chance to appear in the same document as "elliptical trainer".

Query 52: #SYN(avp #NEAR/3(association volleyball professional)) avp.url
Since "avp" here stands for "Association of Volleyball Professionals", I added this phrase and combined it with "avp" through #SYN. In addition, with similar reason as query 12, I also added another "avp" but in url field.

Query 71: #SYN(living life) in india
According to the query terms, I suppose the background information need may be the knowledge about living or life experience in India, so I added the "life", which may lead to the similar result as "living".

Query 102: fickle creek farm #NEAR/2(fickle creek farm)
I think these 3 words stand for a particular farm (not quite sure), so I added a #NEAR operator which make them a phrase..

Query 149: uplift at #SYN(#NEAR/2(yellowstone national park) #NEAR/2(yellowstone park))
Similar as Query 29, since "yellowstone national park" has specific meaning, and "yellowstone park" may have the same meaning, so I applied #NEAR for each of them and then combined their result by #SYN.

Query 190: #NEAR/2(brooks.title brothers.title) #NEAR/2(brooks brothers) clearance

“Brooks brothers” should be the name of a brand. Since this kind of information is more likely to appear in the title, I used #NEAR operator for them in title field. Besides, I also added #NEAR for the body field to enhance the searching result.

2.3 Experimental Results

	Ranked Boolean	BM25 BOW	Indri BOW	Ranked Boolean Syn/Phr	BM25 Syn/Phr	Indri Syn/Phr
P@10	0.1500	0.2900	0.2400	0.3300	0.4000	0.3900
P@20	0.1800	0.3050	0.2750	0.3650	0.3950	0.4100
P@30	0.1667	0.3267	0.2967	0.3433	0.3700	0.4167
MAP	0.0566	0.1325	0.1275	0.1333	0.1921	0.2015
Time	00:09	00:09	00:09	00:04	00:04	00:04

2.4 Discussion

Operators & With vs Without Synonyms and phrases

In this experiment, #SYN and #NEAR operators are used to construct my own structured queries. The #SYN operator combines the inverted list of each arguments. It provides a chance to improve the recall, but if you can add some “right” synonymous term, it can also improve the precision. #NEAR/n operator is used to combine the phrases here. In this situation, the parameter n is often taken as 1-3. The #NEAR operator is very helpful here as it can not only improve the precision, but also improve the searching efficiency by reducing the length of inverted list.

Comparing the results before and after using these synonyms and phrases, it is obvious that the use of synonyms and phrases helps all the three retrieval models outperform the result on BOW, which matches my expectation that the synonyms and phrases can be quite helpful no matter which retrieval model is used. Except for the benefit from using these two operators, another high-level explanation for this is that by using synonyms and phrases, the structured query may have a better representation of the background information need, then the retrieval model can take advantage of the corresponding concepts and phrases to perform a more effective search, thus produce a better result.

Ranked Boolean vs BM25/Indri

Besides, comparing the results between Ranked Boolean and BM25/Indri on each query set, it is not difficult to find the BM25/Indri model always achieve a better performance than Ranked Boolean, which indicates the difference between exact-match retrieval model and best-match retrieval model. In Ranked Boolean retrieval, the search engine only picks out the documents that have exactly the same query terms and rank then by the term frequency. This selection process mainly focuses on how important a query term in a specific document, but doesn’t consider how important this term in the whole documents set. Moreover, this ranking method also doesn’t take the influence of different document length into consideration. While, on the contrary, these two best-match retrieval model both have the idf or idf-like factors and they both compensate the influence of document length, which makes them more reasonable

and more effective. Of course, one thing that I have to mention is that under the constrain of not using boolean #AND operator, the performance of Ranked Boolean maybe not as good as it should be.

