High Accuracy Retrieval with Multiple Nested Ranker

Irina Matveeva*
University of Chicago
5801 S. Ellis Ave
Chicago, IL 60637
matveeva@uchicago.edu

Chris Burges
Microsoft Research
One Microsoft Way
Redmond, WA 98052
cburges@microsoft.com

Timo Burkard
MSN Search
One Microsoft Way
Redmond, WA 98052
tburkard@microsoft.com

Andy Laucius
MSN Search
One Microsoft Way
Redmond, WA 98052
andrewla@microsoft.com

Leon Wong
MSN Search
One Microsoft Way
Redmond, WA 98052
leonw@microsoft.com

ABSTRACT

High precision at the top ranks has become a new focus of research in information retrieval. This paper presents the multiple nested ranker approach that improves the accuracy at the top ranks by iteratively re-ranking the top scoring documents. At each iteration, this approach uses the RankNet learning algorithm to re-rank a subset of the results. This splits the problem into smaller and easier tasks and generates a new distribution of the results to be learned by the algorithm. We evaluate this approach using different settings on a data set labeled with several degrees of relevance. We use the normalized discounted cumulative gain (NDCG) to measure the performance because it depends not only on the position but also on the relevance score of the document in the ranked list. Our experiments show that making the learning algorithm concentrate on the top scoring results improves precision at the top ten documents in terms of the NDCG score.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation, Performance

Keywords

Ad-hoc retrieval, high accuracy retrieval, re-ranking

*Work performed while visiting Microsoft Research

SIGIR'06, August 6–11, 2006, Seattle, Washington, USA.

1. INTRODUCTION

Traditionally, the goal of ad-hoc information retrieval was to achieve good performance in terms of both precision and recall. Recently, the focus has shifted to high precision at the top of the results list. With the growing size of the Web collections, users are now primarily interested in high accuracy defined as high precision at the top ranks [16, 13, 10, 18]. Users' studies showed that users typically look at very few results and mostly look at the results at the very top of the list returned by the search engine, see [9, 11, 8]. Recognizing this trend, TREC introduced the High Accuracy Retrieval from Documents (HARD) track that includes user specific information to improve retrieval accuracy [6, 7]. Jarvelin et al. proposed to base the evaluation of the IR methods on the retrieval of highly relevant documents [10] and presented normalized discounted cumulative gain (NDCG) as a new measure. NDCG was then applied to analyse the TREC's web track results [18]. More recently, Shah et. al [16] integrated some question answering techniques into the ad-hoc retrieval to improve precision at the top of the results list. They also addressed the issue of performance evaluation in terms of precision only and used the mean reciprocal rank (MRR) as performance measure.

We pose the problem of achieving high accuracy as learning the re-ranking of the results at the top of the results list. We propose the multiple nested ranker approach which is applied to the list of results returned by the search engine. This approach re-ranks the documents on the results list in stages, at each stage applying the RankNet [2] algorithm to learn a new ranking.

Typically, ranking methods are applied to the full set of the per query results. Even when the ranked list is generated iteratively, for example when using relevance feedback or meta-data, at each iteration the retrieval algorithm is applied to the full set of the documents. Boosting [3], and in particular RankBoost [4], also performs learning in stages. But it uses the whole training set as input at each stage, with more weight put on difficult examples. It is a very difficult task, however, to learn how to rank a very large number of documents for any possible query. We adapt the

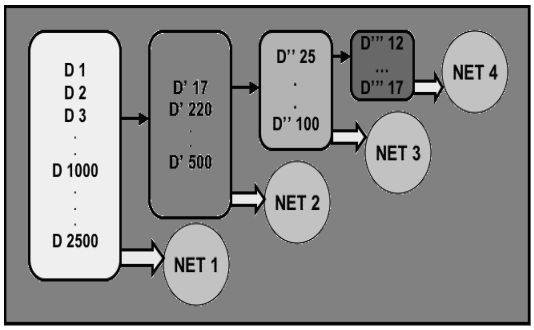

Figure 1: Training procedure for the multiple nested ranker. NET 1 is trained on the sets of 2500 documents D per query, NET 2 is trained on the sets of the top 1000 documents D′ per query, NET 3 is trained on the top 100 documents D″, NET 4 is trained on the top 10 documents D‴.

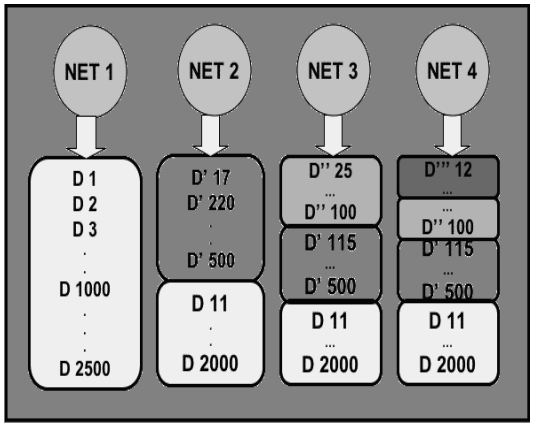

Figure 2: Re-ranking procedure for the multiple nested ranker. NET 1 is applied to the sets of 2500 documents D per query, NET 2 is applied to the sets of the top 1000 documents D′ per query, NET 3 is applied to the top 100 documents D″, NET 4 is applied to the top 10 documents D‴.

