Assignment 3 - Information Retrieval

Fije van Overeem 10373535 & Diede Rusticus 10909486

January 2017

Abstract

This is an assignment report covering Homework 3, for the course Information Retrieval on the University of Amsterdam. The task was to transform a (provided) neural net pointwise algorithm to a pairwise (Ranknet) and a listwise algorithm (LambdaRank) and evaluate them by their average NDCG@10 measure through 5 fold cross validation. It showed that there were no significant differences between the three algorithms after 5 epochs. However, the pointwise algorithm converges much slower than pair and listwise. Their mean NDCG scores lie around 0.7.

1 Experiment

This section describes the experiment by the used data, algorithm comparison and setup.

1.1 Data

The training data consists of queries based on a search for a homepage. This means there is only 1 relevant document per query. Each query comes with 1000 ranked documents of which most cases only one is found relevant. However, there is some noise in the data: some queries come with multiple relevant documents and some of them with none. As it should not affect training results significantly, this fact is ignored.

1.2 Algorithm Comparison

In this task we compare different kinds of ranking algorithms: pointwise (provided), pairwise and listwise ranking. It is interesting to investigate beforehand which one of the algorithms should be appropriate for a task with the particular data provided, i.e. where the relevance is binary (0: irrelevant, 1: relevant) and solely one relevant document per query. An answer is already provided in the assignment: pairwise ranking is expected to perform better than pointwise ranking because pointwise will try to predict for all zeroes. Because there's only one relevant document, it will more likely predict this document as zero as well. This in contrast to pairwise ranking, which takes into account whether the labels of two documents are a tie or not. And when there's a non-tie, it automatically means there's a relevant document in the pair. A listwise approach might be faster to train, but doesn't add much when there's only 1 relevant document.

1.3 Setup

The experimental design needed to be implemented for a 5 fold validation for NDCG@10. Per fold, all algorithms are trained for 5 epochs. After every epoch the model is evaluated on the validation set. The best model is saved and after the 5 epochs evaluated on the test set. The mean test NDCGs are finally compared of the three algorithms.

Regularisation of the loss is used for all three algorithms. The weights in a network are then regularised.

2 Results

The results of pointwise, pairwise and listwise are here denoted through graphs showing the NDCG scores on the validation set and some experiment conclusions are depicted.

2.1 Pointwise algorithm

Figure 1 shows the NDCGs@10 of the pointwise algorithm over time on the validation set. Pointwise has a fast runtime because of the simple squared error loss function during training. A pointwise algorithm is characterised by trying to predict all the zeroes. Moreover, it will simply predict everything as a zero and ignore the single relevant document.

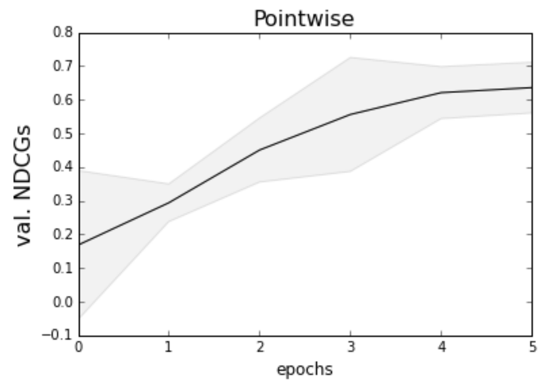

Figure 1: The mean NDCGs and their standard deviations on the validation set of the pointwise algorithm over time.

2.2 Pairwise algorithm

A pairwise algorithm would not try to predict all the zeroes. Instead, it would only consider the non-ties as those are the ones involving the single relevant document. For each document a force is calculated, either positive or negative. This force expresses the 'push up' or 'push down' in the ranking for this document. In order to transform the pointwise algorithm to the pairwise algorithm, we had to change the loss function by implementing the lambda loss. Figure 2 shows the NDCGs@10 of the pairwise algorithm using RankNet. The performance already converges after one epoch at an NDCG of around 0.7. With a very small standard deviation overall, the uncertainty is low. The implementation lacked some efficiency as it calculates the lambdas by evaluating all documents compared to all documents, resulting in 1000 x 1000 comparisons. This would not have been necessary, as the 1000x1000 matrix is antisymmetric. Therefore, exploring only the document pairs where u > v would have sufficed.

