PA4 Learn2Rank Report

Dhruv Amin
James Webb

Design choices:

We built much of our system off of the provided skeleton code. Most notably we used the Learner.java abstraction class and generated 3 subclasses of it (PointwiseLearner.java, PairwiseLearner.java, and ComboLearner.java). PointwiseLearner uses standard Linear Regression trained on the tfidf scores for each field in a given document. PairwiseLearner uses an SVM to serve as a comparator of relevance between two documents. ComboLearner can run both Linear Regression and an SVM, as it is an experimental class for testing different combinations of systems. ComboLearner can also be modified to include (along with tfidfs) the BM25 score, the Smallest Window score, and the PageRank of each document. In practice, we achieved the best results when all three of these additional features were trained on.

Most of the functionality for this task we placed in Util.java, so that other learners would be able to easily calculate the tfidfs, BM25, Smallest Window, PageRank, and other features of a given document. In order to calculate BM25 and Smallest Window, we included the Scorer classes from PA3.

NDCG scores:

- Pointwise Logistic Regression Task 1:
  - Raw TF: 0.8360942810102602
  - Log TF (body): 0.829540800506027
  - Log TF (body, title): 0.8301445766527987
  - **Log TF (title): 0.8609851790570774**

  We experimented with different applications of log term frequencies as opposed to raw term frequencies. We figured the two most likely zones where raw term frequencies could be misleading were the title and the body zones. Empirically we determined log scaling when applied to only the title zone produces the best NDCG score. Our theory for this is that multiple appearances of a query term in the title of an document should not have a linearly scaling impact on that document's relevance.

- Pairwise SVM Task 2:
  - Linear SVM (standard C): 0.8369210630246372
  - Linear SVM (optimized C): 0.8381155280498082

    - RBF SVM (standard C, gamma): 0.8361273282499163
    - RBG SVM (optimized C, gamma): 0.8393345555759827

  We built the pairwise SVM system as described in the handout and while we came close to the baselines, we were ultimately unable to hit them. Our process was to generate the feature vectors in a similar fashion to how we did for Task 1, except we didn't use log scaling since it decreased our scores. We then generated an equal number of positive and negative difference vectors and trained the classifier to output a prediction. In our testing step, we made a feature vector for every document. Then for each queries' set of documents, we used a standard Java sorting algorithm but had the model influence the ordering the comparator choose by picking between two docs based on their difference vector and the output prediction. In order to optimize our NDCG scores, we did a grid search over all suggested C and gamma values to choose the ones that gave the best score. We think a minor bug is blocking us from reaching our optimal baseline score but were unable to track it down prior to the deadline.

- Combo Task 3:
    - BM25, Smallest Window, Pagerank: 0.8516883373822829
    - Smallest Window, Pagerank: 0.8498386689622913
    - BM25, Pagerank: 0.8367424756684713
    - BM25, Smallest Window: 0.8367424756684713
    - BM25: 0.8365990453196429
    - Smallest Window: 0.8365990453196429
    - Pagerank: 0.8365990453196429
    - **BM25, Smallest Window, Pagerank, Anchor Count, isSHTML: 0.8609851790570774**
    - BM25, Smallest Window, Pagerank, Anchor Count: 0.8607586948372336
    - BM25, Smallest Window, Pagerank, isSHTML: 0.8607586948372336

  Using the log scaling determined in Task 1, we incorporated more features into our Logistic Regression pointwise classifier. For the first part of this task, we did an ablation test of the three recommended features (BM25 score, Smallest Window, and PageRank). After empirically testing all combinations of these features, we were able to produce the best NDCG results by using all three.

  We then looked into the results we produced against the pa4.rel.dev gold standard. We noticed that many "important" pages (i.e. main pages of sites with very high anchor counts) were appearing lower than pages with almost no anchor counts. Our TF-IDF scorer only considered anchor counts that matched words in the query. We hypothesized that if two documents both had no anchor text in common with the query, the document with more anchor texts (counts) overall is probably more relevant anyway. We confirmed this by adding a feature that totals up all anchor counts for a

given document, regardless of what anchor text it is associated with. This feature feature led to an increased NDCG score.

After looking at our results further, we noticed that many queries (e.g. “is there free parking at stanford campus?”) had many highly relevant results of the type “.shtml.” We noticed our results often did not rank .shtml documents as high as it should have. Thus we added a feature that simply determines if the url of the document in question ends in “.shtml”. This feature also led to an increase in NDCG score.