problem of learning the ranking of the retrieved results to the high accuracy task in the following way. We start with the assumption that the results list returned by the search engine already produces a sufficiently good ranking so that some relevant documents are placed somewhere near the top of the ranked list. Given the success of many retrieval systems and commercial search engines, this assumptions seems very reasonable, see for example [8]. Since we are interested in high accuracy as opposed to recall, we concentrate on improving the ranks of relevant documents at top ranks. The multiple nested ranker performs re-ranking of the results in stages, at each stage generating a new distribution of the results. The training set for each subsequent stage is pruned to include only the results that are ranked high by the previous ranker. We will refer to this pruning procedure as telescoping. Telescoping splits the problem into smaller and, hopefully, easier sub-tasks to learn the ranking for each of the stages separately. At the last stage, only a few (e.g. 10) documents are re-ranked to make sure that the most relevant among them will be placed on the top of the list.

Since in real life the relevance assignment is often not binary, but reflects the degree of relevance of each of the results [10, 18], we evaluate the performance of our approach using normalized discounted cumulative gain (NDCG) [10]. The NDCG score depends not only on the position but also on the relevance score of the document in the ranked list.

The rest of the paper is organized as follows. Section 2 describes the multiple nested ranker algorithm and outlines the RankNet algorithm. Section 3 describes the NDCG measure, section 4 contains the details about the data set. Sections 5 and 6 describe our experiments, section 7 contains the analysis of the experimental results. We conclude with section 9.

2. MULTIPLE NESTED RANKER

We propose to use the multiple nested ranker as the second part of the retrieval process. A search engine retrieves documents in respond to the query and ranks them using some ranking algorithm which we refer to as "basic ranker". We make the assumption that the basic ranker already produces a good ranking and that a number of relevant documents are placed somewhere near the top of the ranked list. Telescoping is applied to the first few thousands results returned by the search engine to learn a better ranking for the relevant results. The multiple nested ranker algorithm has two components: the telescoping procedure and a re-ranking algorithm. The re-ranking algorithm learns how to score documents so that the relevant documents receive higher scores. It uses the set of training queries $Q = (q_1, ..., q_{|Q|})$ to learn the scoring function. For each query q_i we have a set of documents that were ranked among the top $N1$ results by the basic ranker used in the search engine, $D_i = (d_{i1}, ..., d_{iN1})$. Some of these documents have manually assigned relevance labels, the rest is unlabeled. The training set for the re-ranking algorithm contains all documents returned for the training queries, $D = (D_1, ..., D_{|Q|})$. The multiple nested ranker approach uses the RankNet algorithm [2], discussed below. The RankNet learns a neural net to assign scores to documents. One net is learned for all training documents. The sets of documents corresponding to individual queries are sorted by the net output to produce their ranking.

In the training phase, telescoping determines the subset of the data used to train the RankNet algorithm. Figure 1 illustrates how telescoping is applied to the results set for each query. At each stage the RankNet is presented with a new distribution of the per query results containing subsets of the high ranked documents. At the first stage the RankNet is trained on the whole set of the top $N1$ per query results. In our experiments we used N_1=2500 documents per query. The training procedure computes the first net, Net_1. We sort each set of documents D_i by decreasing score according to Net_1. After that, the training set is modified so that only the top N_2 documents that receive the highest scores according to Net_1 remain for each query, i.e. $D'_i = (d'_{i1}, ..., d'_{iN2})$ and the next training set is $D' = (D'_1, ..., D'_{|Q|})$. At the second stage the RankNet is trained on these sets of top N_2 documents. The second stage produces Net_2 and only the N_3 top scoring documents per query are kept for the next training set.

Telescoping is also applied in the test phase. The re-ranking is done using the same number of stages as during training. At the first stage Net_1 is applied to all N_1=2500 documents per test query. Then Net_2 is applied to the top N_2 documents that receive the highest scores according to Net_1 and so on. This amounts to fixing the Net_1 ranks of the documents at ranks from $N1$ to ($N2$-1) after the first stage, re-ranking the top $N2$ documents with Net_2, again fixing the ranks of the documents placed from the rank $N2$ to ($N3$-1) after the second stage, re-ranking the top $N3$ results with Net_3 and so on. Thus, after each telescoping stage we have a ranked list for all $N1$ results per query which we use for the evaluation, as can be seen in Figure 2.

We used four stages with N_1=2500, N_2=1000, N_3=100, N_4=10 and also three stages with N_1=2500, N_2=100, N_3=10. The same telescoping procedure was applied to the validation set.