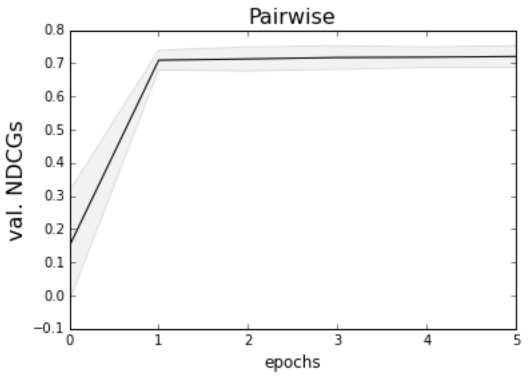

Figure 2: The mean NDCGs and their standard deviations on the validation set of the pairwise (RankNet) algorithm over time.

2.3 Listwise algorithm

LambdaRank is used for the listwise algorithm. It differs from RankNet by calculating the influence on the NDCG if document u and v are flipped in the ranking. Implementing LambdaRank was just an extension of RankNet, by multiplying the lambdas with the absolute difference of NDCG caused by the flip. Figure 3 shows the NDCGs@10 of the listwise algorithm using LambdaRank. It converges a little slower than RankNet, that is after 4 epochs to an NDCG of 0.7. The standard deviations are higher so the uncertainty is higher. It has the same runtime as the pairwise algorithm so there was also still room for improvements in the efficiency of the algorithm.

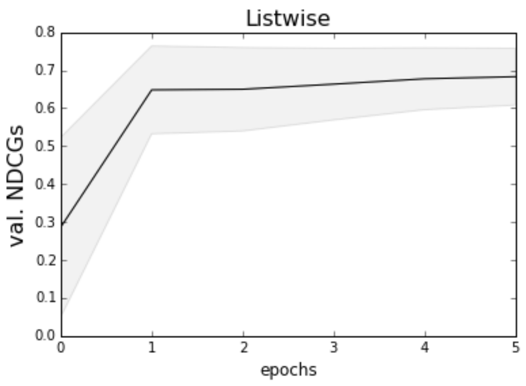

Figure 3: The mean NDCGs and their standard deviations on the validation set of the listwise (LambdaRank) algorithm over time.

3 Analysis

Figure 4 shows the differences of the mean NDCGs of the five folds of the three algorithms on the test sets, after 5 epochs. This gave the following results: Pointwise: mean = 0.61, stddev = 0,12 Pairwise: mean = 0.71, stddev = 0,08 Listwise: mean = 0.73, stddev = 0,06. The results show no significant differences in their means. For our experiment however, pointwise performed worse compared to pairwise and listwise. In Figure 1,2, and 3 it showed that all three algorithms have a positive influence on the ranking because of the low NCDG before training.

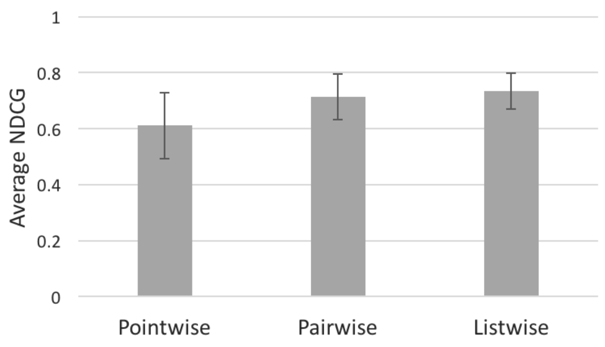

Figure 4: Testset average NDCGS of the three algorithms. The errorbars represent the average +- the standard deviation