BM25 vs Indri

Moreover, it can be seen from the table that the performance of BM25 and Indri is quite similar. Even if in my test the Indri outperforms BM25 a little bit, I never think Indri must be better than BM25, it really depends on the query you use. Although these two retrieval models are based on different theory, they do have the similar architecture based on my implementation experience. Just as what I have mentioned in the last paragraph, they both consider the importance of a term in a document (tf) and in the whole document set (idf), and they also compensate for the documents of different lengths. One difference of these two model is the way they combine the score list with their default operators, the Indri need to calculate a default score for those terms that don't have a given document in their score list, while the BM25 model treats this kind of score as 0. In my opinion, this process makes Indri more robust.

Running Time

At last, one thing that is beyond my expectation is the running time. At first I think the structured query may need more time to run, but the result turns out that, if we take the running time on BOW as the baseline, the structured queries only take half of that time to run. However, this is not hard to understand as the field or the query operator like #NEAR can significantly decrease the length of term's inverted list. Since all the models' default operators will have to traverse all the inverted lists of the terms, the shorter length makes searching and ranking more efficient. Furthermore, this also implies that the structured query do improve the result.

3 Experiment 3: BM25 Parameter Adjustment

3.1 k_1

	k_1					
	0.3	0.6	1.2	2.4	4.8	9.6
P@10	0.2800	0.2800	0.2900	0.2900	0.3100	0.3000
P@20	0.3050	0.3150	0.3050	0.2950	0.3000	0.2700
P@30	0.3333	0.3333	0.3267	0.3200	0.3067	0.2900
MAP	0.1290	0.1301	0.1325	0.1333	0.1327	0.1255
Time	00:09	00:09	00:09	00:09	00:09	00:09

.

3.2 b

	b						
	0.15	0.30	0.45	0.60	0.75	0.90	0.95
P@10	0.2700	0.2500	0.2500	0.2500	0.2900	0.2700	0.2500
P@20	0.3000	0.3050	0.3000	0.3100	0.3050	0.3100	0.3100
P@30	0.2933	0.3033	0.3133	0.3133	0.3267	0.3367	0.3333
MAP	0.1266	0.1301	0.1337	0.1338	0.1325	0.1320	0.1291
Time	00:09	00:09	00:09	00:09	00:09	00:09	00:09

3.3 Discussion

According to the model of BM25, both k_1 and b are used in the term frequency (tf) weight part as shown below:

$$\text{tf\_weight} = \frac{\text{tf}_1}{\text{tf}_1 + k_1\left((1-b) + b\frac{\text{doclen}}{\text{avg\_doclen}}\right)}.$$

The parameter k_1 is a positive variable that calibrates the document term frequency scaling, and both a very large and small very k_1 will probably hurt the result. To be specific, if k_1 is too large, then the effect of tf_1 in the denominator will diminish, this corresponds to using the raw term frequency. While on the other hand, if k_1 is too small, then the tf_weight for documents will all be near to 1, this corresponds to a binary model (i.e., not using frequency term at all). Based on that, I choose the value of k_1 in the interval of [0.3, 9.6], with 2 times larger for each pick to get an idea about the trend. According to the result, the performance tends to become better when k_1 grows from 0.3 to 2.4 at first. However, as k_1 becomes even larger, the performance turns out to go down. Just as what I discussed above, this kind of trend matches my expectation well, and it also indicates that there should be some optimal value to achieve the best trade-off.

Next, it is not difficult to notice that the parameter b determines the scaling by document length. Though the valid value of b is in [0, 1], we can see that if b is very close to 0, the normalization part will be near to 1 and thus there will be no normalization at all. And if b is very close to 1, then it corresponds to fully scaling the term weight by document length. Based on these analyses, I first pick the value of b in the range between 0.15 and 0.90 with an equal step length 0.15. Besides, I also added another sample with b=0.95 to give a more clear intuition about the trend. For adjusting b, the result also becomes better as the b increases from 0.15 to 0.60. However, the result begins to go down slowly as b grows even larger. This is under my expectation since, in my view, this trend suggests that the term $\frac{\text{doclen}}{\text{avg\_doclen}}$ can be helpful in certain extent. However, suppose that b is very large (or even assume b=1), if the search engine encounters a sample with doclen >> avg_doclen, then the punishment for document length will be too large so as to have some bad effect on the results. Of course this is not what we want, and it could also be the reason that why we use b to tune the scaling for different corpus instead of using $\frac{\text{doclen}}{\text{avg\_doclen}}$ directly.