As opposed to boosting, this approach splits the problem into smaller pieces so that each net has a smaller and simpler task. Telescoping removes presumably difficult relevant documents at the bottom of the ranked list from the training set and forces the algorithm to concentrate on the ranking of the high scoring relevant documents. In addition, as we decrease the size of the training set roughly exponentially, more sophisticated algorithms can be used to learn the ranking at later stages.

2.1 RankNet

For completeness, we provide a brief overview of the RankNet algorithm [2] to give the reader some intuition about the learning process. We omit the details because this algorithm is used as a black box within the multiple nested ranker.

At each stage of the multiple nested ranker the RankNet algorithm learns how to rank the results so that the relevant documents appear at the top of the list. To achieve this, RankNet tries to learn the correct ordering of pairs of documents in the ranked lists of individual queries. The cost function of the RankNet algorithm depends on the difference of the outputs of pairs of consecutive training samples (x_1, x_2). The cost is minimized when the document x_1 with a higher relevance label receives a higher score, i.e. when $f(x_1) > f(x_2)$.

Burges et al. [2] propose to learn ranking using a probabilistic cost function based on pairs of examples. They consider models where the learning algorithm is given a set of pairs of samples [A,B] in R^d together with the target probabilities $\bar{P}_{AB}$ that sample A is to be ranked higher than sample B. With models of the form $f : R^d \longmapsto R$, the rank order of a set of examples is specified by the real values taken by f. More specifically, it is assumed that $f(x_i) > f(x_j)$ means that the model ranks x_i higher than x_j. With the modeled posterior probabilities $P_{ij} \equiv$ Prob("x_i is ranked higher than x_j") and their target probabilities $\bar{P}_{ij}$, Burges et al. [2] develop their framework using the cross entropy cost function

$$c(o_{ij}) = -\bar{P}_{ij} \log P_{ij} - (1 - \bar{P}_{ij}) \log(1 - P_{ij})$$

where $o_{ij} \equiv f(x_i) - f(x_j)$. The map from the outputs to probabilities is modeled using a logistic function [1]

$$P_{ij} \equiv \frac{e^{o_{ij}}}{1 + e^{o_{ij}}}$$

So that final cost becomes

$$c(o_{ij}) = -\bar{P}_{ij} o_{ij} + \log(1 + e^{o_{ij}})$$

The above cost function is very general. The RankNet algorithm uses it with the neural network models [12] to learn the ranking. Burges et al. [2] use a two-layer net with a single output node.

As a reminder, the neural net output function for the ith sample is described using the transfer function of each node in the jth layer of the nodes, g^j, and the weights w^{ji}_{kn} on the connections between the nodes in different layers with the corresponding offsets b^{ji}_{kn}. Here the upper indices index the node layer, and the lower indices index the nodes within each corresponding layer. The net output function of a two-layer net with one output node for the ith sample, f_i is

$$f_i \equiv f(x_i) = g^3(\sum_j w^{32}_j g^2(\sum_k w^{21}_{jk} x_k + b^2_j) + b^3)$$

The parameters α_k of the neural net model are updated depending on their contribution to the cost function measured as the derivative $\frac{\partial c}{\partial \alpha_k}$. The parameter value is updated using a positive learning rate η_k as

$$\alpha_{k+1} = \alpha_k + \delta\alpha_k = \alpha_k - \eta_k \frac{\partial c}{\partial \alpha_k}$$

Burges et al. [2] generalize the above derivations to the ranking problem in the following way. The cost function becomes a function of the difference of the outputs of two consecutive training samples: c(f_1-f_2), assuming that the first sample has a higher or the same rank as the second sample. The gradient of the cost becomes

$$\frac{\partial c}{\partial \alpha_k} = (\frac{\partial f_1}{\partial \alpha_k} - \frac{\partial f_2}{\partial \alpha_k}) c',$$

where $c' \equiv c'(f_1 - f_2)$. The subscripts denote the index of the training sample.

All other derivatives also take the form of the difference of a term depending on x_1 and a term depending on x_2, which are coupled by an overall multiplicative factor of c' which depends on both.

3. EVALUATION

3.1 NDCG score

We use the normalized discounted cumulative gain measure (NDCG) [10] averaged over the queries to evaluate the performance of the multiple nested ranker algorithm. We choose this performance measure because it incorporates multiple relevance judgements and depends not only on the position but also on the relevance score of the document in the ranked list. Jarvelin et al. [10] showed that NDCG gives more credit to systems with high precision at top ranks than other evaluation measures.

Our data was labeled using 5 degrees of relevance, ranging from "excellent match" to "poor match". To compute the NDCG score, we map the relevance levels to numerical values, with 4 corresponding to the highest level of relevance and 0 corresponding to the lowest level of relevance. Unlabeled documents were given rating 0. Jarvelin et al. [10] used a similar map, with labels ranging from 3 to 0. The labels can be seen as weights or information gain for the user [18]. The difference in gain values assigned to highly relevant and relevant documents changes the NDCG score. Larger ratios put more weight on precision with respect to the highly relevant documents, see [18]. We used a relatively small gain ratio which was sufficiently discriminative in our experiments.

