COMP 5790/ 6790/ 6796

Special Topics: Information Retrieval

Instructor: Shubhra ("Santu") Karmaker

Assignment #7: Learning to Rank [100 points]

⚠ **Notice:** This assignment is due **Monday, April 13, 2020 at 11:59pm**.

Please submit your solutions via Canvas (https://auburn.instructure.com/). You should submit your assignment as a **typeset PDF**. Please do not include scanned or photographed equations as they are difficult for us to grade.

In this assignment, you will get to play with some popular learning-to-rank techniques. You will conduct some experiments based on the tool "Ranklib" and prepare a corresponding report in the following format.

1. Data Preprocessing [20 pts]

We will use the **MSLR-WEB10K** dataset provided by Microsoft Research (Thanks to them) which can be downloaded from the following link:

https://www.microsoft.com/en-us/research/project/mslr/

MSLR-WEB10K is a machine learning dataset, in which queries and urls are represented by IDs. The dataset consists of feature vectors extracted from query-url pairs along with relevance judgment labels:

a. The relevance judgments are obtained from a retired labeling set of a commercial web search engine (Microsoft Bing), which take 5 values from 0 (irrelevant) to 4 (perfectly relevant).
b. The features are basically extracted by Microsoft Research and are those widely used in the research community.

In the data files, each row corresponds to a query-url pair. The first column is relevance label of the pair, the second column is query id, and the following columns are features. The larger value the relevance label has, the more relevant the query-url pair is. A query-url pair is represented by a 136-dimensional feature vector.

Below are two rows from MSLR-WEB10K dataset:

==========

0 qid:1 1:3 2:0 3:2 4:2 … 135:0 136:0

2 qid:1 1:3 2:3 3:0 4:0 … 135:0 136:0

==========

For a deeper understanding of the 136 features as well as their meaning, refer to
https://www.microsoft.com/en-us/research/project/mslr/

MSLR-WEB10K is a big dataset with 10000 queries. For computational feasibility, we will use only the first 1000 queries in this assignment. So, your first job is to download this dataset and write a code to create a smaller dataset which contains data related to only the first 1000 queries. Create 3 datasets from first 1000 queries as follows:

a. Training Data – consisting of queries 1-600
b. Validation Data– consisting of queries 701-800
c. Testing Data - consisting of queries 801-1000

2. Training [30 pts]

Now that you have created the dataset, your next job is to train the 4 learning to rank (LETOR) models based on your training dataset. The LETOR models are:

a. RankBoost
b. Coordinate Ascent
c. LambdaMART
d. Random Forests

You will use the RankLib library directly to conduct the training.

RankLib is a library of learning to rank algorithms. Currently, it has eight popular ranking algorithms implemented in the library. It also implements many retrieval metrics as well as provides many ways to carry out evaluation. More details about the installation and usage of Ranklib:

https://sourceforge.net/p/lemur/wiki/RankLib/

The following instructions must be followed while doing the training:

a. NDCG@10 should be used as the metric to optimize on the training data
b. You must use the validation set while doing the training
c. You can run each LETOR method with their default parameters
d. Once the training is completed, the model should be saved in a file through the RankLib tool.
e. In your report, include the following table and report numbers you saw during the training process. Explain your observations.

Method	Training NDCG@10 score	Training time in minutes
RankBoost		
Coordinate Ascent		
LambdaMART		
Random Forests		

3. Testing [20 pts]

Now it is time to evaluate the 4 learning to rank (LETOR) models you have trained on your testing dataset. Use Ranklib table to fill in the following table with numbers. Do you see a winner among the 4 LETOR methods? Explain your observations.

Method	Testing NDCG@10 score	Testing ERR@10 score
RankBoost		
Coordinate Ascent		
LambdaMART		
Random Forests		

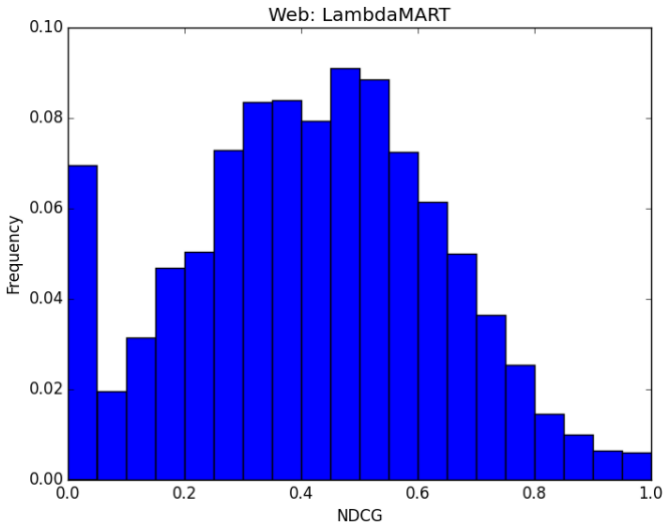
4. Query Level Analysis [30 pts]

Now it is time to do some deeper analysis on the testing performance of the 4 learning to rank (LETOR) methods.

a. **[10 pts]** Use Ranklib to find the nDCG@10 each method achieves for each of the test queries. It should be list like the following:

```
NDCG@10   801   0.0
NDCG@10   802   0.6722390270733757
NDCG@10   803   0.4772656487866462
NDCG@10   804   0.539003131276382
NDCG@10   805   0.6131471927654585
NDCG@10   806   1.0
NDCG@10   807   0.6309297535714574
NDCG@10   808   1.0
NDCG@10   809   0.2532778777010656
NDCG@10   810   1.0
NDCG@10   811   0.6131471927654585
NDCG@10   812   0.4772656487866462
NDCG@10   813   0.0
.....
```

b. **[10 pts]** Now create a normalized histogram of query-level-performances for each learning to rank (LETOR) methods. Below, a sample normalized histogram has been shown for LambdaMART. You will create 4 histograms in total from this step (one of each LETOR method). Do you see any patterns from the plot? Explain your observations.



c. **[10 pts]** Now, repeat 4(a) and 4(b) with testing metric ERR@10 instead of nDCG@10.