Learning to Rank

February 2, 2017

Autors:

- Dana Kianfar - 11391014
- Jose Gallego - 11390689

1 Introduction

The main goal of this assignment was to implement and compare the performance of several approaches to the Learning to Rank (LTR) problem. We received an implementation of a point-wise method base on squared-loss minimization and implemented RankNet (pairwise) and LambdaNet (listwise) algorithms.

2 Methodology

The training of the methods was executed on a database with 5 cross-validation folds, each containing the 90 training, 30 validation and 30 test examples with no overlap. The dataset is constructed for a homepage finding task, therefore each query should have exactly one relevant document label. Initial data exploration indicated that some queries were noisy as they had zero or more than one relevant document labels. We took these cases into consideration in our implementation.

For each algorithm, we obtained results on all 5 folds of the data where we trained a model on the training data for 200 epochs and reported the objective function evaluation, training mNDCG, and validation mNDCG for each epoch. After training on all folds of the dataset, we obtained five candidate models for each algorithm. For each algorithm we then chose a top candidate according to its validation mNDCG at the end of 200 epochs. For all three algorithms, fold 3 provided the best validation mNDCG. We used the top candidate to report the test mNDCG on the fold it was obtained from, which is fold 3.

We used L1 and L2 regularization using the default assignment values.

3 Pointwise MSE

As our dataset is heavily unbalanced, with approximately one positive label per query, we did not expect to achieve reasonable mNDCG results from this algorithm. This is due to the fact that a point-wise approach does not take the ranking structure into account, and the training involves a squared-loss function instead of an IR metric directly.

