CS 6501: ADVANCED TOPICS IN INFORMATION RETRIEVAL

MP—Retrieval Functions and Evaluations

In this assignment, you will be implementing a complete retrieval pipeline with the Lucene toolkit, with a special focus on document ranking and search evaluations. We are aware that some of the functions already exist in Lucene, but we'd like you to add them yourself for this assignment. This assignment is required to be finished by students individually.

The assignment is composed of the following parts consisting of **100 total points**:

- **Overview of Retrieval Models**
  - Boolean model
  - Vector space models: TF-IDF dot product, Okapi BM25 and pivoted length normalization
  - Language models: Jelinek-Mercer and Dirichlet prior smoothing
- **Scoring functions in Lucene** *(30 points)*
- **Running Lucene**
- **Evaluation functions** *(20 points)*
  - Precision at k
  - Mean Average Precision
  - Mean Reciprocal Rank
  - Normalized Discounted Cumulative Gain
- **Questions** *(50 points)*

The 30 points in the scoring functions and 20 points in evaluation functions are for coding completion -- it has to be done in the correct way, and you need to explain your implementation in your report (paste the modified code section).

The 50 points are for correct, justified answers to the questions at the end, and reasonable search performance generated by your implementations.

Please download the provided project **here**, inside which you will find all the necessary files. *You have to use the provided Lucene library, because the latest version of Lucene is not compatible with our provided sample implementation.*

In the next section, we'll give a brief overview of the six retrieval functions you'll need to implement.

Boolean Models

$$r(q,d) = \sum_{w \in q \cap d} 1$$

Vector Space Models

TF-IDF dot product

$$r(q,d) = \sum_{w \in q \cap d} \left(1 + \log c(w,d)\right) \cdot \log\left(\frac{N+1}{df}\right)$$

where

- $c(w;d)$ is the count of word w in the document d
- N is the total number of documents, and
- df is the document frequency.

Okapi BM25

Parameters: $k_1 \in [1.2, 2]$, $k_2 \in (0, 1000]$, $b \in [0.75, 1.2]$.

$$r(q,d) = \sum_{w \in q \cap d} \ln\left(\frac{N - df + 0.5}{df + 0.5}\right) \cdot \frac{(k_1 + 1) \cdot c(w;d)}{k_1(1 - b + b\frac{n}{n_{avg}}) + c(w;d)} \cdot \frac{(k_2 + 1) \cdot c(w;q)}{k_2 + c(w;q)}$$

where

- $c(w;q)$ is the count of word w in query q
- n is the document length, and
- n_{avg} is the average document length.

Pivoted Length Normalization

Parameter: $s \in [0, 1]$.

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

This is another version of TF normalization, which we did not cover in our course lecture. The notations follow those in Okapi BM25.

Language Models

As we have discussed in class the language models rank documents according to query likelihood:

$$r(q,d) = \sum_{w \in q} \log p(w|d)$$

After proper smoothing, the scoring functions for Language Models become,

$$r(q,d) = \sum_{w \in q \cap d} \log \frac{p_s(w|d)}{\alpha_d p(w|C)} + |q| \log \alpha_d$$

In the following two language model retrieval functions, we define different smoothing strategies. You can then plug these two smoothed document language models into the general language model formula above, and we will only use unigram language models.

Jelinek-Mercer Smoothing

Parameter: $\lambda \in [0, 1]$.

$$p_s(w|d) = (1 - \lambda) p_{ml}(w|d) + \lambda p(w|C)$$

where p_{ml} is the maximum likelihood estimation. This is also called linear interpolation smoothing, accordingly $\alpha_d = \lambda$.

Dirichlet Prior Smoothing

Parameter: $\mu > 0$. Emperically value for μ lies in the range of 2000 to 3000.

$$p_s(w|d) = \frac{c(w;d) + \mu p(w|C)}{n + \mu}$$

Accordingly $\alpha_d = \frac{\mu}{\mu + n}$

Scoring Functions in Lucene

In Lucene, all the retrieval functions have the following function signature to score an individual query term against a given document:

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

This would be equivalent to one term in each sum above; this function is called once per word in the query for each document **where that word occurs**. Once all the documents are scored, Lucene outputs a list of documents ranked by their score.

The `BasicStats` object has the following functions that will be useful:

- `getAvgFieldLength()` : average document length
- `getNumberOfDocuments()` : total number of documents in the index
- `getDocFreq()` : the number of documents the current term appears in

For the language models, you will need the additional functionality of the member variable `model`, which is of type `LMSimilarity.DefaultCollectionModel`. It has the following function that will be of use:

- `computeProbability(stats)` : computes $p(w|C)$ taking the `BasicStats` object described above as a paramter

For the language models, also note:

- To compute $p_{ml}(w|d)$, you can use two existing variables
- There is a member variable `queryLength` that you can use for the value of $|q|$

Your task is to complete the `score` function for each of the six retrieval models listed above. All the retrieval models are located in the package `edu.illinois.cs.index.similarities`.

