Your Name: Poorna Omprakash

Your Andrew ID: pompraka

Homework 2

1 Statement of Assurance

I certify that all the work in this homework was done entirely by me.

2 Experiment 1: Baselines

	Ranked Boolean	BM25 BOW	Indri BOW
P@10	0.2467	0.4967	0.3400
P@20	0.2283	0.4667	0.3000
P@30	0.1211	0.4178	0.2711
MAP	0.2370	0.4788	0.3134
num_q	30	30	30
Time	13.3 s	18.7s	12.7s

BM25 and Indri perform much better on both time taken to run, as well as Precision and MAP. These results were expected.

However, the results for my Indri experiments were not as good as I had hoped they would be. I had expected a close performance to BM25, if not better. However, the results were much lower than that for BM25, although they were much better than the Ranked Boolean model results. Time taken to run by the BM25 model was slightly higher than that of Indri or Ranked Boolean, which had comparable run times.

Although I had not explicitly expected this result, it does not come as a surprise.

3 Experiment 2: Parameter adjustment

Describe what parameters you chose to adjust and the range of parameter values that you explored. Give a brief justification for your choices.

For BM25, I chose two different sets of parameters: One with k_1 = 50 and b=0.50, and the other with k_1=100 and b=0.2.

The reason I chose these parameters was so that I could experiment with parameters on either side of the default parameters I had used earlier, i.e, I wanted to test out large k_1 values, along with b values that were both smaller and larger than the default values, in the hope that I could observe some trends.

For Indri, I tested out results by changing the smoothing parameter from df to ctf.

BM25:

	Setting 1		**Setting 2**	
	K_1 = 50	B = 0.50	K_1=100	B=0.2
P@10	0.5033		0.4100	
P@20	0.4267		0.3883	
P@30	0.4011		0.3867	
MAP	0.4471		0.3787	
num_q	30		30	
Time	18.18s		18.78s	

Indri:

	Setting 1	
	Smoothing parameter = ctf	Lambda=0.4, mu=2500
P@10	0.2400	
P@20	0.2283	
P@30	0.2211	
MAP	0.2374	
num_q	30	
Time	12.96s	

For BM25, increasing the value of k_1 and reducing the value of b significantly reduced the precision and recall of the results produced. I was not sure what kind of results to expect with tweaking of parameters, but turned out that increasing the value of k_1 reduced the precision and recall for BM25.

For Indri, changing the smoothing parameter to ctf had a negative impact on the precision and recall.

4 Experiment 3: Different representations

For this experiment, I used the default parameters.

	Indri BOW (body)	url	keywords	title	inlink
P@10	0.3400	0.2133	0.1733	0.1800	0.2333
P@20	0.3000	0.1767	0.1567	0.2167	0.1700
P@30	0.2711	0.1689	0.1489	0.2044	0.1356
MAP	0.3134	0.1281	0.1123	0.1583	0.100
num_q	30	30	30	30	30
Time	12.7s	5.12s	5.49s	5.04s	3.93s

For this experiment, I observed that queries with the "body" field took the longest time to run. It also had the best results in terms of precision and accuracy, compared to queries with other fields. This result was something I had expected, since the general belief is that the body section will contain more terms, and hence there is likely to be more matches for query terms in the body section of the document. This was the same as Indri base case.

The URL field had the second highest scores in terms of precision and recall. Although this wasn't a result I had expected, it didn't come as a surprise, since the URL field contains significant information that could be relevant to many queries.

The keywords field had a low precision and recall, but the runtime was fast. I did not expect very good results for the keywords field, so this did not come as a surprise to me. Faster runtime came at the cost of lower precision and accuracy

The title field did surprisingly badly on the precision and recall statistics. I would expect the title section to be relevant and return matches for the important query terms. However, this does not seem to be the case.

The inlink field had a very, very short run time, which really surprised me. It also had a surprisingly good precision at ten, but this precision fell rapidly.

5 Experiment 4: Sequential dependency models

The weights for my SDM were mostly by trial-and-error, although I did give a greater weight parameter to documents that contained all the query terms - #AND(all query terms) got the highest weight.

Query #1: Provide your structured query for query "obama family tree"

1:#WEIGHT(0.60 #AND(obama family tree) 0.3 #AND(#NEAR/2(obama family) #NEAR/1(family tree)) 0.2 #AND(#UW/7(obama family) #UW/5(family tree)))

Query #2: Provide your structured query for query "uss yorktown charleston sc"

56:#WEIGHT(0.6 #AND(uss yorktown charleston sc) 0.3 #AND(#NEAR/1(uss yorktown) #NEAR/1(yorktown charleston) #NEAR/1(charleston sc)) 0.2 #AND(#UW/5(uss yorktown) #UW/5(yorktown charleston) #UW/5(charleston sc)))

	Indri BOW	BM25 BOW	Indri SDM
P@10	0.3400	0.4967	0.240
P@20	0.3000	0.4667	0.228
P@30	0.2711	0.4178	0.221
MAP	0.3134	0.4788	0.23
num_q	30	30	30
Time	12.7s	18.7s	229s

Surprisingly, accuracy fell quite a bit compared to baseline Indri queries, and were significantly lower than BM25BOW queries. I would have expected more complex queries to return more relevant documents, but this did not seem to be the case. It is possible that my queries were not crafted well enough.