The NDCG score is computed for the sorted list of results for the i^{th} query q_i as follows:

$$\text{NDCG}_{q_i} = N_i \sum_{j=1}^{k} \frac{2^{label(j)} - 1}{log_b(j+1)}$$

where N_i is the normalization constant chosen so that a perfect ordering of the results for the query q_i will receive the score of one. $label(j)$ is the gain value associated with the label of the document at the j^{th} position of the ranked list. In the NDCG formula, the sum computes the cumulative information gain to the user from the already inspected documents. $log_b(j+1)$ is a discounting function that reduces document's gain value as its rank increases. The base of the logarithm, b, controls the amount of the reduction. We used b=2 in our experiments which correspond to a sharper discount. Unlabeled documents affect the NDCG score by changing the ranks of the labeled documents. Since some unlabeled documents may be very relevant, $\text{NDCG}_{q_i} = 1$ is hard to achieve even for a good ranker.

We computed NDCG at the top k=10 document since it is the number of results usually viewed by users. The NDCG score is computed for each query and then averaged.

To obtain an intuition about the change in the NDCG score, consider the following perfect ranking R=[4,4,3,3,2,2,2,1,1,1]. In this case, NDCG(R)=1. When we swap the first result with every of the other labels, we receive the NDCG scores that are plotted in Figure 3. For example, swapping the first label 4 with the label 2 at the position five gives a two percentage points decrease in the NDCG score.

4. DATA

In our experiments we used several thousands queries in English from August 2005 provided by an Internet search engine. We had 26,744 queries, each with up to 2500 returned documents. These documents are the top 2500 results per query as produced by the basic ranking algorithm.

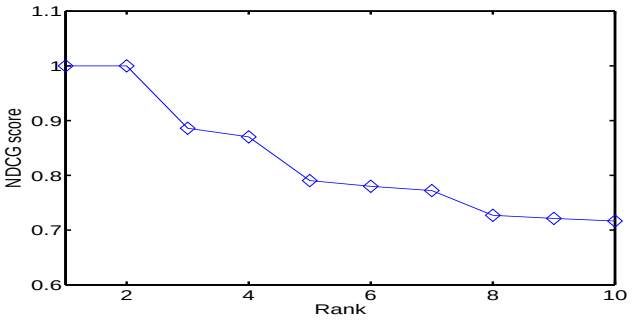

Figure 3: The change in the NDCG score when swapping the fist result in the perfect ranking with each of the other ranks.

Table 1: First data set. Number of queries, number of unlabeled results per query used for training, validation and testing. We used n={1,3,10}.

	Training	Validation	Test
#Queries	3,514	691	688
# Unlabeled	#Labeled*n	1,000	2,500

The document vectors have query-dependent features extracted from the query and four document sources: the anchor text, the URL, the document title and the body of the text. They also have some query-independent features. The document vectors had around 400 features many of which were weighted counts.

For each query there is a number of manually labeled results. As mentioned before, five degrees of relevance were used for labeling this data set, ranging from 4 (meaning "excellent match") to 0 (meaning "poor match"). Unlabeled documents were given label -1. Since originally the labels were produced for evaluation and comparison of top ranked documents, some documents with label 0 are quite relevant. Burges et al. [2] found that adding randomly chosen unlabeled documents as additional examples of low relevance documents to the training set helps to improve the performance. We had a similar approach in our experiments. At each stage of telescoping, we sampled the current training sets for individual queries in the following way. We used all labeled documents and added at random a certain number n of unlabeled results for the same query. This number was a multiple of the total number of labeled results, we used $n = \{1, 3, 10\}$. The unlabeled documents were rated 0 during training.

The number of unlabeled examples included in the training set had a noticeable impact on the performance. We tried a few values of the multiplicative factor n, ranging from 0 to 10. The performance with no unlabeled examples was not satisfactory. The other values gave similar performance; in all experiments described here we included in the training set three times as many unlabeled examples per query as there are labeled.

To speed up the training, we also sampled the original validation set by keeping all the labeled documents and adding at random 1000 unlabeled documents per query. We did

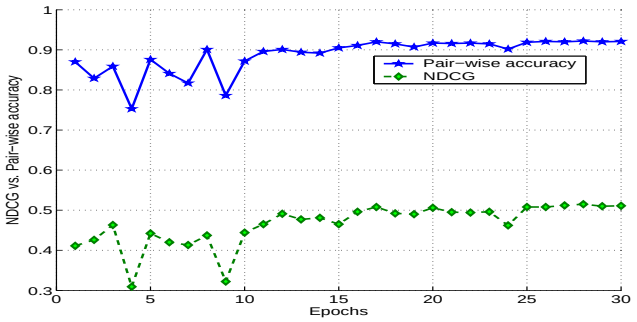

Figure 4: NDCG score vs. pair-wise accuracy on the validation set for 30 training epochs at the first stage of telescoping.