Implementation Details

- If there are parameters in the scoring function (e.g., as in BM25), it's probably easiest to add these as member variables (e.g., as a constant)
- You can use any logical values for parameters that you'd like
- You may assume that the queries are short -- that is, **you may assume that the query term frequency is always one**. This simplifies your code a bit.
- Default values in the ranking functions: 1) in Okapi BM25, set $k_1 = 1.5$, $k_2 = 750$ and $b = 1.0$; 2) in Pivoted Length Normalization, set $s = 0.75$; 3) in Jelinek-Mercer smoothing,

set $\lambda = 0.1$; and 4) in Dirichlet Prior smoothing, set $\mu = 2500$.

Running Lucene

Creating an index

There is a small data set distributed with this assignment. It is the NPL dataset of physics paper titles located in the `data/` folder in the root of the project.

Two different main functions are provided in the `edu.illinois.cs.index.Runner.java` file. You can read the comments and decide which one you will use.

Searching the index

Two different main functions are provided in the `edu.illinois.cs.index.Runner.java` file for you to interactively search the index. You should read the comments and decide which one you will use.

Keep in mind the documents you're searching are physics paper titles. You can also specify which retrieval function to use when starting the search engine.

The complete list of options is

```
--dp     Dirichlet Prior
--jm     Jelinek-Mercer
--ok     Okapi BM25
--pl     Pivoted Length Normalization
--tfidf  TFIDF Dot Product
--bdp    Boolean Dot Product
```

Search Evaluation Functions

You need to implement the required search evaluation functions in `edu.illinois.cs.eval.Evaluate.java`. A sample implementation of Mean Average

Precision has been given in this file, but you need to complete it according to the comments.

Base on this sample implementation, you need to implement Precision at k (P@K), Mean Reciprocal Rank (MRR) and Normalized Discounted Cumulative Gain (NDCG).

In this assignment, we will only evaluate Precision at 10, i.e., P@10, and NDCG@10. And we will implement DCG according to the following formula,

$$DCG@k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{\log_2(1+i)}$$

Although we only provided binary evaluation in this assignment, NDCG can still be applied.

To test your retrieval functions, you should use the main functions provided in `edu.illinois.cs.eval.Evaluate.java`. Notice the option to use a specific retrieval function. That is the same option as in the interactive search engine.

For each query, the corresponding retrieval performance is printed. Once you implement the required ranking functions, you should get at least a MAP of 0.10 for each one. Some will be better than others. *Hint: the language model-based methods did not perform very well on this dataset (but you can still get them over 0.10). We had the two good vector space models get around 0.25.*

Questions

1. Copy and paste your implementation of each ranking algorithm and evaluation function into your report, together with the corresponding final MAP/P@10/MRR/NDCG@10 performance you get from each ranking function. *Use the default parameter settings suggested here* (30pts + 20pts)

2. Please carefully tune the parameters in BM25 and Dirichlet prior smoothed Language Model. Report the best MAP you have achieved and correposnding parameter settings. (15pts)

3. In `edu.illinois.cs.index.SpecialAnalyzer.java`, we defined a special document analyzer to process the document/query for retrieval purpose. Basically, we built up a pipeline with filters of `LowerCaseFilter`, `LengthFilter`, `StopFilter`, and `PorterStemFilter`. Please disable some of the filters, e.g., without stopword removal or stemming, and test the new analyzer with the BM25 model (with your best parameters). What is your conclusion about the effect of document analyzer on retrieval effectiveness? (15pts) *Note: this analyzer has to be used in both indexing time and query time!*

4. With the default document analyzer, choose one or two queries, where TF-IDF dot-product model performed significantly better than Boolean dot-product model in average precision, and analysis what is the major reason for such improvement? Perform the same analysis for TF-IDF dot-product model v.s. BM25, 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). (10pts)

5. Pick one of the previously implemented scoring functions out of

    - Okapi BM25
    - Pivoted Length Normalization
    - Langauge Model with Dirichlet Smoothing
      to analyze under what circumstance the chosen scoring function will mistakenly favor some less relevant document (*i.e.*, ranks a less relevant document than a more relevant one).

After reading the paper An Exploration of Axiomatic Approaches to Information Retrieval, how do you think you can fix the problem? Please relate your solution and corresponding implementation in the report. Also report the corresponding ranking performance of your revised ranking algorithm. (10pts)

Deadlines & How to submit

This assignment is due on **April 14th 11:55pm**. Therefore, you have in total 15 days to finish this MP. The late policy for all our homework has been carefully discussed in

the course syllabus.

The collab assignment page has been created for this MP. Please submit your written report strictly following the requirement specified above. The report **must be in PDF** format.

Updated March 29, 2017

HCDM Lab **Department of Computer Science** **University of Virginia**