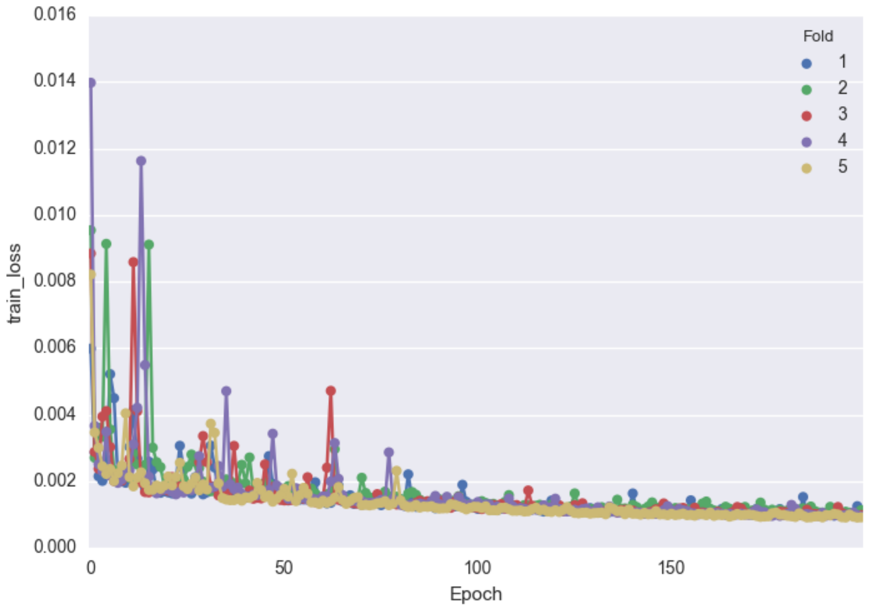

Figure 1. Pointwise method training loss for each fold

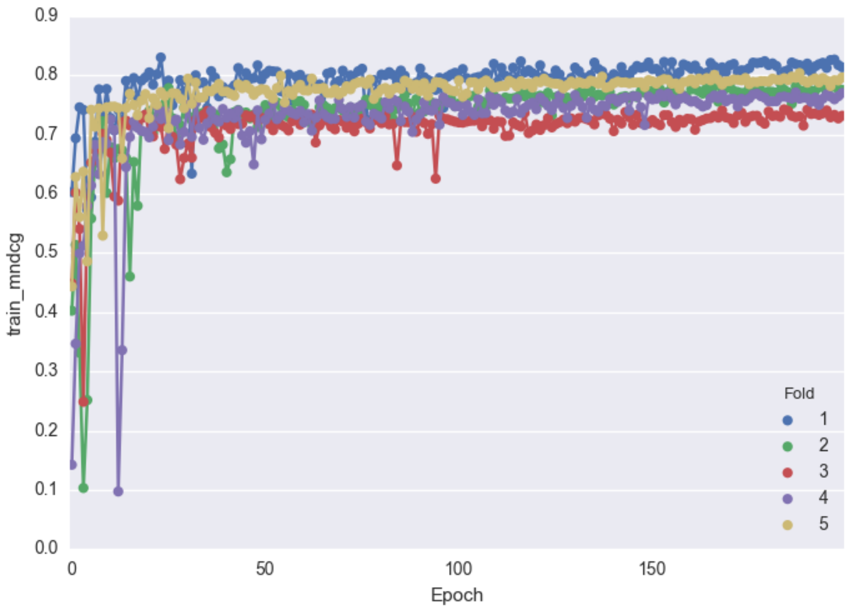

Figure 2. Pointwise method training mNDCG for each fold

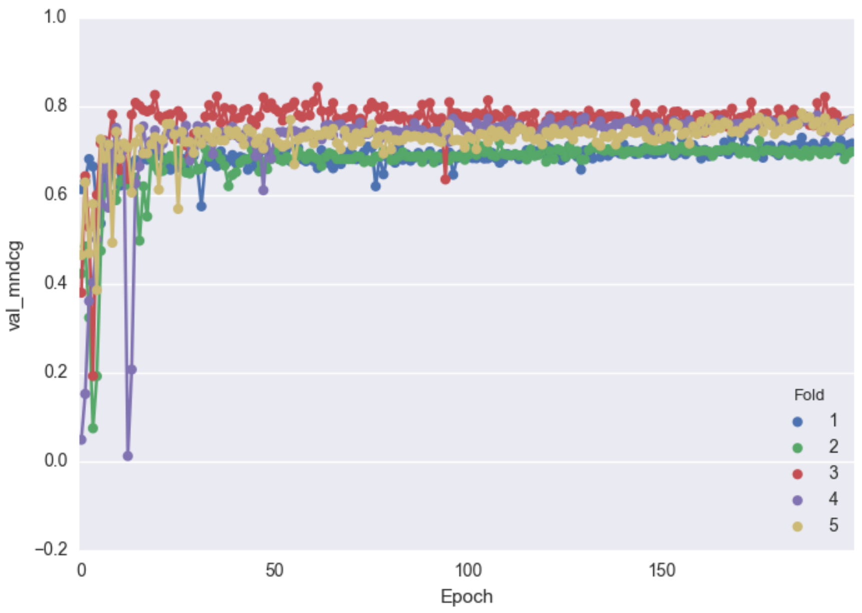

Figure 3. Pointwise method validation mNDCG for each fold

The training algorithm displays good behavior, with a high error function drop in the initial epochs and a relatively low convergence after iteration 50. There is also a decrease in the variance of the errors across folds over time. This might configure a overfitting scenario in terms of the training error. However, the fact that the objective function and the mNDCG are not directly coupled by the training scheme does not induce *perfect* mNDCG values on the training data, as can be seen in Figure 2. Besides, there variance on the mNDCG obtained on the training and validation sets is low across folds.

4 Pairwise RankNet

We took advantage of the sparsity of the S matrix used in RankNet and LambdaNet to improve the runtimes of our algorithms. We computed the S matrix for each query in the training and validation set for each fold in advance, to reduce overhead. Given the extremely low number of relevant labels per query, each S matrix is very sparse and can be computed and stored efficiently. Besides, we decided to use $\sigma = 1$ for mathematical simplicity on RankNet and LambdaRank.

For this model, we see a clear overfitting trend. The training utility flattens around 50 iterations for all folds, however the corresponding mNDCG values increase until around iteration 200 but the validation mNDCG values have an almost monotonously decreasing behaviour. The results confirm the effectiveness of the introduction of the λs as an strategy to learn ranking structures (reflected on a much better training and validation mNDCG) and to alleviate the imbalanced-data problem. In this case, the model with the best mNDCG was found at epoch 30 on fold 3.