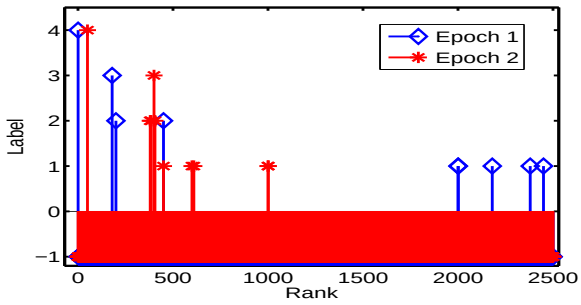

Figure 5: The distribution of labels over ranks for one query between the training epochs 1 and 2.

not change the distribution of the documents in the test set and used all labeled and unlabeled documents available for a given query.

5. EXPERIMENTS ON A SUBSET OF THE DATA

5.1 Small Data Set

To validate our approach, we used a subset of the data in the first set of the experiments, see Table 1.

5.2 Effect of the RankNet Cost Function

As outlined in section 2.1, the cost function of the RankNet algorithm depends on the difference of the outputs of two consecutive training samples. Due to the current form of the cost function, the RankNet algorithm tries to learn the correct pair-wise ordering of the documents regardless of their position in the ranked list. It is, therefore, possible that during training the net improves the pair-wise error by significantly moving up documents that are at the bottom of the list even at the price of slightly moving down some of the relevant results at the top of the list. Telescoping is designed to alleviate this problem by removing the difficult documents at the low ranks and making the ranker to concentrate on the top results.

First, we needed to verify this assumption. Averaged over all queries, the pair-wise error and the NDCG are very well correlated. Figure 4 shows the pair-wise accuracy and the NDCG score on the validation set averaged over the queries after each of the training iterations at the first telescoping stage. In this example, the correlation coefficient is 0.946. However, there are cases where their changes are anti-correlated. For single queries this effect can be quite striking, as illustrated in Figure 5. Figure 5 shows the distribution of labels for a particular query over the ranks after the first and the second epochs of training. As before, label "4" stands for "excellent match", label "0" means "poor match", and label "-1" is used for unlabeled documents.

Figure 5 clearly shows how some documents with label "1", which are poor match, improve their ranks by over 1000 positions in the ranked list, moving from the ranks between 2000 and 2500 to the ranks between 500 and 1000. At the same time, the documents with labels "4", "3" and "2" that were at the top positions after the first epoch of training are moved a few ranks to the left after the second epoch. Thousands of pair-wise errors incurred by placing the documents with label "1" at the position 2000-2500, lower than many documents with label "0", are repaired and there are only few new errors introduced by shifting the highly relevant documents to lower ranks. Therefore, the pair-wise accuracy increases. However, only the positions of these latter documents are important for the NDCG score as well as for the user.

5.3 Results for the Small Data Set

First, we used 4 telescoping stages with the number of top results for telescoping being $N_1 = 2500$, $N_2 = 1000$, $N_3 = 100$, $N_4 = 10$. We included 3 unlabeled examples per each labeled result in the training set at each stage.

Previous results with the RankNet algorithm showed that a two-layer net outperforms the RankNet with a linear net and other related approaches on this task [2]. At the first stage of telescoping, the data sets are not changed, and the net is computed and used on all 2,500 results per query. Therefore, we use the performance of a two-layer net at the first stage as our baseline. We tried different numbers of hidden nodes (nH). As shown in Table 2, on this data set a two-layer net with 4 hidden nodes had the best performance on the validation and the test set after the first stage. Thus, for this data set our baseline is the average NDCG score of 0.451. Using telescoping with a linear net improves the average NDCG score by over 2 percentage points from 0.445 to 0.473. The multiple nested ranker with linear nets outperforms the baseline and also achieves the same or better performance than the multiple nested ranker with two-layer nets for the numbers of the hidden nodes that we tried. Table 2 shows that the multiple nested ranker approach improves the performance for almost all neural net parameters that we tried. The two-layer net with two hidden nodes had the worst performance. However, this net performed similar to the net with four hidden nodes when we used a different proportion of unlabeled results in the training set. The linear net and the two-layer net with $nH = \{4, 8, 16, 32\}$ hidden nodes achieved an over 2 percentage points increase in the NDCG score between the first and the last stages of telescoping. The largest increase was for the $nH = \{16, 32\}$. However, the NDCG of these nets for the first stage of telescoping was lower than for the linear net. There was an insignificant decrease in the NDCG score between the third and the last stages of telescoping for $nH = \{4, 8\}$. These nets achieve an improvement of the NDCG score compared to the first stage of telescoping, showing that our approach

Table 2: First data set. Average NDCG score at the top 10 results for a linear net (nH=0) and a two-layer net with different numbers of hidden nodes nH. Telescoping with 4 stages St.

