Submitted by: - Nikita Bafna

Assignment2 : Eval trec results

Short Query Results:

Evaluation Metrix	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	0.2820
P@20	0.1350	0.2600	0.2710	0.2890	0.2440
P@100	0.0980	0.1648	0.1678	0.1692	0.1606
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	0.3470	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	0.3325	0.2888
NDCG@100	0.1919	0.3213	0.3259	0.3313	0.3121

Long Query Results:

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	0.2840	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

Summarization:

1) **Your Algorithm:**
   - In Task 1 and Task 2, we are using traditional method to calculate the relevance score of a query in a document using the formula:

     *Relevance_Score = tf * idf where,*

     tf = term frequency

     idf = inverse document frequency

   - Term frequency indicates how often a term occurs in the document.

- Inverse document frequency indicates how many documents a term appears in and if it is a unique term or repeating word.
- We have used Normalization technique which gives importance of query according to the size of document.
- Our algorithm doesn't seem to fare well compared to other 3.

2) Vector Space Model:

- We have used Classic Similarity algorithm which is the default scoring implementation method.
- Each document is represented as real valued vector of tf idf weights.
- A cosine similarity is used to calculate similarity between query and document vector.
- Here it encodes norm values as a single byte before being stored.
- We have calculated the 'normvalues' which gives importance of query according to the size of document. At search time, the norm byte value is read from the index directory and decoded back to a float norm value.
- This encoding/decoding, while reducing index size, comes with the price of precision loss. However, it is efficient with the memory at search time as once a field is referenced, its norms - for all documents - are maintained in memory.

3) BM25 similarity:

- BM25 stands for Best Match 25 algorithm.
- The idf calculation of BM25 is similar to Classic similarity.
- BM25's IDF has the potential for giving negative scores for terms with very high document frequency. So IDF in Lucene's BM25 does this one amazing trick to solve this problem. They add 1 to the value, before taking the log, which makes it impossible to compute a negative value.
- BM25 has its roots in probabilistic information retrieval. Basically, it casts relevance as a probability problem. A relevance score, according to probabilistic information retrieval, ought to reflect the probability a user will consider the result relevant.
- For short as well as long queries, BM25 works as the best model.

4) Language model using Dirichlet smoothing:

- This is a generative model which considers probability distributions of string over text.
- Smoothing is used to calculate probabilities of unseen or missing words in training language model.
- The smoothing factor depends on the length of document.
- It used likelihood method to calculate similarity.
- It works better for short queries compared to Jelinek Mercer smoothing.

5) Language model using Jelinek Mercer smoothing:

- This is also a generative model which considers probability distributions of string over text.
- In this case, we use fixed smoothing factor of 0.7.
- It works better for long queries than Dirichlet smoothing.

References:

1. https://lucene.apache.org/core/5_4_0/core/org/apache/lucene/search/similarities/ClassicSimilarity.html
2. https://opensourceconnections.com/blog/2015/10/16/bm25-the-next-generation-of-lucene-relevation/