Search Engines:
11-442 / 11-642

Carnegie Mellon University

HW4: Learning to Rank
Due Apr 3, 11:59pm

The purpose of this assignment is to gain experience with using a machine learning algorithm to train a feature-based retrieval model.

This assignment consists of three major parts:

1. Add learning-to-rank capabilities to the program that you developed for homework assignments 1-4;
2. Conduct experiments;
3. Write a report describing your experiments and conclusions.

The report is an important part of your grade. Leave enough time to do a good job on it.

The Program

This homework adds four new capabilities to the search engine that you developed for previous homework assignments. Your program must:

1. Read training queries and relevance judgments from input files;
2. Calculate feature vectors for training documents;
3. Write the feature vectors to a file;
4. Call SVM^{rank} to train a retrieval model;
5. Read test queries from an input file;
6. Use BM25 to get inital rankings for test queries;
7. Calculate feature vectors for the top 100 ranked documents (for each query);
8. Write the feature vectors to a file;
9. Call SVM^{rank} to calculate scores for test documents;
10. Read the scores produced by SVM^{rank}; and
11. Write the final ranking in trec_eval input format.

Your program must implement the following features.

- **f_1:** Spam score for d (read from index).
  Hint: The spam score is stored in your index as the `spamScore` attribute. It is returned as a string.
  int spamScore = Integer.parseInt (Idx.getAttribute ("spamScore", docid));
- **f_2:** Url depth for d(number of '/' in the rawUrl field).
  Hint: The raw URL is stored in your index as the `rawUrl` attribute.
  String rawUrl = Idx.getAttribute ("rawUrl", docid);
- **f_3:** FromWikipedia score for d (1 if the rawUrl contains "wikipedia.org", otherwise 0).
- **f_4:** PageRank score for d (read from index).
  Hint: The PageRank is stored in your index as the `PageRank` attribute. It is returned as a string.

float prScore = Float.parseFloat (Idx.getAttribute ("PageRank", docid));

- **f_5:** BM25 score for <q, d_{body}>.
- **f_6:** Indri score for <q, d_{body}>.
- **f_7:** Term overlap score (also called `Coordination Match`) for <q, d_{body}>.
  Hint: Term overlap is defined as the percentage of query terms that match the document field.
- **f_8:** BM25 score for <q, d_{title}>.
- **f_9:** Indri score for <q, d_{title}>.
- **f_{10}:** Term overlap score (also called `Coordination Match`) for <q, d_{title}>.
- **f_{11}:** BM25 score for <q, d_{url}>.
- **f_{12}:** Indri score for <q, d_{url}>.
- **f_{13}:** Term overlap score (also called `Coordination Match`) for <q, d_{url}>.
- **f_{14}:** BM25 score for <q, d_{inlink}>.
- **f_{15}:** Indri score for <q, d_{inlink}>.
- **f_{16}:** Term overlap score (also called `Coordination Match`) for <q, d_{inlink}>.
- **f_{17}:** A custom feature - use your imagination.
- **f_{18}:** A custom feature - use your imagination.

Use what you have learned in the course so far to guide your development of custom features. You have considerable discretion about what features you develop. However, we will deduct points for features that we consider trivial or ridiculous. There needs to be a good reason why your feature might be expected to make a difference. Your features are hypotheses about what information improves search accuracy. It is not necessary that your features actually improve accuracy (although we hope that they will). We will evaluate your hypotheses, not the success of your hypotheses.

You may use whatever software design you wish, as long as your software is functionally correct and written entirely by you alone. The instructor and TAs have sketched a design approach that offers some guidance about how to implement your software, and issues that you may face. We strongly recommend that you read this document.

Your source code must run within our homework testing service.

Input

As in previous homeworks, your software must accept only one parameter, which is the name of a parameter file that is located in the current working directory. In addition to the parameters used in previous homeworks, your software must also support several new parameters for the query expansion algorithm, as described below.

- **letor:trainingQueryFile=** A file of training queries.
- **letor:trainingQrelsFile=** A file of relevance judgments. Column 1 is the query id. Column 2 is ignored. Column 3 is the document id. Column 4 indicates the degree of relevance (0-2).
- **letor:trainingFeatureVectorsFile=** The file of feature vectors that your software will write for the training queries.
- **letor:featureDisable=** A comma-separated list of features to disable for this experiment. For example, "letor:featureDisable=6,9,12,15" disables all Indri features. If this parameter is missing, use all features.
- **letor:svmRankLearnPath=** A path to the svm_rank_learn executable.
- **letor:svmRankClassifyPath=** A path to the svm_rank_classify executable.
- **letor:svmRankParamC=** The value of the c parameter for SVM^{rank}. 0.001 is a good default.
- **letor:svmRankModelFile=** The file where svm_rank_learn will write the learned model.
- **letor:testingFeatureVectorsFile=** The file of feature vectors that your software will write for the testing queries.
- **letor:testingDocumentScores=** The file of document scores that svm_rank_classify will write for the testing feature vectors.

