RETRIEVAL EVALUATION

Information Retrieval Machine Problem #2
By Muhammad Nur Yanhaona
UVA ID: mny9md

OBJECTIVE:

The objective of this homework is to evaluate and reason about performance of several standard retrieval functions in Lucene Toolkit. The experiment is done on a small dataset of physics paper titles that comes along with the instructor provided project. Almost all of the code for retrieval testing was already given by the instructor. We only had to implement the scoring functions for different retrieval methods and tune corresponding parameters to optimize performance.

IMPLEMENTATION OF RETRIEVAL METHODS:

BOOLEAN DOT PRODUCT

```
protected float score(BasicStats stats, float termFreq, float docLength) {
        return (termFreq > 0) ? 1 : 0;
}
```

Mean Average Precision: 0.20682027649769577

TF-IDF DOT PRODUCT

```
protected float score(BasicStats stats, float termFreq, float docLength) {
        long df = stats.getDocFreq();
        long N = stats.getNumberOfDocuments();
        return (float) (termFreq * Math.log((N + 1) / df));
}
```

Mean Average Precision: 0.1381063321385902

OKAPI BM25

```
protected float score(BasicStats stats, float termFreq, float docLength) {
        float df = stats.getDocFreq();
        float nAvg = stats.getAvgFieldLength();
        long N = stats.getNumberOfDocuments();
        float queryTermFreq = 1.0f;
        float docRelevance = (float) Math.log((N - df + 0.5) / (df + 0.5));
        float termFrequencyWeight = ((k1 + 1) * termFreq)
                        / (k1 * (1 - b + b * docLength / nAvg) + termFreq);
        float queryTermFrequencyWeight = (k2 + 1) * queryTermFreq / (k2 + queryTermFreq);
        return docRelevance * termFrequencyWeight * queryTermFrequencyWeight;
}
```

Mean average precision for b: 0.75 and k_1: 2.0: 0.24052782044717538

PIVOTED LENGTH NORMALIZATION

```
protected float score(BasicStats stats, float termFreq, float docLength) {
        float nAvg = stats.getAvgFieldLength();
        long N = stats.getNumberOfDocuments();
        float df = stats.getDocFreq();
        float queryTermFreq = 1;
        double idf = Math.log((N + 1) / df);
        double relevance = (1 + Math.log(1 + Math.log(termFreq))) / (1 - s + s * docLength / nAvg);
        return (float) (relevance * queryTermFreq * idf);
}
```

Mean Average Precision: 0.26501706775900324

JELINEK-MERCER

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

-2

Mean Average Precision for λ = 0.1: 0.249989332650623

Sorry, this is exact implementation since $|q| \log \lambda$ is constant for all the docs

Here the implementation is only an approximation of the actual retrieval function which has a term **|qeryLength|log λ** being added to the total scores of individual term matching. This approach is taken as the score function is called for individual terms that are both in the query and in the document currently under examination. Thus there is no easy way to apply a quantity to overall document ranking.

DIRICHLET PRIOR

```
protected float score(BasicStats stats, float termFreq, float docLength) {
        float p = model.computeProbability(stats);
        float pSeen = (termFreq + mue * p) / (docLength + mue);
        float alpha_d = mue / (mue + docLength);
        return (float) (Math.log(pSeen / (alpha_d * p)) + Math.log(alpha_d));
}
```

Mean Average Precision for μ = 2000: MAP: 0.16063321385902035

CALCULATION OF A_D IN LANGUAGE MODELS

The equation for α_d is as follows

$\alpha_d = (1 - \text{sum}_{w\text{-seen}}(P_{seen}(w|D))) / \text{sum}_{w\text{-unseen}}P(w|D)$

So derivation of the exact value for this in Jelinek-Mercer is

$$\begin{aligned}\alpha_d &= (1 - \text{sum}_{w\text{-seen}}(P_{seen}(w|D))) / \text{sum}_{w\text{-unseen}}P(w|D)\\ &= (1 - \text{sum}_{w\text{-seen}}((1-\lambda)P(w|D) + \lambda P(w|C))) / \text{sum}_{w\text{-unseen}}P(w|D)\\ &= (1 - \text{sum}_{w\text{-seen}}P(w|D) + \lambda\text{sum}_{w\text{-seen}}P(w|D) - \lambda\text{sum}_{w\text{-seen}}P(w|C)) / \text{sum}_{w\text{-unseen}}P(w|D)\end{aligned}$$

$= (1 – 1 + \lambda - \lambda \text{sum}_{\text{w-seen}} P(w|C)) / \text{sum}_{\text{w-unseen}} P(w|D)$
$= \lambda(1 – \text{sum}_{\text{w-seen}} P(w|C)) / \text{sum}_{\text{w-unseen}} P(w|D)$
$= \lambda$

