Yunlu Li
CS 4780
MP3

Question 1: Copy and paste your implementation of each ranking algorithm, together with the corresponding final MAP/P@10/MRR/NDCG@10 performance you get from each ranking function.

1. Boolean Dot Product

```
protected float score(BasicStats stats, float termFreq, float docLength) {
    return 1;
}
```

2. TF-IDF Dot Product

```
protected float score(BasicStats stats, float termFreq, float docLength) {
    return (float) ((1 + Math.log10(termFreq))*Math.log10((stats.getNumberOfDocuments()+1)/stats.getDocFreq()));
}
```

3. Okapi BM25

```
protected float score(BasicStats stats, float termFreq, float docLength) {
    double k1=1.5;
    double k2=750;
    double b=1.0;
    float firstTerm =  (float) Math.log((stats.getNumberOfDocuments()-stats.getDocFreq()+0.5)/(stats.getDocFreq()+0.5));
    float secondTerm = (float) (((k1+1)*termFreq)/(k1*(1-b+b*(docLength/stats.getAvgFieldLength()))+termFreq));
    //Term Frequency in query is assumed to be one so the last term is simplified to be (k2+1)*1/(k2+1)=1
    float thirdTerm = (float) ((k2+1)*1/(k2+1));
    return firstTerm*secondTerm*thirdTerm;
}
```

4. Pivoted Length Normalization

```
protected float score(BasicStats stats, float termFreq, float docLength) {
    double s=0.75;
    float firstTerm = (float)((1+Math.log(1+Math.log(termFreq)))/(1-s+s*(docLength/stats.getAvgFieldLength())));
    float secondTerm = 1; // because term frequency in query is assumed to be 1
    float thirdTerm = (float) (Math.log((stats.getNumberOfDocuments()+1)/stats.getDocFreq()));
    return firstTerm*secondTerm*thirdTerm;
}
```

5. Jelinek-Mercer LM

```
protected float score(BasicStats stats, float termFreq, float docLength) {
    float lambda = (float) 0.1;
    float alpha = lambda;
    float smoothing = (float)((1 - lambda) * (termFreq/docLength) + lambda * model.computeProbability(stats));
    float score = (float) (Math.log(smoothing/(alpha*model.computeProbability(stats)))+ Math.log(alpha)*queryLength);
    return score;
}
```

6. Dirichlet Prior LM

```
protected float score(BasicStats stats, float termFreq, float docLength) {
    float u = 2500;
    float alpha = u/(u + docLength);
    float smoothing = (float) ((termFreq+u*model.computeProbability(stats))/(docLength+u));
    float score = (float) (Math.log(smoothing/(alpha*model.computeProbability(stats)))+ Math.log(alpha)*queryLength);
    return score;
}
```

The following results are achieved by setting *defaultNumResults* = 1000

	MAP	P@10	MRR	NDCG
BDP	0.2247326822947187	0.2881720430107527	0.5949318511875813	0.3498102343135518
TF-IDF	0.2737293951670014	0.35806451612903223	0.6863789237900358	0.42432380360103067
BM25	0.23314194908914251	0.30752688172043013	0.5937579327977649	0.3681466348357197
PLN	0.16821988454438003	0.23978494623655922	0.4358765510730296	0.27464230861257105
JM LM	0.277298608266259	0.34193548387096767	0.6784570463143281	0.4201580145765151
DP LM	0.15983300184797877	0.1946236559139785	0.47029024116346074	0.24092906541224793

Question 2: Please carefully tune the parameters in BM25 and Dirichlet prior smoothed Language Model. Report the best MAP you have achieved and corresponding parameter settings.

For all following results, the *defaultNumResults* is set to be 1000.

For BM25, the optimal parameter is k1=1.2 and b=0.75. Since we assume the term frequency in query is 1, so we have $\frac{(k_2+1)*1}{(k_2+1)} = 1$. This means parameter k2 does not affect scoring, we leave k2 as the default value 750. The optimal parameter combination of k1=1.2, k2=750 and b=0.75 gives best MAP=0.2781581045361776.

For Dirichlet Prior, the MAP increases as parameter u drops. It achieves best MAP=0.17968278559179396 when u=600.

Question 3: With the default document analyzer, choose one or two queries, where Pivoted Length Normalization model performed significantly better than BM25 model in average precision, and analyze what is the major reason for such improvement? Perform the same analysis for Pivoted Length Normalization v.s. Dirichlet Prior smoothed Language Model, and BM25 v.s. Dirichlet Prior smoothed Language Model, and report your corresponding analysis (using your best parameters for BM25 and Dirichlet Prior smoothed Language Model).

PLN vs. BM25
Query: the synthesis of networks with given sampled data transfer functions
PLN Average Precision: 0.1923438669612841
BM25 Average Precision: 0.10877491801532908

If we look at the doc lists returned by two functions, they both returned doc 7697 as the top one which is irrelevant. The first relevant doc 4412 is ranked as the second by PLN but as the sixth by BM25. The second relevant doc 6138 is ranked as the third by PLN but as the twelfth by BM25. I found the matched term appear once in both docs, so c(w, d)=1. By using the best parameters for BM25 and s=0.75 for PLN, we can simplify the scoring functions and notice the difference is actually caused by the document length. PLN favors docs whose length is less than average whereas BM25 favors docs whose length is larger than average. Comparison for doc length in top positions returned by PLN and BM25 supports my point. The length of doc returned by PLN in second and third place are much shorter than those returned by BM25 in second and third place. In this case, those longer docs returned by BM25 are not relevant, and this leads to the significant difference between BM25 and PLN.