Your program will also use parameters developed for previous homework assignments. The usage of most of the previous parameters is just what you would expect (e.g., the index parameter, the BM25 and Indri parameters). However, please note the usage of the following two parameters.

- **retrievalAlgorithm=**letor. This is a new value for an existing parameter. It indicates that your software should be used for a learning-to-rank experiment.
- **queryFilePath=** Your software already supports this parameter. In this assignment, it is the path to the testing queries (not the training queries).
- **trecEvalOutputPath=** Your software already supports this parameter. In this assignment, it is where your software writes the final, reranked output that will be used as input to trec_eval.

Output

Your software must write search results to a file in trec_eval input format, as it did for previous homework.

Data (Index)

You must download a new search engine index for this assignment. You will use this index for HW4 and HW5.

The corpus is 552,682 documents from the ClueWeb09 dataset. The corpus was indexed with Lucene. The Lucene index is available for downloading and running the experiments locally. The index is provided in two different file formats:

- A .zip file (2.2 GB or 2,324,846,184 bytes compressed, 2.7 GB uncompressed).
  md5 checksum: ef39fb6fdc393a59266bd49b847f41df
- A .tgz file (2.2 GB or 2,323,530,779 bytes compressed, 2.7 GB uncompressed).
  md5 checksum: f4283e722baf7562091e020b7a3aec5d

You should download the index as soon as possible in order to avoid longer download times as the homework deadline approaches.

Testing Your Software

Use the HW4 Testing Page to access the trec_eval and homework testing services. You used similar services for Homework 4. **Note:** The link for these services is **not the same** as it was for Homework 3. The requirements for this homework are a little different, so the testing apparatus must also be a little different.

Experiments and Analysis

After your system is developed, you must conduct experiments and an analysis that investigate the effectiveness of your custom features and the learning-to-rank approach in different situations.

Experiment: Baselines

Use your existing BM25 and Indri implementations to produce three baseline document rankings for the test queries.

- BM25 with unstructured, bag-of-words queries;
- Indri with unstructured, bag-of-words queries; and
- Indri with sequential dependency model queries.

Present the results for the three retrieval methods in a table that makes it easy to compare the results. These values serve as the baseline methods.

Experiment: Learning to Rank

Use your new learning-to-rank software to train four models that use different groups of features.

- **IR Fusion:** BM25 and Indri features (f_5, f_6, f_8, f_9, f_{11}, f_{12}, f_{14}, f_{15})
- **Content-based:** Features f_5-f_{16}
- **Base:** Features f_1-f_{16}
- **All:** Features f_1-f_{18}

Test your models on the test queries. Present the results for the four retrieval methods in a table that makes it easy to compare the results. These values serve as the baseline methods.

Experiment: Features

Experiment with four different combinations of features. Try to find the smallest set of features that delivers accurate results. We do not expect you to investigate all combinations of features. Your goal is to investigate the effectiveness of different features or groups of features, and to discard any that do not improve accuracy.

Present the results for the four retrieval methods in a table that makes it easy to compare the results. Discuss the trends that you observe; whether the learned retrieval models behaved as you expected; how the learned retrieval models compare to the baseline methods and the full feature set; and any other observations that you may have.

Analysis

Examine the models produced by SVM^{rank} to see whether you can identify the features that are more useful or less useful than other features. You may use svm2weight.pl, a perl script that computes the weight vector of linear SVM based on the model file; note that the file downloads as svm2weight.pl.txt, so you may want to rename it; usage is "perl svm2weight.pl MODEL_FILE". If you need more information, the SVM^{light} FAQ page provides information and tools for reading model files.

Discuss which features appear to be more useful and which features appear to be less useful. Support your observations with evidence from your experiments. Keep in mind that some of the features are highly correlated, which may affect the weights that were learned for those features.

Some of this discussion may overlap with your discussion of your experiments. However, in this section we are primarily interested in what information, if anything, you can get from the SVM^{rank} model files.

The Report and Source Code

You must submit one .zip, .gz, or .tar file to the software test web service that contains a report and your source code. Your latest submission before the homework deadline will be used for grading. When you submit your materials for grading, you do not need to select tests to run. We will run a complete set of tests on your software.

1. **Report:** You must describe your work and your analysis of the experimental results in a written report. A report template is provided in Microsoft Word and pdf formats. Please address all the questions in this template report within the specified section. Do not change the section headings.

   Submit your report in pdf format. Name your report *yourAndrewID*-HW4-Report.pdf. When your uploaded file is unzipped, the report should be in the same directory as your source code.

2. **Source code:** Your submission must include all of your source code, and any files necessary to make and run your source code. Your source code must run within our testing service, so please check that it does before you make your final submission. We will look at your source code, so be sure that it has reasonable documentation and can be understood by others.

FAQ

If you have questions not answered here, see the HW4 FAQ and the Homework Testing FAQ.

Updated on February 26, 2018
Jamie Callan