St	nH	av. NDCG	St	nH	av. NDCG
1	0	0.445 (± 0.023)	1	8	0.436 (± 0.022)
2	0	0.460 (± 0.022)	2	8	0.452 (± 0.022)
3	0	0.470 (± 0.022)	3	8	0.470 (± 0.022)
4	0	0.473 (± 0.022)	4	8	0.468 (± 0.022)
1	2	0.450 (± 0.022)	1	16	0.436 (± 0.023)
2	2	0.455 (± 0.022)	2	16	0.456 (± 0.022)
3	2	0.444 (± 0.023)	3	16	0.466 (± 0.022)
4	2	0.454 (± 0.023)	4	16	0.472 (± 0.022)
1	4	0.451 (± 0.022)	1	32	0.436 (± 0.022)
2	4	0.449 (± 0.022)	2	32	0.443 (± 0.022)
3	4	0.473 (± 0.022)	3	32	0.463 (± 0.022)
4	4	0.469 (± 0.022)	4	32	0.473 (± 0.022)

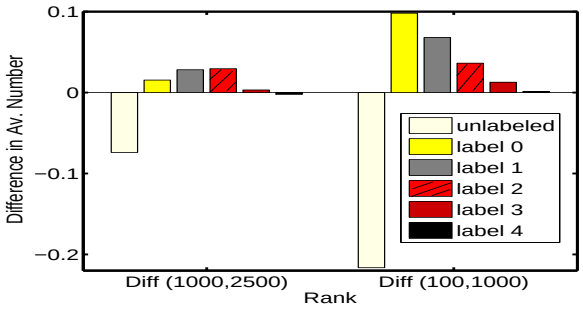

Figure 6: Difference in average numbers of results with particular label in the first top ten results between the first and the second ($Diff(1000, 2500)$) and between the second and the third ($Diff(100, 1000)$) stages.

is not sensitive to the choice of the net parameters.

Figure 6 illustrates the effect of telescoping on the distribution of relevant documents at the top ten positions in the ranked list. It shows the difference in average numbers of results with a particular label between the subsequent stages of telescoping. The actual numerical values of the difference are small because for most queries there are only a few relevant results and not all queries have results with the top 2 levels of relevance. There is no difference between the 3 and 4 stages because these numbers are computed for the top 10 results at the 3 stage which become the training set for the last stage; therefore the last step is omitted in the figure. Compared to the first stage, the number of unlabeled examples at top ten ranks at the second stage decreased and the number of labeled relevant examples increased. Unlabeled examples and low relevance examples with label 0 and do not contribute to the NDCG score directly, but they affect the ranks and thus the contribution of other labeled examples. At the third stage, the number of unlabeled results decreased and the largest increase was for the results with label 0 corresponding to the label "poor match". However, since the number of relevant labeled examples increased as well, the overall NDCG score improved. During training, unlabeled examples are used as examples with label zero. Therefore,

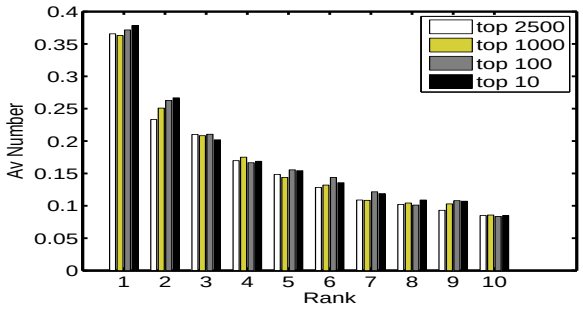

Figure 7: Average number of results with label "excellent" or "good" match at the top ten ranks.

Table 3: NDCG scores averaged over 11 random reshufflings of the training set with the linear net.

Stage	av. NDCG	Stage	av. NDCG
1	0.444 (±0.002)	3	0.470 (±0.001)
2	0.459 (±0.001)	4	0.473 (±0.002)

it is interesting to see that the RankNet prefers to improve the ranks of documents with the label "poor match" but not the ranks of unlabeled documents. This may be attributed to the aforementioned property of labeling. Many of the results with label 0 that are placed among the first 1000 by the first net and among the first 100 by the second net may be in fact quite relevant. Whereas the randomly chosen unlabeled documents are probably not relevant.

Figure 7 shows the average number of results with the two highest levels of relevance at each of the top ten positions in the ranked list. We plotted the results for the test set and for all stages of telescoping. The right most columns correspond to the last stage of telescoping. In five out of ten cases, the top 10 stage has most documents labeled "excellent match" or "good match". In seven out of ten cases, there are more results with label "excellent match" or "good match" in the final ranking than at the first stage which is reflected in the higher NDCG score.

To see whether the multiple nested ranker is sensitive to the initial parameters of the training the neural nets, we ran this experiment over 11 random reshufflings of the training set with the linear net. Table 3 shows the NDCG scores averaged over 11 runs for the test using the linear net. It can be seen that the performance is very stable.

5.4 Importance of Individual Stages

Since the number of telescoping stages that we used in the previous set of experiments was rather arbitrary, we investigated the contribution of each stage individually. As shown in Figure 7, the distribution of the relevant documents at the top of the ranked list after second stage improves relative to the first stage for most of the ranks. This means that the first net separates the relevant and irrelevant documents quite well. It was however, interesting to see, whether the first net can already produce a good separation of relevant documents so that only the top 100 need to be re-ranked. We used telescoping with fewer stages, omitting the second stage with 1000 top ranked results as the training set.

