CSCE-470 Programming Assignment #1 Ranking models VSM and BM25F

Due: Sunday, Sep. 22th, 2024 by 11:59pm

In this programming assignment, you will devise ranking functions to rank some given documents with respect to a query. The ranking functions will calculate a relevance value for each document wrt. a query. For each query-document pair, you are provided with several features that will help you rank the documents. You are also provided with a training set consisting of query-document pairs along with their relevance values, so that you can manually tune parameters (i.e., no machine learning) based on the training set. You will be implementing both zone-aware VSM and BM25F ranking functions. You will measure the effectiveness of a ranking function with a tuned set of parameters by applying it to rank documents for a new set (the development set) of queries. You will use the NDCG metric (provided in the starter code) to score your document ranking results.

NOTE: The included README.md file provides instructions on the initial steps and explanations of the starter code structure.

DATA

We have partitioned the data into two sets for you:

1. Training set of 247 queries ((signal|rel).train)
2. Development set of 51 queries ((signal|rel).dev)

The idea is that while tuning and maximizing performance on the training set, you should also verify how well the tuned parameters are doing on the separate development set to ensure you are not overfitting your model. There is a hidden test set of 50 queries which we have reserved to grade your final model. For each set, there are two types of files:

1. **Signal File** signal.(train|dev): lists queries along with documents returned by a widely used search engine for each individual query (the list of documents is shuffled and is not in the same order as returned by the search engine). Each query has 10 or less documents. For each document (d) of a query (q), the file contains the url, document title, title length, body hits, body length and the pagerank value of the document. This pattern repeats for the next document until all of the documents for a query are done and then the overall pattern repeats for the next query.

- The title length line states how many terms are present in the title of the document.
- The body hits line specifies the query term followed by the positional postings list of that term in the document (sorted in increasing order).
- The body length line states how many terms are present in the body of the document.
- The *pagerank* is an integer from 0 to 9 that signifies a query-independent quality of the page (the higher the *PageRank*, the better the quality of the page).

2. **Relevance File** rel.(train|dev): lists the relevance judgments for each of the query-document pairs in the corresponding signal file. The relevance value ranges from 0 to 3, with a higher value indicating that the document is more relevant to that query. The file starts with a query and a set of url lines. Each url line can be broken into the document url and the relevance judgment for the query-document pair. This pattern repeats for the next query until all of the queries in the file are done.

Important: The ranking functions also require certain collection-wide statistics (such as inverse document frequency) and we cannot infer this information just from the relatively small training set itself. As a result, you will first need to index a large text corpus in Lucene to derive the above statistics (instructions provided in the other instruction document).

The Evaluation Metric NDCG

NDCG is a standard ranking evaluation metric that assigns a score to your ranked list of documents considering both the predicted ranking order and the human deemed relevance value for each document (document relevance values are in rel.(train|dev) files). Intuitively, the higher position (e.g., the top position) your ranking algorithm predicts for a highly relevant document (with a high relevance value), the more credit you receive. We will study NDCG in lecture time later when we talk about retrieval evaluation.

We will calculate a NDCG based on a ranked list of documents for a query and we will use the NDCG averaged over a set of queries for measuring the effectiveness of a ranking algorithm. The higher the value, the better your ranking algorithm works.

Note, you do not need to do anything with the evaluation metric, and the starter code provides the implementation of the metric for you.

Ranking

Task 1. **VSM (3 Points!)** The first task is to implement a variant of cosine similarity (with the L1-Norm) as the ranking function. This essentially involves constructing the document vector and the query vector and then taking their dot product. Instead of constructing a single document vector, we construct a document vector for each of the two fields, title and body, and we implement zone-aware VSM that first takes dot product between the query vector and each field vector and then takes weighted sum of individual dot product values.

In order to construct the vectors, we need to decide on how we compute a term frequency, a document frequency weighting, and a normalization strategy. Let's discuss these for both the vectors separately.

Document Vector (for the fields Title and Body)

- **Term Frequency** We compute the raw term frequencies for each query term in the different fields and construct tf vectors, one tf vector for each of the fields. We can compute the tf vectors, either using the raw frequencies (rf) themselves or by applying sublinear scaling on the raw frequencies. In sublinear scaling, we have $tf_i = 1 + log(rf_i)$ if $rf_i > 0$ and 0 otherwise.

- **Document Frequency** We will not use any document frequency in the document vector. Instead, it is incorporated in the query vector as described below.
- **Normalization** We cannot use cosine normalization as we do not have access to the contents of the document and, thus, do not know what other terms (and counts of those terms) occur in the body field. As a result, we use length normalization instead. Moreover, since there can be huge discrepancies between the lengths of the different fields, we divide the term frequencies in a field by the *length* of the corresponding *field*.

Query Vector

- **Term Frequency** The raw term frequencies can be computed using the query (should be 1 for most query terms but not necessarily true). Since most terms only appear once in a query, you do not need to consider sublinear scaling here.
- **Document Frequency** Each of the terms in qv should be weighed using the idf value of the term. Computing the idf of a term requires going to a large text corpus (provided in the folder data/corpus) to determine how many documents contain the query term.
- **Normalization** No normalization is needed for query length because any query length normalization applies to all docs and so is not relevant to ranking.