The increase in runtime was manifold compared to baseline Indri and BM25BOW. However, this increase in computation cost did not translate to better accuracy or recall.

6 Experiment 5: Multiple representations + SDMs

I used the title field from experiment 3. I have been, and still am, under the impression that the title field is informative and should return relevant documents when queried, and I wanted to test this theory out.

Query #1: Provide your structured query for query "obama family tree"

1:#WEIGHT(0.2 #AND(obama.title family.title tree.title) 0.8 #WEIGHT(0.75 #AND(obama family tree) 0.1 #AND(#NEAR/1(obama family) #NEAR/1(family tree)) 0.15 #AND(#UW/8(obama family) #UW/8(family tree)))

Query #2: Provide your structured query for query "uss yorktown charleston sc"

56:#WEIGHT(0.1 #AND(uss.title yorktown.title charleston.title sc.title) 0.9 #WEIGHT(0.65 #AND(uss yorktown charleston sc) 0.2 #AND(#NEAR/1(uss yorktown) #NEAR/1(yorktown charleston) #NEAR/1(charleston sc)) 0.15 #AND(#UW/8(uss yorktown) #UW/8(yorktown charleston) #UW/8(charleston sc)))

	Indri BOW	Results From Experiment 3	SDM	Experiment 3 Query & SDM Query
P@10	0.3400	0.1800	0.240	0.226
P@20	0.3000	0.2167	0.228	0.2050
P@30	0.2711	0.2044	0.221	0.1922

MAP	0.3134	0.1583	0.23	0.1783
num_q	30	30	30	30
Time	12.7s	1.04s	229s	212.72s

This experiment ran very slowly. This was something I expected, since the queries were complex, and the SDM experiments, which had similar queries, also took a log time to run.

Yet again, surprisingly, the accuracy was even lower than that I achieved for the SDM queries. The computation cost increased manifold, but this did not reflect in the accuracy. It is possible that these results were due to queries that were not crafted too well, but I got similar results irrespective of how I crafted my queries.

7 Analysis of results

The three approaches to forming queries (BOW, SDM and Multiple Representations) each behaved quite like I expected them to, in almost all ways except one. The run times were as expected, with BOW running fast, while SDM and Multiple Representations running really slowly. However, SDM and Multiple Representations queries ran much slower than I'd expected. I had expected the rum time to be about 10x slower than BOW. However, the queries ran about 40x slower, which was very surprising. Although this could be due to the way I formed my queries, I don't expect the results to get significantly better even with very well crafted queries.

What I found most surprising was the fact that BOW achieved greater precision and recall than SDM or multiple representations. I had expected BOW to be the least efficient, but it turned out to be otherwise. It is possible that SDM and multiple representations on this particular corpus and query set did not function that well – this is something I cannot comment on based on my experiments.

Between Indri and BM25, I did not have any preconceived expectations. I had assumed that that two would have similar performances in terms of run time and accuracy. However, I was surprised to find BM25 doing much better on the query set than Indri. This was a trend I observed irrespective of how I tweaked my parameters for both Indri and BM25. BM25 always yielded better results than Indri. This difference could be because of the difference in mathematical models used in Indri and BM25. A deeper analysis of the math behind these two methods might yield a better answer. BM25 also seemed to have a higher computational cost than Indri, since it took longer to run than Indri in all the experiments.

The different fields used in queries reinforced my belief that the body section of a document is the most important. The queries that were run on the body section yielded the best results among all the fields. There were some unexpected results from running experiments on different fields, specifically when I saw that the title field did not return results that were as good as I'd expect. The inlink field does not seem as important a part as the other fields when it comes to running queries.

Among the different operators, I enjoyed implementing the #WEIGHT operator as much as I enjoyed learning about it. Sadly, though, the results from this operator were not as good as I'd expected. The operator makes a lot of sense logically, but the retrieved documents did not seem to do too well. The #SUM operator returned good results, which were expected of the operator.

The biggest disappointment for me was Indri's #AND, because it just didn't do as well as I had expected it to do. I tweaked the parameters and experimented, but wasn't able to achieve significant differences in results. Although this was computationally similar to BM25, it came as a surprise that it did not return results that were as good. This could be due to the selection of my mu and lambda parameter values – I did not try tweaking these when I ran my experiments.

8 The software implementation

My software fixes a lot of issues I had from Homework 1, specifically the speed constraint.

While building the software for homework 2, I came across difficulty in implementing the Unordered Window operator, and many of my implementations failed. It took me four different approaches before I got to the right one.

My system also encountered errors: "Exception in thread "main" java.lang.OutOfMemoryError: Java heap space" when I was running the SDM queries for Indri.

My implementation of the BM25 SUM was the best part of my software, and also possibly the easiest implementation. It gave me better results than I expected.

I spent a lot of time trying to figure out the implementation of the UW operator. In hindsight, I should not have dwelled upon it for two whole days, since it took up a lot of my time.