# CS6200 Information Retrieval

## Homework6: Machine Learning for IR

# Objective

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

We have created a discussion thread in Piazza. It is meant to be a supportive space to help each other succeed in this assignment. Whether you're encountering hurdles, have discovered something interesting, or want to share your progress, this is the place

# Workflow

## Data

**Documents:** Limit the data to documents contained in the QREL file. This means that for each of the 25 queries, only documents with a qrel assessment should be considered. You should have about 14193 documents. Some of the docIDs will appear multiple times for different queries, while most documents won't appear at all. For each query, include additional non-relevant documents (not from qrel) to have roughly 1000 non-relevant docs per query. These can be obtained using IR runs from HW1.

**Queries:** Split the data randomly into 20 “training” queries and 5 “testing” queries.

## Document-Query IR Features

The general plan is to build a query-doc static feature matrix using the format below:
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 IR features for each query-doc pair. The feature include your HW1 and HW2 models such as BM25, Language Models, Cosine, and Proximity Search. The cells represent feature values, which are the scores given by the IR functions (retrieval models). The label correspond to the qrel relevance value.

You can include other features (columns) that are document-query dependent (such as the number of query terms in the document) or features that are document dependent and query independent (like pagerank).

However, refrain from adding features that have different meanings for a given document across different queries. For instance, the unigram "China" might be significant for one query but irrelevant for another. Using "China" as a feature in this case won't add any value.

**Notes:**

- As your IR models from HW1 and HW2 are truncated at 1000 docs, the ranked lists wont contain features for every qrel document. A preferable approach is to re-run these IR models and compute the scores for all required qrel docs.
- **Alternative option (NOT RECOMMENDED, AVOID):** Assign a made-up score (feature value) to every document not within the top 1000 for an IR function. *This made up score should not default to 0; instead, it should be the lowest IR score form HW1 for that query, specifically the score at rank 1000*. (While a 0 score might be fine for some IR functions, for others like language models, 0 actually represents a sibut for certain IR functions like language models 0 is actually a large value.)

## Train a learning algorithm

Using the “training” queries static matrix, train a learner to compute a model relating labels to the features. You can use a learning library like SciPy/NumPy, C4.5, Weka, LibLinear, SVM Light, 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 *treceval* 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 *treceval* . 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

For each document in the collection, extract "static" features that are query independent, such as document length, timestamp, pagerank, etc. Include these features to the feature matrix, and then 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 generate predictions, rank documents accordingly, and evaluate using the qrel produced in HW4 for your crawl.

### EC3: Advance Learning Algorithms

Run fancy learning algorithms such as SVM or Neural Networks

### EC4: Ranking Algorithms

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

### Rubric

Check Canvas for a detailed rubric