The derivation of the exact value of α_d in Dirichlet Prior is as follows

$\alpha_d = (1 – \text{sum}_{\text{w-seen}}(P_{\text{seen}}(w|D))) / \text{sum}_{\text{w-unseen}} P(w|D)$
$= (1 - \text{sum}_{\text{w-seen}}(c(w, d) + \mu P(w|C))/(n + \mu)) / \text{sum}_{\text{w-unseen}} P(w|D)$
$= (1 – (1 / (n + \mu))(\text{sum}_{\text{w-seen}} c(w, d) + \mu \text{sum}_{\text{w-seen}} P(w|C))) / \text{sum}_{\text{w-unseen}} P(w|D)$
$= (n + \mu – (\text{sum}_{\text{w-seen}} c(w, d) + \mu \text{sum}_{\text{w-seen}} P(w|C))) / (n + \mu)\text{sum}_{\text{w-unseen}} P(w|D)$
$= (n + \mu – (n + \mu \text{sum}_{\text{w-seen}} P(w|C))) / (n + \mu)\text{sum}_{\text{w-unseen}} P(w|D)$
$= \mu (1 – \text{sum}_{\text{w-seen}} P(w|C)) / (n + \mu)\text{sum}_{\text{w-unseen}} P(w|D)$
$= \mu / (n + \mu)$

Parameter Tuning in Dirichlet Prior and Okapi BM25

Okapi BM25 scoring function takes three parameters k_1, k_2, and b for tuning the weights of term frequency, query term frequency, and document length respectively in the scoring process. As we assumed that the frequency of each term in the query is 1, k_2 becomes ineffective. I varied the values of k_1 and b within the parameter ranges [1.2, 2] and [0.75, 1.2] respectively. The best result is received for $k_1 = 1.2$ and b = 0.75 and corresponding mean average precision is 0.26156255333674694. Increasing either k_1 or b from their minimum value decreases retrieval performance. If we treat both parameters independently, we realize that increasing k1 has an effect of increasing the effect of term frequency in the scoring method and increasing b has the effect of emphasizing document length. It is difficult to determine the ideal values for these parameters. Nonetheless, I suspect performance worsening with increased values is a sign of overcorrection. On the other hand, if I go beyond the limits and lower the value of k_1 and b to 1 and 0.5 then mean average precision improves to ~0.277368. Thus the appropriate choice of these parameters seems to be difficult.

Dirichlet Prior scoring function has a single parameter, μ. We are asked to try values around 2000 to 3000 for that. Just like the previous case, increasing μ only decrease retrieval performance. The best mean average precision within the specified limit is for μ = 2000 and 0.16063321385902035. If we go below from 2000 to 1000, 500, and 250 then mean average precision improves to 0.185915685270524, 0.206326591568527, and 0.22921744324970125 respectively. Given that μ is a measure of likelihood of the corpus for generating terms in the query, better performance with lower values suggests the corpus is not a good enough representative for generating queries we are given to evaluate.

Evaluating Effects of Different Filters on Okapi BM25

We are given a pipeline of filters that process document and queries to improve retrieval effectiveness. The filters are

1. Standard Filter: does token normalization
2. Lower Case Filter: self explanatory
3. Length Filter: filters out too short or long tokens
4. Stop Filter: removes stop words
5. Porter-Stem Filter: does stemming

I got the following results

- Just Standard Filter MAP: 0.07853601297149686
- Standard + Lower Case Filters MAP: 0.07853601297149686
- Standard + Lower Case + Length Filters MAP: 0.0785509472606247
- Standard + Lower Case + Length + Stop Filters MAP: 0.08684545144222565
- Standard + Lower Case + Length + Stop + Porter-Stem Filters MAP: 0.26156255333674694
- Lower Case + Porter-Stem Filters MAP: 0.2578498890595665

As we can see from the above, retrieval performance is greatly affected by stemming. Stop word removal has some additional benefits – much less than stemming though. The other filters seem to be mostly inconsequential. My intuition suggests that at least lower case filter alone should have significant import in retrieval performance, but the results indicate otherwise. I believe this happens because most of the words in the document and queries are already in lower case.

COMPARISONS BETWEEN SELECTED RETRIEVAL FUNCTIONS FOR SPECIFIC QUERIES

To do this part of the homework, I modified the main function in Evaluate.java class. The new main function considers two retrieval functions simultaneously as it goes through the queries in npl-judgment.txt file. The query with maximum difference in average precisions of the two functions is recorded. Then I used the Runner.java class to see the results returned for the recorded query and reason about the performance difference. So this is a subjective reasoning method and I may have deductions that are completely wrong.