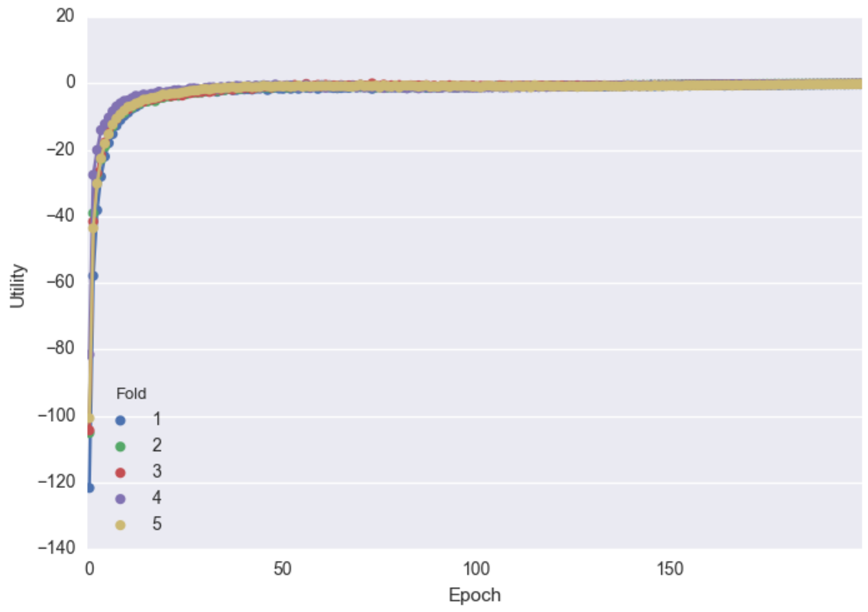

Figure 4. Pairwise method utility for each fold

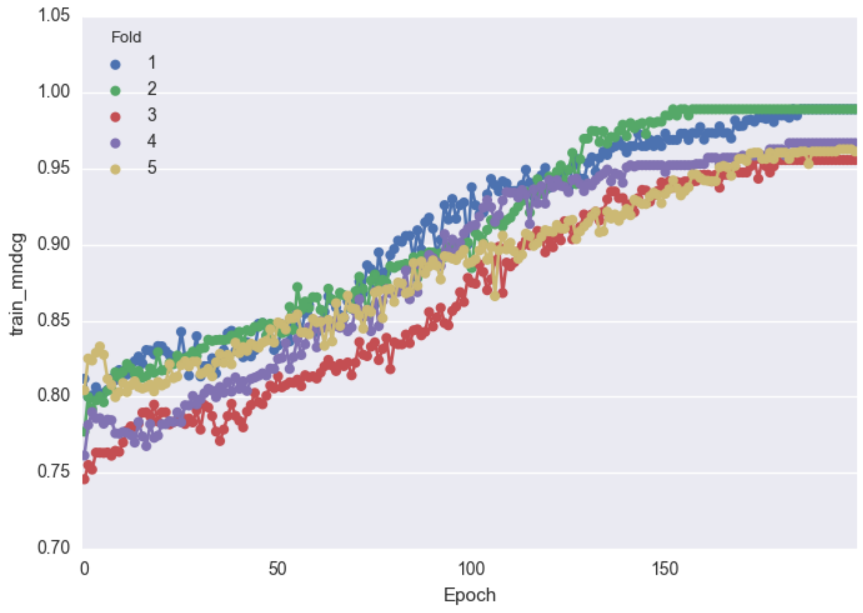

Figure 5. Pairwise method training mNDCG for each fold

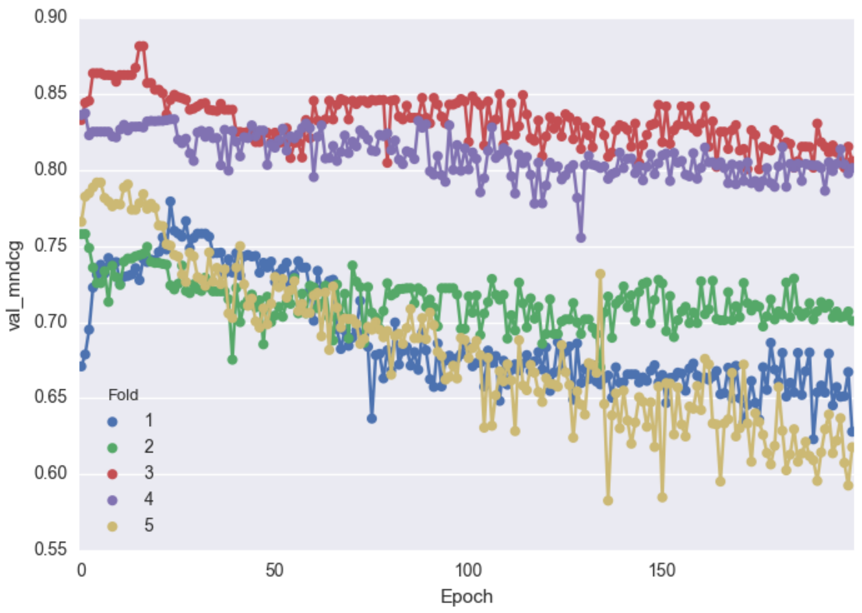

Figure 6. Pairwise method validation mNDCG for each fold

5 Listwise LambdaNet

Note that the calculation of $|\Delta NDCG|$ can be simplified from calculating NDCG twice to only considering the discounted gain that each of the swapped documents generates. Let's assume that we have a ranking in which the documents at position i and j are to be interchanged:

$$R1 = [1, 2, 3, \ldots, i, \ldots, j, \ldots, 1000]$$
$$\uparrow \_\_ \uparrow$$

This generates the ranking:

$$R2 = [1, 2, 3, \ldots, j, \ldots, i, \ldots, 1000]$$

Thus, calculating the $|\Delta NDCG|$ is equivalent to:

$$\begin{aligned}
\Delta NDCG &= |NDCG(R2) - NDCG(R1)| \\
&= \left| \left[\sum_{k \neq i,j} \frac{2^{rel_k} - 1}{\log(1+k)} + \frac{2^{rel_i} - 1}{\log(1+j)} + \frac{2^{rel_j} - 1}{\log(1+i)} \right] - \left[\sum_{k \neq i,j} \frac{2^{rel_k} - 1}{\log(1+k)} + \frac{2^{rel_i} - 1}{\log(1+i)} + \frac{2^{rel_j} - 1}{\log(1+j)} \right] \right| \\
&= \left| \left(2^{rel_i} - 2^{rel_j} \right) \left(\frac{1}{\log(1+j)} - \frac{1}{\log(1+i)} \right) \right|
\end{aligned}$$

The behavior of this model is very similar to the one observed in RankNet. There is a good performance on the training mNDCG, however the advantage of introducing the $|\Delta NDCG|$ factor in the computation on λ is not very noticeable. This might be due to the presence of only one relevant document per query. As well as the pairwise method, the best validation mNDCG was found on fold 3, around 30 iterations.