Finally, we can also find that adjusting parameters doesn't affect the running time, this is true since the changing of parameters doesn't reduce any calculation cost.

4 Indri Parameter Adjustment

4.1 μ

	μ					
	500	1500	2500	3500	4500	5500
P@10	0.3300	0.2300	0.2400	0.2000	0.1900	0.1800
P@20	0.3300	0.3050	0.2750	0.2700	0.2950	0.3000
P@30	0.3233	0.3067	0.2967	0.2967	0.3000	0.3033
MAP	0.1364	0.1317	0.1275	0.1224	0.1203	0.1209
Time	00:09	00:09	00:09	00:09	00:09	00:09

4.2 λ

	λ					
	0.05	0.2	0.4	0.6	0.8	0.95
P@10	0.1500	0.2000	0.2400	0.2600	0.2800	0.2900
P@20	0.2150	0.2550	0.2750	0.3050	0.3100	0.3100
P@30	0.2467	0.2700	0.2967	0.3033	0.3067	0.3133
MAP	0.1100	0.1202	0.1275	0.1330	0.1364	0.1373
Time	00:09	00:09	00:09	00:09	00:09	00:09

4.3 Discussion

Consider the smoothed version of $p(q|d)$, i.e.

$$p(q_i|d) = \frac{\mathrm{tf}_{q,d} + \mu \cdot p_{mle}(q_i|C)}{\mathrm{length}(d) + \mu},$$

we can see that the parameter μ is used for smoothing, and the result estimate of $p(q|d)$ will be between $\frac{\mathrm{tf}_{q,d}}{\mathrm{length(d)}}$ (i.e., when μ is small) and $\frac{\mathrm{ctf}_q}{|\mathrm{length}_{tokens}(C)|}$ (i.e., when μ is large). As the $\mathrm{length}(d)$ may be large, I took the step as 1000 in order to get a clear intuition for the result of tuning μ, and the parameters I chose are from 500 to 5500. When adjusting μ, the results turns to become bad as the μ increases, which matches what I expect. According to my analysis, this is reasonable because for a given term, $\frac{\mathrm{tf}_{q,d}}{\mathrm{length(d)}}$ should be a more useful way to express the relevance of this document than $\frac{\mathrm{ctf}_q}{|\mathrm{length}_{tokens}(C)|}$, as the latter mainly describes the importance of this term in the whole collection.

Besides, according to the form of Jelinek-Mercer smooth, i.e. $p(q|d) = \lambda p_{mle}(q|d) + (1-\lambda)p_{mle}(q|C)$, it can be seen more clearly that the parameter λ determines the weight between $p_{\mathrm{mle}}(q|d)$ and $p_{\mathrm{mle}}(q|C)$. A larger λ indicates more weight is given to $p_{\mathrm{mle}}(q|d)$, while a smaller λ indicates the $p_{\mathrm{mle}}(q|C)$ has more weight. Since λ is in the [0, 1] interval, I first picked the samples with a step length of 0.2, e.g. 0.0, 0.2, 0.4, 0.6, 0.8, 1.0. Moreover, I adjusted the 0, 1.0 into 0.05 and 0.95 respectively to avoid hitting the extreme cases. When adjusting λ, the result tends to become better as λ increases, this indicates the result becomes better as more weight given to $p_{\mathrm{mle}}(q|d)$, which is consistent with the effect of adjusting μ, so as to my expectation.

Of course, the running time here also remains unchanged for the same reason as talked in the last section.