COMPARISON BETWEEN TF-IDF AND BOOLEAN DOT PRODUCT

The best performance of TF-IDF against Boolean Dot Product can be describes as follows
TF-IDF average precision: 0.4; BDP average precision: 0.10666666666666666
Difference in precision: 0.29333333333333333
Query: high frequency oscillators using transistors theoretical treatment and practical circuit details

After examining the outputs for both functions for this query, it seems to me that in TF-IDF terms like 'practical' and 'details' have much less significance than they do in BDP. As a result, BDF returns many results that do not address 'high frequency oscillation' or 'transistor.' Probably in TF-IDF the latter have more significance due to consideration of inverse document frequency.

COMPARISON BETWEEN TF-IDF AND BM25 ($k_1 = 1.2$, $b = 0.75$)

The best performance of TF-IDF against Okapi BM25 can be described as follows
TF-IDF average precision: 0.4; Okapi BM25 average precision: 0.06928571428571428
Difference in precision: 0.33071428571428574
Query: high frequency oscillators using transistors theoretical treatment and practical circuit details

We see that Okapi BM25 perform bade for the same query. Unlike in the case of BDP, it is much difficult to reason about BM25's performance inefficiency. As we have seen in the performance tuning case, reducing the values of both parameters beyond the instructor specified lower limit continues to give better and better retrieval performance. So my assumption is that – despite having term frequency, inverse document frequency, and document length – all in the scoring method, BM25 performs badly in this case because of suboptimal parameter settings.

Once I check the best BM25 performance against TF-IDF reasoning becomes much easier. The result is as follows

Okapi BM25 average precision: 0.7577777777777778; TF-IDF average precision: 0.05833333333333333
Difference in precision: 0.6994444444444444
Query: solution of differential equations by computer

Closer examination of the results returned by the two retrieval functions for this query shows that BM25 returns several results having both ‘differential equation’ and ‘computer’ in them while TF-IDF produces several results that have only ‘computer.’ This suggests that terms like ‘computer’ may be over-weighted in TF-IDF that results in its dismal performance.

Difference Between Okapi BM25 and Dirichlet Prior

The best performance difference of BM25 against Dirichlet Prior smoothing happens under the following circumstances.

Okapi BM25 average precision: 0.645; Dirichlet Prior: 0.16499999999999998
Difference in precision: 0.48000000000000004
Query: spherical harmonic analysis of the earth’s magnetic field

There is an implementation problem in Dirichlet Prior method (that we discussed before). That might have contributed to its bad performance. My intuition, however, suggests that the use of document frequency in BM25 instead of total term frequency as in Dirichlet Prior has significant import for this query. Furthermore, comparatively poor mean average precision of Dirchlet Prior suggests that the corpus is not a good representative set for generating the kind of queries used for performance judgment.

Examination of some results returned by the two functions bolsters my intuition. For example the first result for Okapi BM25 is as follows

... rocket measurements of the ****magnetic**** ****field**** above new mexico absolute total intensity was measured in a rocket at heights up to and compared with the intensity predicted by ****spherical**** ****harmonic**** coefficients ...

While the first two results for Dirichlet Prior are

... the possibility of electron total energy distribution ****analysis**** in a quasi ****spherical**** capacitor the energy distribution of electrons emitted at various angles from a plane disk shaped target at the centre of a ****spherical**** collector is determined from an ****analysis**** of electron trajectories the discrepancy between these and those for a ****spherical**** capacitor does not exceed one per cent the provision...

... or ****spherical**** wave are dealt with the incident wave is represented by a line source acoustic or electromagnetic parallel to the edge the ****spherical**** wave is emitted by an acoustic point source or by a hertz dipole with its axis parallel to the edge the case of an incident plane wave ****field**** is obtained ... or ****spherical**** wave excitation ...

The terms ‘magnetic’ and ‘harmonic’ are not in the results. Probably the likelihood of generating terms like ‘spherical’ and ‘analysis’ are high in the corpus that gives less relevant documents higher priority in the retrieval process for Dirichlet Prior method.

Finally if we go outside the suggested range for μ and use 250 as its value, a different query gives the best performance difference as described below.

Okapi BM25 average precision: 0.48333333333333333; Dirichlet Prior: 0.09833333333333334
Difference in average precision: 0.38499999999999995
Query: observations of the sun during eclipses giving the distribution of sources on the disc in the microwave range

In conclusion, I believe determining the best retrieval strategy is an art, or at the very least not an exact science. The decision should depend on the quality and size of the reference corpus and user feedbacks.