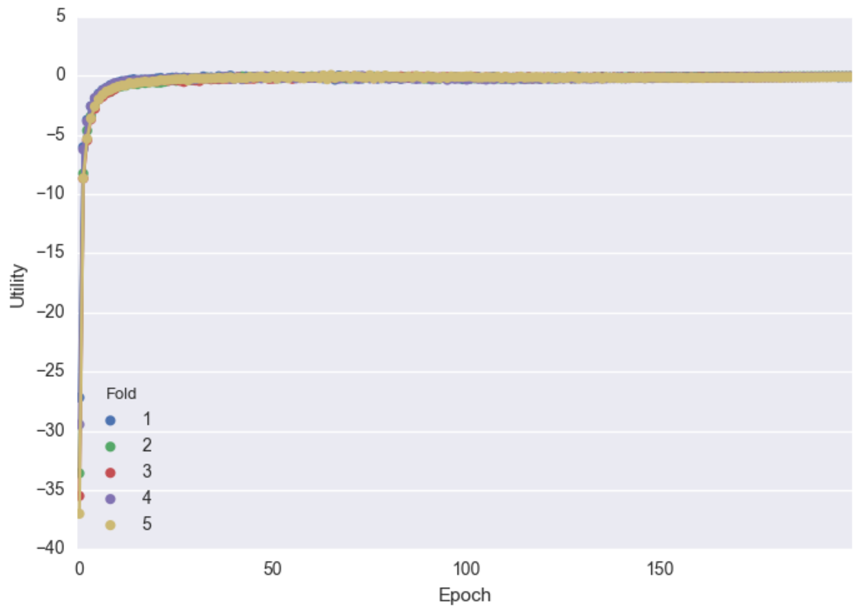

Figure 7. Listwise method utility for each fold

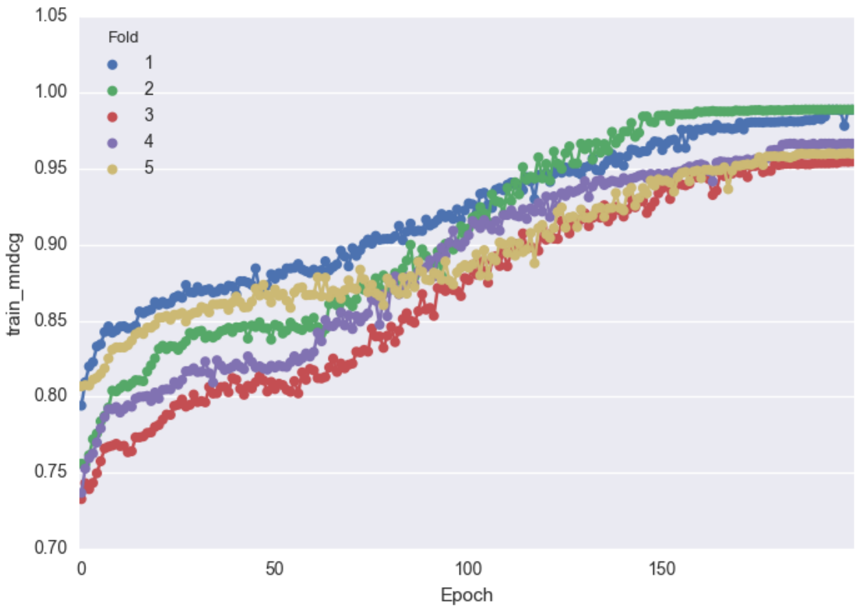

Figure 8. Listwise method training mNDCG for each fold

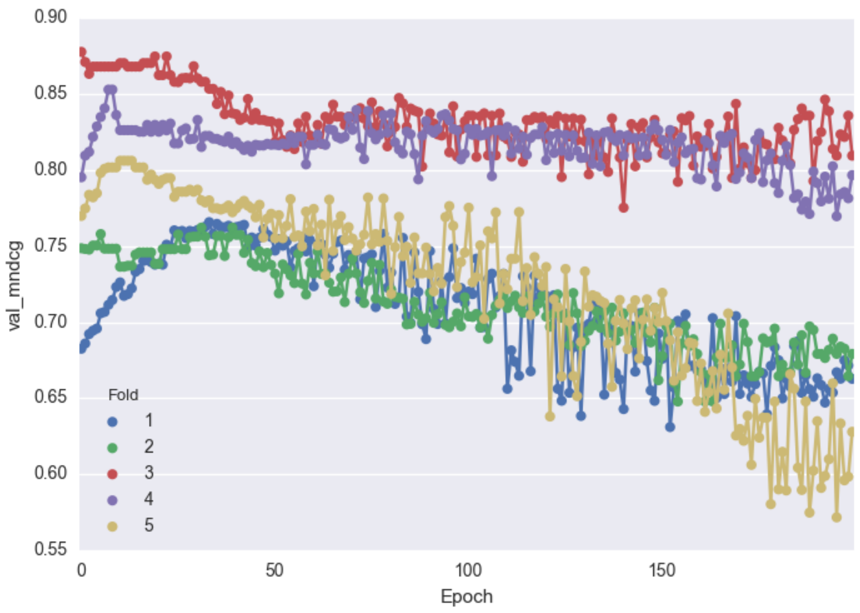

Figure 9. Listwise method validation mNDCG for each fold

6 Model Comparison

We compare the three algorithms by their training and validation mNDCG across all 5 folds on 200 epochs. We display their learning curves below, and for each algorithm summarize the training and validation mNDCG to provide a concise plot. The variance across the folds is displayed by the vertical bars which represent their 95% confidence intervals.

We observe in Figure 10 that both training and validation mNDCG scores for the pointwise algorithm converge around epoch 50. The best ranker for this algorithm was trained after 50 epochs on fold 3 with a validation mNDCG of 0.72. We selected this ranker to report the test score on the same fold. It achieved a test mNDCG of 0.77.

In Figure 11, we see that both training and validation mNDCG scores for the pairwise algorithm converge around epoch 30. The best ranker for this algorithm was trained after 30 epochs on fold 3 with a validation mNDCG of 0.84. This ranker obtained a test mNDCG of 0.89 on the same fold.

We observe in Figure 12 that both training and validation mNDCG scores for the listwise algorithm converge around epoch 30. The best ranker for this algorithm was trained after 30 epochs on fold 3 with a validation mNDCG of 0.87. This ranker obtained a test mNDCG of 0.91 on the same fold.

Overall, the pairwise and listwise methods achieve a better validation mNDCG than the pointwise method. The listwise method achieves a slightly higher validation mNDCG than the pairwise.

6.1 Statistical tests

Before conducting our experiments we expected both pairwise and listwise methods to outperform the pointwise method due to reasons discussed previously; namely that a pointwise ap-