Table 4: First data set. Average NDCG score at the top 10 results for a linear net (nH=0) and a two-layer net with different numbers of hidden nodes (nH). Telescoping with 3 stages *St*.

St	nH	av. NDCG	St	nH	av. NDCG
1	0	0.445 (±0.023)	1	8	0.436 (±0.022)
2	0	0.469 (±0.022)	2	8	0.468 (±0.022)
3	0	0.473 (±0.022)	3	8	0.466 (±0.022)
1	2	0.450 (±0.022)	1	16	0.436 (±0.023)
2	2	0.468 (±0.022)	2	16	0.453 (±0.022)
3	2	0.469 (±0.022)	3	16	0.464 (±0.022)
1	4	0.451 (±0.022)	1	32	0.436 (±0.022)
2	4	0.463 (±0.022)	2	32	0.470 (±0.022)
3	4	0.458 (±0.022)	3	32	0.471 (±0.022)

Table 5: Second data set. Number of queries, number of unlabeled results per query used for training, validation and testing. We used n={1,2,3}.

	Training	Validation	Test
#Queries	23,407	1,132	2,205
# Unlabeled	#Labeled*n	300	2,500

Table 6: Second data set. Average NDCG score at the top 10 results for a linear net (nH=0) and a two-layer net with different numbers of hidden nodes. Telescoping with 4 stages.

St	nH	av. NDCG	St	nH	av. NDCG
1	0	0.462 (± 0.013)	1	4	0.455 (± 0.013)
2	0	0.467 (± 0.013)	2	4	0.470 (± 0.013)
3	0	0.479 (± 0.013)	3	4	0.479 (± 0.013)
4	0	0.483 (± 0.013)	4	4	0.481 (± 0.013)
1	2	0.454 (± 0.013)	1	8	0.444 (± 0.013)
2	2	0.465 (± 0.013)	2	8	0.457 (± 0.013)
3	2	0.477 (± 0.013)	3	8	0.480 (± 0.013)
4	2	0.479 (± 0.013)	4	8	0.483 (± 0.013)

Table 7: Second data set, NDCG scores on the test set, averaged over 5 random reshufflings of the training set for the linear net.

Stage	av. NDCG	Stage	av. NDCG
1	0.459 (±0.001)	3	0.479 (±0.001)
2	0.468 (±0.000)	4	0.481 (±0.001)

Table 4 shows the results. It appears that the first net is sufficient to place relevant documents that can be learned efficiently at the top 100 positions of the ranked list. For all net parameters, the NDCG improvement from the 2500 directly to 100 stage as shown in Table 4 is comparable to the improvement between these two stages when the 1000 stage is used in between.

6. EXPERIMENTS ON THE FULL DATA SET

For our second data set we used the whole training set, a subset of the validation set and the full test set, see Table 5. Again, at each telescoping stage, we sampled the training set by keeping all the labeled documents and adding at random some unlabeled documents. We did not change the test set. Table 6 shows the NDCG scores at each stage of telescoping for this data set. Similar to our first experiments, the multiple nested ranker with a linear net achieves a significant improvement in the NDCG score from 0.461 to 0.483. For this data set, the linear net outperformed all two-layer nets that we tried. On this data set, the two-layer net improves the NDCG score between each stage of telescoping.

We repeated this experiment over 5 random reshufflings of the training set with the linear net. As in the previous case, the multiple nested ranker appears very robust to the initial setting, see Table 7. When we used fewer telescoping stages with a linear net and omitted the second stage with the top 1000 results, the improvement was very similar to the improvement achieved with four telescoping stages, from 0.462 to 0.480. The NDCG score after the first stage with the top 2500 results was 0.462; the NDCG score after the second stage with the top 100 results was 0.467 and the NDCG score with the top 10 results was 0.48.

7. ANALYSIS

The experimental results presented here showed that telescoping improves the precision at the top ranks robustly over the number of settings. The improvement on the large data set was similar to the improvement on the small data set. The exact number of the telescoping stages also did not appear to be crucial for the performance of our approach. The multiple nested ranker with three telescoping stages gave the same improvement as with four stages.

The number of hidden nodes in the two-layer neural net that we used in our experiments is much more important for the performance. However, the relative improvement due to telescoping was over two percentage points for most parameters that we tried. The two exceptions were the net with two hidden nodes for the small data set and the net with four hidden nodes for the small data set with three telescoping stages. In our preliminary experiments, the net with two hidden nodes also showed a two point improvement in the NDCG score on the small data set when we included more unlabeled examples for each training example. The role of the unlabeled examples needs further investigation. They are considered to be labeled as not relevant during the training phase. However, the fact that they are placed among the top 100 and 10 at the last stages of telescoping suggests that they may also be relevant.

