Sneha Shankar Narayan
CS646 — Fall 2014 snehas@cs.umass.edu

Report on P 2

The system

Search Engine: Galago

I used the search engine Galago, and I felt it was fairly simple to use. The setup was painless; all I had to do was download the binary and run it.

The command line options were intuitive and I could easily perform batch-searches that were needed for the third step of the project. Galago can also run a web server which I found was simple to use during the query formulation process.

The results gotten from Galago varied when it comes to the documents retrieved and that is described in the later sections.

Query creation

Query creation was interesting in the sense you wanted to get enough relevant documents, not too many or too less. It was interesting to play around with the exact words in the query and see how the system reacted. I came up with the query "curie radiation discovery" for the books dataset thinking that since books was about data before 1920 I could get some pretty interesting results. When I just used the query "Marie curie", I got a list of all relevant documents in the top 10 result mostly which mentioned "Marie Curie" in passing. However adding more words to the query made the result set more interesting. This exercise also helps in defining what "relevance" actually means.

Document judging

This was a pretty interesting exercise and gave an insight to how evaluation of an IR system actually works. Reading the document and coming up with relevance values showed the importance of evaluation measures for an IR system.

Creating the evaluation

The final step of P2 was about creating the evaluation. This just involved writing a script to read through the relevance judgments provided and comparing it against the ranked list given by Galago. The evaluation measures used were Average Precision, Mean average precision, and NDCG. This makes do for interesting conclusions about the search engine. The formulas used for calculating these values are as follows.

- Average precision [1]:

$$AveP = \frac{\sum_{k=1}^{n} P(k) * rel(k)}{NumberOfRelevantDocuments}$$

where rel(k) is an indicator function equaling 1 if the item at rank k is a relevant document, zero otherwise.

- Mean average precision: Compute the average of average precisions across all queries.
- Precision at 10: This is equal to the number of relevant documents that are retrieved when there 10 documents are retrieved/10
- NDCG (Formula used is the one mentioned in Kaggle) [2]

$$\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

$$\mathrm{nDCG_p} = \frac{DCG_p}{IDCG_p}$$

The evaluation

Performed experiments and results

Summarizing the results obtained.

Books

Query No	Average Precision	Precision at 20	NDCG at 20
2	0.352221662181	0.8	0.61114492934
5	0.0754985754986	0.0	0.354367255313
6	0.559106075896	0.4	0.59046588894
7	0.0	0.0	0.0
11	0.234361345472	0.4	0.571989907287
16	0.696378846753	0.6	0.892767104217
17	0.727555220791	0.9	0.85198961753
20	0.311071428571	0.5	0.658837225852
22	0.242262536013	0.2	0.523246797041
28	0.31403057931	0.4	0.604024655918
29	0.5	0.2	0.652012130191

Robust

Query No	Average Precision	Precision at 20	NDCG at 20
1	0.0619001610306	0.1	0.418400105268
5	0.0698717948718	0.0	0.282167911853
10	0.453585579208	0.6	0.635648770624
12	0.395469345469	0.4	0.715928000847
15	0.286652134378	0.4	0.556991610934
17	0.607142857143	0.3	0.953001274216
18	0.798737373737	0.8	0.926020454977
20	0.102674363486	0.0	0.477379984208
24	0.263716356108	0.1	0.640425522164
27	0.605326623504	0.5	0.875841503874
30	0.553289908485	0.6	0.523913511398
33	0.714285714286	0.5	0.999273228036
34	0.0625	0.0	0.321954465307
35	0.5	0.1	0.630929753571

Robust - community generated

Query No	Average Precision	Precision at 20	NDCG at 20
301	0.153224715678	0.6	0.758304318499
302	0.498046209353	0.7	0.915000273559
303	0.21855776526	0.1	0.770087137143
304	0.0860814985251	0.3	0.560990257318
305	0.00367556971331	0.0	0.278942945651
306	0.204549293041	0.8	0.971213155415
307	0.143441627969	0.6	0.723512939507
308	0.435897435897	0.2	0.919720789148
309	0.0	0.0	0.0
310	0.141025641026	0.3	0.786472491993
311	0.637789474316	0.9	0.970201337961
312	0.190909090909	0.3	0.852927865061
313	0.767989560453	1.0	1.0
314	0.0	0.0	0.0
315	0.00199306497814	0.0	0.270238154427
316	0.56980698544	0.7	0.878690728282
317	0.152380952381	0.4	0.771760914325
318	0.0138187229437	0.2	0.592432998229
319	0.041062562918	0.2	0.556419264628
320	0.0397031539889	0.1	0.333333333333

The rest of the values for the community robust corpus is present in the result file *robust04.qrels-output* present in the zip file for the code.

Mean average precision values:

- **Books:** 0.364771479135
- **Robust:** 0.391082300836
- **Community generated robust:** 0.229561528619

Conclusion from above experiments

- The mean average precisions are fairly small which indicates that the system can do better.
- There are a lot of queries where the precision at 10 is zero but the average precision is not zero which indicates that the system is really slow in retrieving the relevant documents.

Performance on Class queries vs community robust queries

Looking at the mean average precision values we can come to the conclusion that the system performs better with the class queries rather than the community generated queries. This can mean that since the relevance scale was 0 - 5 in the case of class queries the definition of a relevant document is better defined and since documents with relevance judgement greater than or equal to 2 is considered relevant the number of documents judged like this would be more than in the case where a binary decision has to be made.

Also since I have retrieved 100 documents from galago for the queries in the community robust collection but we have judgments for around 1500 documents per query in that collection, the mean average precision might be a bit skewed. I've tried to negate this effect by considering the fact that if the total relevant documents is greater than 100 (because of the 1500 judgements that we have) then I consider the number of relevant documents to be 100 while calculating average precision.

Performance on robust vs books

The mean average precision for the books collection and the class robust collection does not differ too much. Robust performs slightly better (0.39 as opposed to 0.36 in the case of the books collection). This can be due to the fact that the books corpus has very limited data (in the form of variety of topics) and the queries have no co-relation on the data that the corpus contains.

Weak point and fix

I have noticed that for a few queries in all the collections no relevant documents were retrieved. For example Query Number 7 in the books collection "rio de gabio" seems to have no relevant documents and the judgements also show that none of the documents are relevant. This indicates that there are absolutely no relevant documents for the query in the collection and because of this the IR system should not be penalized (it's precision reduces because of this). One fix that I can think of for this would be to do a quick check if there are any relevant documents at all in the corpus for a query and if not just return an appropriate result instead of retrieving documents which are obviously not relevant because there are no relevant documents.

What can make a better IR system?

Some of the techniques I can think of are:

- Query expansion: Words can be added to the queries from thesaurus or from the concept that the query is talking about to get better, more relevant results

- Pseudo relevance feedback: This can also improve the results.
- The IR system should also somehow have a vague idea about the corpus that is being searched through such that more relevant details are retrieved

References

[1] `en.wikipedia.org/wiki/Information_retrieval`

[2] `http://www.kaggle.com/wiki/NormalizedDiscountedCumulativeGain`