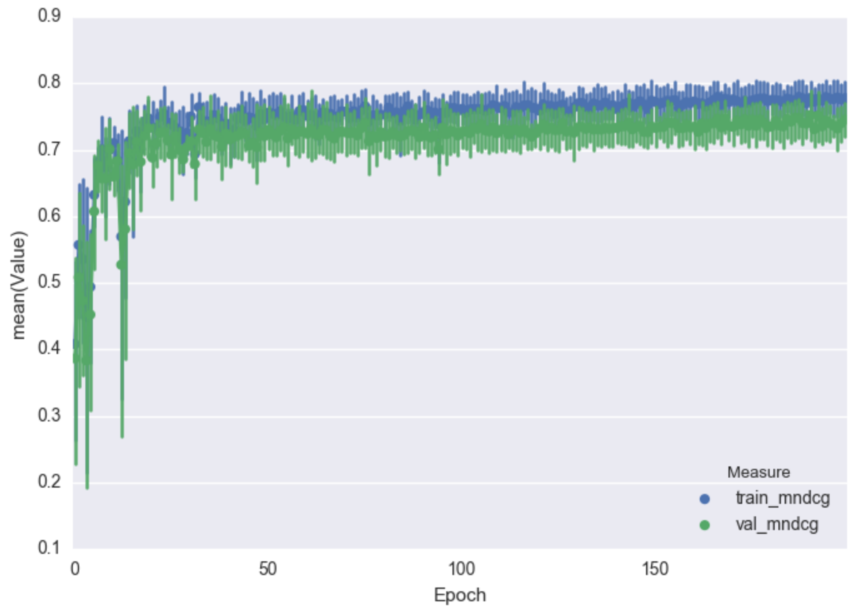

Figure 10. Pointwise method training average mNDCG across folds

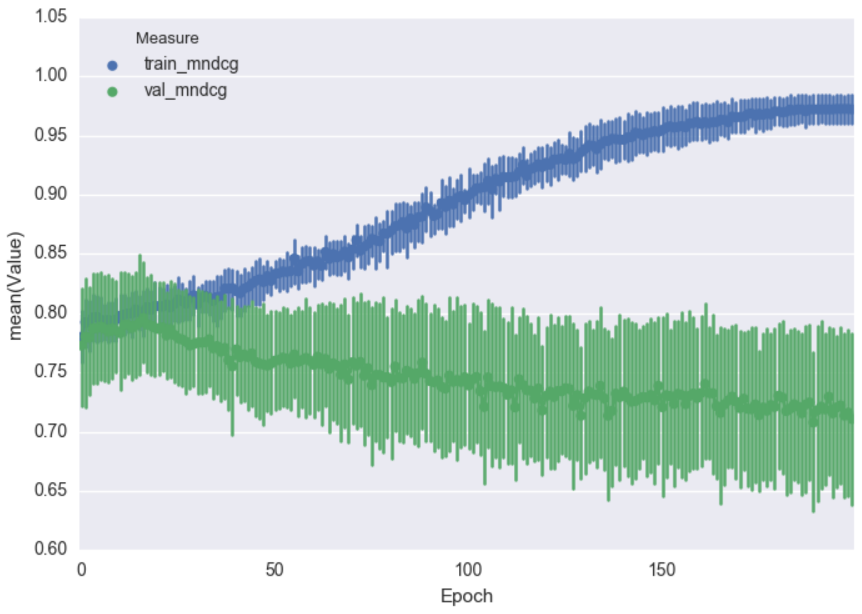

Figure 11. Pairwise method training average mNDCG across folds

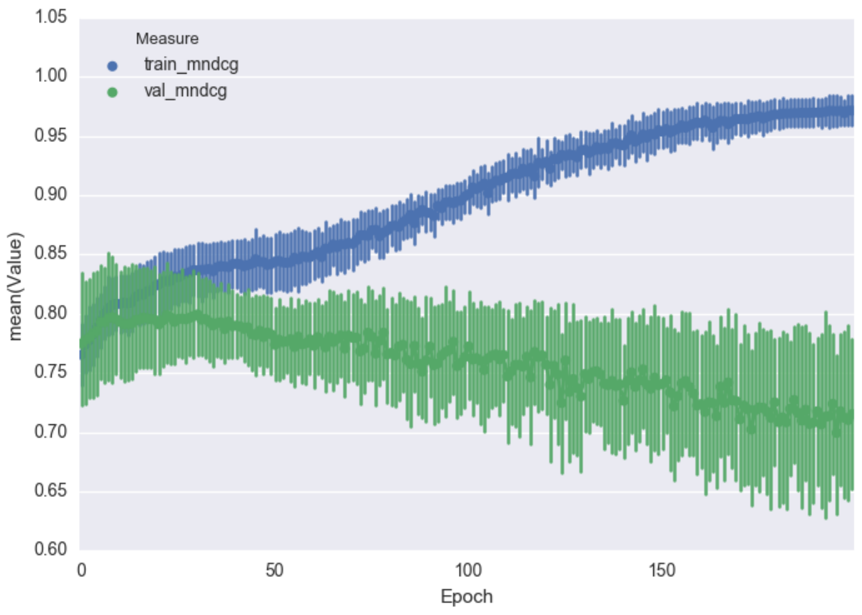

Figure 12. Listwise method training average mNDCG across folds

proach is ill-suited for the heavily skewed distribution of labels, and moreover the ranking structures are invisible to the squared-loss gradient descent optimizer. As all three methods obtained their best rankers on fold 3, we were able to perform one-sided sign-binomial 30-sample tests on the NDCG of each test query. We conducted a total of three tests, where we compared both pairwise and listwise algorithms against pointwise, and listwise against pairwise. We excluded queries where any two paired algorithm's NDCG values tied.

Algorithm	Best fold	Validation mNDCG	Test mNDCG
Pointwise	3	0.72	0.77
Pairwise	3	0.84	0.89
Listwise	3	0.87	0.91

Pairwise vs Pointwise As expected, the null hypothesis was rejected in favor of the pairwise algorithm with a p-value of 0.01.

Listwise vs Pointwise The null hypothesis was rejected in favor of the listwise algorithm with a p-value of 0.0002.

Listwise vs Pairwise There was not sufficient evidence to reject the null hypothesis in this test, where we obtained a p-value of 0.63.