Since the training set is pruned after each stage, it is possible that some of the relevant documents are excluded from the following re-ranking. It is not a problem when the major focus is high accuracy. The ranks of those documents remain fixed at each of the following stages. Since all of them were below rank 10, they do not contribute to the NDCG score at any of the stages. In our experiments, the fraction of the relevant documents that were placed at ranks below 1000 after the first stage and below 100 after the second stage was very small. This supports the claim that RankNet produces a good ranking at every stage by placing relevant documents near the top of the results list. The multiple nested ranker approach refines their ranking by improving the ranks of the highly relevant documents.

8. RELATED APPROACHES

The high scoring documents from the top positions on the results list have been used extensively in the variants of relevance feedback to expand the query and compute new weights for the query terms [15, 14]. Although these approaches also perform ranking in stages, at each stage the retrieval algorithm is often applied to the whole document collection.

He et al. [5] used the user-specific information for re-ranking the top n documents. Document's score was changed by some predefined factor on the basis of the genre or domain preference provided by the user. Xiao et al. [19] re-rank the top n results using an additional similarity score computed based on the query and the document title.

Boosting [3] performs learning in stages. A new distribution of the data points is generated at each stage depending on the performance of the weak learner at the previous stage. Boosting puts more weight on learning the examples which were difficult for the previous learner. The aim of telescoping is, on the contrary, to exclude such difficult data points, i.e. low scoring high relevance documents, and to concentrate the learning on the top scoring results. In addition, boosting does not control the size of the training set for the subsequent learner. As we reduce the training set size roughly exponentially, more sophisticated rankers can be used at later stages.

Shen et al. [17] is one of the closely related approaches to re-ranking. They use a variant of the perceptron learning algorithm to learn new ranks for the top scoring results. One variant of their algorithm learns to separate the top r scoring results from the rest. This algorithm can be extended to work in stages similar to telescoping by separating the top $r = \{2500, 1000, 100, 10\}$ results. However, this approach would not exclude low ranking high relevance documents from the training set on later stages.

9. CONCLUSION AND FUTURE WORK

We presented the multiple nested ranker algorithm for an efficient re-ranking of the high scoring results at the top of the ranked list. We applied this approach to real world data. Our experiments showed that at each telescoping stage the RankNet ranker learns the ranking for a new distribution of documents. The ranker concentrates on the relevant documents which are placed near the top of the ranked list. The improvement in the averaged NDCG score confirms that the new sub-problems are easier to learn so that in the end a better ranking of the top few documents is computed.

The fact that the low scoring documents are removed from the training set at later stages of training, can be viewed as an attempt to introduce the information about the rank of the documents into the training procedure of the RankNet algorithm. The next step in the development of this algorithm will be to modify the RankNet algorithm to use this information directly during training.

10. REFERENCES

[1] E. B. Baum and F. Wilczek. Supervised learning of probability distributions by neural networks. *Neural Information Processing Systems*, pages 52–61, 1987.

[2] C. J. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of ICML*, 2005.

[3] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, pages 121(2):256–285, 1995.

[4] Y. Freund, R. Iyer, and R. Shapire. An efficient boosting algorithm for combining preferences. In *Journal of Machine Learning Research*, volume 4, pages 933–969, 2003.

[5] D. He and D. Demner-Fushman. HARD experiment at Maryland: From need negotiation to automated HARD process. In *Proceedings of TREC*, 2003.

[6] A. James. HARD track overview in TREC 2003. In *Proceedings of TREC*, 2003.

[7] A. James. HARD track overview in TREC 2004. In *Proceedings of TREC*, 2004.

[8] B. J. Jansen and A. Spink. An analysis of web documents retrieved and viewed. In *Proceedings of the International Conference on Internet Computing*, pages 65–69, 2003.

[9] B. J. Jansen, A. Spink, J. Bateman, and T. Saracevic. Real life information retrieval: A study of user queries on the web. In *Proceedings of SIGIR*, pages 5–17, 1998.

[10] K. Jarvelin and J. Kekalainen. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of SIGIR*, pages 41–48, 2000.

[11] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceeding of SIGIR*, pages 154–161, 2005.

[12] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Mueller. Efficient backprop. In *Neural Networks: Tricks of the Trade*, pages 9–50. Springer, 1998.

[13] W. Lin, A. Hauptmann, and R. Jin. Web image retrieval reranking with a relevance model. In *Proceedings IEEE/WIC*, 2003.

[14] J. Ponte. Language models for relevance feedback. In W. Croft, editor, *Advances in Information Retrieval*, pages 73–96, 2000.

[15] J. J. Roccio. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice Hall, 1971.

[16] C. Shah and W. B. Croft. Evaluating high accuracy retrieval techniques. In *Proceedings of SIGIR*, pages 2–9, 2004.

[17] L. Shen and A. K. Joshi. Ranking and reranking with perceptron. *Machine Learning*, 60(1-3):73–96, 2005.

[18] E. M. Voorhees. Evaluation by highly relevant documents. In *Proceedings of SIGIR*, pages 74–82, 2001.

[19] Y. Xiao, R. Luk, K. Wong, and K. Kwok. Some experiments with blind feedback and re-ranking for Chinese information retrieval. In *Proceedings NTCIR*, 2005.