SEARCH (Z534)
FALL 2017
Assignment II
Snehal Vartak

GitHub Repo: https://github.iu.edu/snehchem/vartak-information-retrieval

GitHub Folder: Assignment_2

Output files from task 2 and task 3 can be found under
https://github.iu.edu/snehchem/vartak-information-retrieval/tree/master/Assignment_2/output

Task 1: Implement your first search algorithm

Code File: easySearch.java

- Update the path to the index by modifying the value of variables "indexPath" in the code.
- The search query can be set by changing the value of queryText in the code.

Task 2: Test your search function with TREC topics

Code File: searchTRECtopics.java

- Update the path to the index by modifying the value of variables "indexPath" in the easySearch.java file.
- Update the path to the topics for generating the search queries by updating the value of "topics" in searchTRECtopics.java
- Similarly update the output path by modifying the variables "shortQuery" and "longQuery".

Task 3: Test Other Search Algorithm

Code File: compareAlgorithms.java

- Update the path to the index by modifying the value of variables "indexPath" in the compareAlgorithms.java file.
- Update the path to the topics for generating the search queries by updating the value of "topics" in compareAlgorithms.java
- Similarly update the output path by modifying the variables "shortQuery" and "longQuery".

Task 4: Algorithm Evaluation

Short query:

Evaluation metric	Your algorithm	Vector Space Model	BM25	Language Model with Dirichlet Smoothing	Language Model with Jelinek Mercer Smoothing
P@5	0.1480	0.2960	0.3080	0.3480	0.2880
P@10	0.1460	0.3020	0.3000	0.3260	0.2800
P@20	0.1350	0.2600	0.2710	0.2890	0.2430
P@100	0.0980	0.1648	0.1678	0.1690	0.1602
Recall@5	0.0234	0.0539	0.0488	0.0620	0.0534
Recall@10	0.0539	0.0960	0.0879	0.0995	0.0911
Recall@20	0.0839	0.1416	0.1378	0.1467	0.1302
Recall@100	0.2196	0.3578	0.3572	0.3468	0.3320
MAP	0.1050	0.1975	0.2005	0.2059	0.1932
MRR	0.2863	0.4696	0.4772	0.4785	0.4637
NDCG@5	0.1627	0.3116	0.3246	0.3517	0.3063
NDCG@10	0.1621	0.3183	0.3199	0.3420	0.3011
NDCG@20	0.1619	0.3043	0.3124	0.3326	0.2888
NDCG@100	0.1919	0.3213	0.3259	0.3311	0.3121

Summary for Short Queries:

- Based on above table we can see that for short queries language Model with Dirichlet Smoothing gives the best results via Precision, Recall, MAP, MRR as well as NDCG evaluation methods.
- Overall the MRR and MAP evaluation scores are very similar for Vector Space, BM25, LM Dirichlet and LM with Jelinek Mercer models. The performance of my search algorithm is the poorest of all.

Long query:

Evaluation metric	Your algorithm	Vector Space Model	BM25	Language Model with Dirichlet Smoothing	Language Model with Jelinek Mercer Smoothing
P@5	0.1280	0.2560	0.2840	0.2560	0.2320
P@10	0.1240	0.2440	0.2440	0.2420	0.2140
P@20	0.1140	0.2210	0.2340	0.2340	0.2120
P@100	0.0742	0.1406	0.1488	0.1460	0.1378
Recall@5	0.0187	0.0349	0.0402	0.0403	0.0406
Recall@10	0.0368	0.0621	0.0703	0.0708	0.0658

Recall@20	0.0595	0.1064	0.1159	0.1270	0.1136
Recall@100	0.1743	0.2929	0.3167	0.3340	0.2901
MAP	0.0664	0.1529	0.1676	0.1586	0.1514
MRR	0.2563	0.4528	0.4597	0.3475	0.3640
NDCG@5	0.1388	0.2819	0.3029	0.2499	0.2348
NDCG@10	0.1341	0.2682	0.2729	0.2473	0.2294
NDCG@20	0.1310	0.2586	0.2718	0.2590	0.2410
NDCG@100	0.1466	0.2694	0.2872	0.2753	0.2609

Summary for Long Queries:

- Based on above table we can see that for long queries language Model with Dirichlet Smoothing gives the best results for Recall whereas BM25 gives the better results for Precision and NDCG.
- MAP value is the poorest for my algorithm whereas it's almost the same for all other search algorithms.
- MRR evaluation scores are better for Vector Space and BM25.
- The performance of my search algorithm is the poorest of all.