For a document d and query q, if qv_q is the query vector, and $tf_{d,t}$ and $tf_{d,b}$ are the term frequency (field length normalized) vectors for the title and body respectively, then the net score is

$qv_q \cdot (c_t \cdot tf_{d,t} + c_b \cdot tf_{d,b}) \qquad (1)$

Here, c_t and c_b are the weights given to title, and body respectively. The goal is to determine the weights for the two fields (and, thus, the ranking function) so that the NDCG function is of an optimal value when run on the test set. You will use the training set given to derive the above parameters.

Hint: *Note that the absolute values of weights won't matter as they will be the same for all documents, only the relative weights for different fields are important; i.e. you can multiply each weight by a constant and the ranking will remain the same. In order to estimate the relative weights, try to reason the relative importance of the different fields.*

Task 2. BM25F (8 Points!)

The second task is to implement the BM25F ranking algorithm. The algorithm is described in detail in the lecture slides. Specifically, you should take a look at slides 19-21 of the lecture slide file bm25_bm25F.pdf before reading further. Here, instead of using the straightforward field length normalized term frequencies for representing a document as in task 1, we use field-dependent parameterized normalizer to obtain normalized term frequencies (ftf). Thus, for a given term t and field $f \in \{body, title\}$ in document d,

$ftf_{d,f,t} = \frac{tf_{d,f,t}}{(1-B_f)+B_f\cdot(len_{d,f}/avlen_f)}$ (2)

where $tf_{d,f,t}$ is the raw term frequency of t in field f in document d, $len_{d,f}$ is the length of f in d and $avlen_f$ is the average field length for f. The variables $avlen_{body}$ and $avlen_{title}$ can be computed using the training set. B_f is a field-dependent parameter and must be tuned for this task. Then, the overall weight for the term t in document d among all fields is

$w_{d,t} = \sum_f W_f \cdot ftf_{d,f,t}$ (3)

Here, W_f is also a field-dependent parameter that determines the relative weights given to each field. This value is similar in theory to the tuning parameters for Task 1.

Since, we also have a non-textual feature, in the form of *pagerank*, we incorporate it into our ranking function using the method described in the bm25_bm25F lecture regarding ranking with non-textual features (slide 21).

Therefore, the overall score of document d for query q is then:

$\sum_{t\in q} \frac{W_{d,t}}{K_1+W_{d,t}} idf_t + \lambda V_j(f)$ (4)

where K_1 is also a free parameter and V_j can be a log/saturation/sigmoid function as mentioned in the slides (you will need to experiment with the other parameter λ' used by the V_j function). Thus, for this task, there are a minimum of 7 parameters to optimize, namely B_{title}, B_{body}, W_{title}, W_{body}, λ, λ' and K_1. Additionally, you also have to select the V_j function appropriately. While in theory, BM25F should give a better NDCG value as it incorporates a lot of more information, this need not necessarily be the case.

Hint: *The weight values obtained in Task1 may be a good starting point for W_{title}, W_{body} in this task. Again note that the weights will depend on the "importance" of the fields. Moreover, as mentioned in the slides, log(pagerank) works well in practice but you should try other functions as well and see how they work.*

The Report

(4 Points!) Please write up a 1-3 page report describing the design choices undertaken for the two tasks as well as the actual parameter values. Explanations as to why the parameters work as well as the overall effectiveness of the ranking functions must also be mentioned. In particular, you need to address the following questions in the report:

1. **(1 Point!)** Report NDCG on both training and development sets for the two tasks.
2. **(1 Point!)** For the two tasks, you should report all final model parameter values. Describe the intuition when tuning your models, and why those weights work in getting a good score. Were there any particular properties about the documents that allowed a higher weight to be given to one field as opposed to another?
3. **(1 Point!)** In BM25F, in addition to the weights given to the fields, there are 5 other parameters, B_{title}, B_{body}, λ, λ' and K_1. How do these parameters affect the ranking function?

4. **(1 Point!)** In task 1, you may either use raw frequencies or sublinearly scale them to compute term frequency. Please report your choice and the reasons behind them. For BM25F, why did you select a particular V_j function?

Extra Credit

(2 Points!) Credit will be given based on the performance of your ranking algorithms on the held out test set. If the average NDCG score of your two ranking algorithms on the test set is in the top 10 of the class, you earn the extra credit.

GRADING CRITERIA

Your code will be graded mainly based on the correctness. You should tune your parameters using the provided training and dev set. You will receive extra credit if your tuned parameters work super well (in the top 10 again) in the held out test set. Your written report will be graded based on the grade allocations on individual items.

ELECTRONIC SUBMISSION INSTRUCTIONS (a.k.a. "What to turn in and how to tun in")

You only need to submit 2 things:

1. The source code files. While you are only supposed to modify the three scorer files, AScorer.java, BM25Scorer.java and VSMScorer.java, **please compress the whole src folder into a .zip file and submit the .zip file**.

2. The written report in pdf format.

NOTE: Please do **NOT** include the data files or any other file in your submission.

How to turn in:

1. please submit the two files, the source code .zip file and the written report pdf file, via canvas.