PLN vs. Dirichlet Prior
Query: characteristics of the single electrode discharge in the rare gases at low pressures
PLN Average Precision: 0.5238095238095238
Dirichlet Prior Average Precision: 0.13425925925925924

The number of relevant docs for this query is small, so the difference of average precision is mainly caused by where each ranking model returns the first relevant doc. PLN returned the first relevant doc 7014 at the first position, while DP LM returned the doc 7014 at the third position. PLN gives doc 7014 highest scoring because nearly all query terms appeared in the doc 7014 and some important terms like "discharge" appeared more than once. However, DP LM mistakenly ranked docs 4754 and 5300 at the top two position. Carefully analyzing the content of these two docs reveals that they are not talking about "characteristics". This is because the likelihood of generating terms like 'electrode', 'discharge' and 'pressures' are high in the corpus that gives irrelevant documents higher priority in DP LM but missed the important query term 'characteristics'. From our tuning process for DP LM, we found better performance is associated with lower u values. This suggests the corpus is not a good enough representative for generating queries given to evaluate.

BM25 vs. Dirichlet Prior
Query: spherical harmonic analysis of the earth's magnetic field
BM25 Average Precision: 0.4133336592126367
Dirichlet Prior Average Precision: 0.14739070495339243

When looking the returned doc lists, I found that difference mainly comes from top five positions. BM25 returned relevant docs at all top five position whereas DP LM mistakenly returned irrelevant docs 981 and 10079 at position two and five. In analyzing the contents of these two docs, I found that terms 'magnetic' and 'harmonic' are not included. I believe this comes from the same reason as we mentioned above. The likelihood of generating terms like 'analysis' and 'spherical' are high in the corpus that gives irrelevant documents higher priority in DP LM but missed the important query terms like 'magnetic' and 'harmonic'. Since better performance is associated with lower u values, we believe the corpus is not a good enough representative for generating queries given to evaluate.

Question 4: Pick one of the previously implemented scoring functions to analyze under what circumstance the chosen scoring function will mistakenly favor some less relevant document Please correspond your analysis with what you have found in Problem 3. After reading the paper An Exploration of Axiomatic Approaches to Information Retrieval, 1) can you briefly summarize the major contribution of this paper? 2) how do you think you can fix the problem you have identified in the ranking result analysis? Please relate your solution and corresponding implementation in the report. Also report the resulting ranking performance of your revised ranking algorithm.

Based on our analysis above, I found that BM25 favors documents which have longer length and multiple matched terms occurrence. However, this may lead to wrong ranking when important terms in query do not appear in document but unimportant terms appear multiple times. For example, under query "the phenomenon of radiation caused by charged particles moving in varying electric and magnetic fields", BM25 ranks documents with multiple occurrences of terms like 'magnetic' and 'charged' and longer length at top positions, but these documents are not relevant since they mentioned neither 'radiation' nor 'phenomenon'.

The major contribution of the paper is to develop a new axiomatic framework for information retrieval model. It defines a set of reasonable retrieval constraints. By searching for candidate retrieval functions that can satisfy all constraints, we can capture relevance directly and this leads to the derivation of new retrieval functions which achieve more stable performance.

Based on what suggested in this paper, I revised the weight term to reflect IDF and replace two parameter k_1 and b with one parameter s. The new score function is

$$r(q,d) = \sum_{t \in D \cap Q} \ln\left(\frac{N+1}{df}\right) * \frac{c(w;d)}{s\frac{n}{n_{avg}} + s + c(w;d)} * c(w;q)$$

where we assume c(w;q)=1 as before and set s =0.5. The implementation is

```
protected float score(BasicStats stats, float termFreq, float docLength) {
    //double k1=1.2;
    //double k2=750;
    //double b=0.75;
    double s = 0.5;
    //float firstTerm =  (float) Math.log((stats.getNumberOfDocuments()-stats.getDocFreq()+0.5)/(stats.getDocFreq()+0.5));
    float firstTerm =  (float) Math.log((stats.getNumberOfDocuments()+1.0)/stats.getDocFreq());
    //float secondTerm = (float) (((k1+1)*termFreq)/(k1*(1-b+b*(docLength/stats.getAvgFieldLength()))+termFreq));
    float secondTerm = (float) ((termFreq)/((s+s*(docLength/stats.getAvgFieldLength()))+termFreq));
    //Term Frequency in query is assumed to be one so the last term is simplified to be (k2+1)*1/(k2+1)=1
    //float thirdTerm = (float) ((k2+1)*1/(k2+1));
    float thirdTerm = (float) 1.0;
    return firstTerm*secondTerm*thirdTerm;
}
```

For the specific query above, the average precision increases from 0.072289465 to 0.113602453. The overall four metrics increase significantly, as shown below.

	Old BM25	New BM25
MAP	0.23314194908914251	0.2884233647689009
P@10	0.30752688172043013	0.3666666666666668
MRR	0.5937579327977649	0.6929403145998692
NDCG	0.3681466348357197	0.44259697217218474