CS6200 Information Retrieval

Homework6: Machine Learning for IR

Objective

In this assignment, you will represent documents as numerical features, and apply machine learning to obtain retrieval ranked lists. The data is the AP89 collection we used for HW1.

Data

Restrict the data to documents present in the QREL. That is, for each of the 25 queries only consider documents that have a qrel assessment. You should have about 14193 documents; some of the docIDs will appear multiple times for multiple queries, while most documents will not appear at all.

Split the data randomly into 20 "training" quereis and 5 "testing" queries.

Part 1: Document-Query IR Features

The general plan is to build a query-doc static feature matrix.
qid-docid f1 f2 f3 … fd label
qid-docid f1 f2 f3 … fd label …
qid-docid f1 f2 f3 … fd label

You can rearrange this matrix in any format required by the training procedure of the learning algorithm.

Extract for each query-doc the IR features. These are your HW1 and HW2 models like BM25, Language Models, Cosine, Proximity, etc. The cells are feature values, or the scores given by the IR functions. The label is the qrel relevance value.

Note: your IR models from HW1 and HW2 might be truncated at 1000 docs, in which case the ranked lists wont contain features for every single qrel document. The better option is to go back to these IR models and compute the scores for all required qrel docs.
Alternative option: every document not in top 1000 for an IR function, gets a make-up score=feature value. **This make up score should not be 0 by default; instead it should be the lowest IR score form HW1 for that query, i.e. the score at rank 1000.** (A 0 score might be fine for some, but for certain IR functions like language models 0 is actually a big value.)

You can add other features (columns) that are doc-query-dependent (like number of query terms in document), or doc-dependent and query-independent (like pagerank).
You cannot add features that have, for a given document, different meanings on different queries, for example the unigram "China"; such a feature might be important for a query and irrelevant for a different query.

Train a learning algorithm

Using the "train" queries static matrix, train a learner to compute a model relating labels to the features. You can use a learning library like SciPy/NumPy (http://www.scipy.org), C4.5 (https://github.com/scottjulian/C4.5), Weka (http://www.cs.waikato.ac.nz/ml/weka/), LibLinear (http://www.csie.ntu.edu.tw/%7Ecjlin/liblinear/), SVM Light (http://svmlight.joachims.org), etc. The easiest models are linear regression and decision trees.

Test the model

For each of the 5 testing queries:

1. Run the model to obtain scores
2. Treat the scores as coming from an IR function, and rank the documents
3. Format the results as in HW1
4. Run trec*eval and report evaluation as "testing*performance".

Test the model on training data

Same as for testing, but on the 20 training queries. Run the learned model against the training matrix, compute prediction/scores, rank, and trec*eval.
Report as "training* performance".

Extra Credit

These extra problems are provided for students who wish to dig deeper into this project. Extra credit is meant to be significantly harder and more open-ended than the standard problems. We strongly recommend completing all of the above before attempting any of these problems.

Points will be awarded based on the difficulty of the solution you attempt and how far you get. You will receive no credit unless your solution is "at least half right," as determined by the graders.

EC1: Document static features

Extract for each document in the collection "static" features (query independent), such as document length, timestamp, pagerank, etc. Add these to the feature matrix, and rerun the learning algorithm(s).

EC2: Test on your crawled data

Extract the same features (as in the training matrix) for your evaluated documents (about 200/query). Run the learned model to obtain predictions, rank documents accordingly, and evaluate using the qrel produced in HW4 for your crawl.

EC3: Advance Learning Algorithms

Run fancy learning algorithms like SVM or Neural Networks

EC4: Ranking Algorithms

Run learning algorithms with a ranking objective, such as SVM-Rank, RankBoost, or LambdaMart.

Rubric

20 points
A proper static matrix setup for feature values and labels
15 points
feature values fro IR functions, at least 5 columns
15 points
Training successfully a learning algorithm
15 points
Evaluate on training queries
15 points
Evaluate on testing queries