Learning to Rank with Incomplete Relevance Judgments

ABSTRACT

Much research in learning to rank has been placed on developing sophisticated learning methods, treating the training set as a given. However, the quality of the training set directly affects the quality of the learned system. Given the expense of obtaining relevance judgments for constructing training data, one often has a limited budget in terms of how many judgments he can get. The major problem then is how to distribute this judgment effort across different queries and how to best utilize the set of available relevance judgments per query. In this paper, we first investigate the tradeoff between the number of queries and the number of judgments per query when training sets are constructed. We show that up to a limit, training sets with more queries but shallow (less) judgments per query are more cost effective than training sets with less queries but deep (more) judgments per query. We then focus on how to better utilize the shallow judgments per query and show that by utilizing the sampling distribution that was used to create these relevance judgments, one can reach significantly better performance than using the relevance judgments directly.

1. INTRODUCTION

Ranking is a central problem in information retrieval. Most modern search engines are based on learning-to-rank: Given a training set composed of queries and associated judged documents, a machine learning procedure learns how to combine the query and the document features in such a way as to effectively assess the relevance of any document to any query and thus rank a collection in response to a user input.

Much thought and research in learning to rank has been placed on feature extraction and the development of sophisticated learning algorithms. However, relatively little research has been conducted on the choice of queries and documents for constructing training data and as to how to better utilize the limited amount of relevance judgments available.

Obtaining relevance judgments for learning is an expensive procedure. When constructing training data for learning to rank, one is often faced with the problem of (1) how to distribute the available judgment budget across different queries so that better performance is obtained, and (2) how to better utilize the limited judgments per query.

One major design decision when training sets are constructed is "Given a fixed judgment budget, is it better to judge as many queries as possible, with fewer judgments per query (shallow judgments) or to judge fewer queries but more documents per query (deeper judgments)?" A similar such question was recently analyzed in context of evaluation in Million Query Track [9]. The results of the track suggest that when the goal is to evaluate the relative quality of search engines, up to a point, it is better to judge more queries with less judgments per query.

In the first part of paper, we ask a similar question for training purposes. Note that training is quite a different task than evaluation as the quality of a training set highly depends on its informativeness. Using LambdaRank [6] and SoftRank [14] as the learning algorithms, we show that similar to evaluation, given a fixed judgment budget for building training sets, better test set performance is obtained when training sets with more queries but shallow judgments per query are used as opposed to training sets with less queries but deep (more) judgments per query. [1]

Having shown that it is better to spend most of the judgment effort on judging more queries, one is then faced with the problem of how to better utilize the limited number of relevance judgments per query. Most state of the art learning to rank algorithms are machine learning methods that work by optimizing the value of an evaluation metric that is assumed to evaluate the quality of a ranked list of documents [14, 6, 17]. It is shown that evaluation metrics are not robust to incomplete judgments [5]. Hence, the quality of the learning to rank algorithms can also be directly affected by the bias in the value of the objective metric introduced by incomplete judgements.

Recently, He et al. [11] analyzed how the quality of learning to rank algorithms are affected by the incompleteness of relevance judgments per query in the training set. Given different evaluation metrics, they showed that at high levels of incompleteness, optimizing for a metric that is more robust to incomplete judgments result in better test performance than optimizing for a less robust metric, suggesting that learning to rank methods should account for the incompleteness in the training data.

Even though He et al. [11] were one of the first ones to ex-

[1] A preliminary version of this work is available as a poster [16].

plore the effect of incomplete judgments in learning to rank, the authors do not propose a solution to the problem that can be applicable given *any* objective metric and *any* sampling distribution that was used to generate the incomplete judgments. They assume that the incomplete judgments are a random subset of complete judgments and they use previous evaluation metrics that are shown to be more robust to incomplete judgments given this setup [15, 5]. Ideally, one would like to have a methodology that can be applicable when the incomplete judgments are generated according to *any* sampling distribution.

While little research has been done regarding the effect of incomplete judgements in learning-to-rank, much research has been devoted to efficient and effective evaluation of retrieval systems given a limited judgment budget [8, 15, 8]. Even though all these methods have their own advantages, the method proposed by Aslam et al. [3] is the one that is guaranteed to compute unbiased estimates of *any* evaluation metric given incomplete judgments and *any* sampling distribution that was used to generate these incomplete judgments. In the second part of this paper, we show that the sampling distribution that was used to generate the incomplete relevance judgments can also be used to compute unbiased estimates of objective evaluation measures for use in learning-to-rank, or (for gradient-descent based learning algorithms) unbiased estimates of gradients.

2. DISTRIBUTING JUDGMENT EFFORT ACROSS DIFFERENT QUERIES

In order to test the effect of reducing the number of queries and the number of judgments per query in the quality of learning algorithms, we mainly focus on LambdaRank [7] and SoftRank [14] as the learning algorithms. Both of these algorithms are neural network based algorithms originally designed to optimize the nonsmooth evaluation metric NDCG [12].

NDCG is one of the most commonly used evaluation metrics in learning to rank as it is one of the few metrics that can incorporate graded relevance judgments. It is defined as the sum of the discounted cumulative gains divided by maximum such value. Most commonly used discount and gain functions used to compute NDCG are $log_2(r+1)$ (where r is the rank a document is retrieved) and $2^{rel(i)}-1$ (where $rel(i)$ is the relevance of document i), respectively.

Another commonly used metric in learning to rank is average precision (AP) [17]. Average precision is defined as the average of precisions at relevant documents. If a system did not retrieve all relevant documents, the remaining documents are assumed to be retrieved at infinity (hence, having precisions 0).

Since the behavior of learning algorithms with respect to the number of queries and the number of judgments per query may vary depending on the objective metric used, we use both average precision and NDCG as the objective metrics and extend these two learning algorithms to optimize for average precision. In the following section, we first describe the two learning to rank algorithms and describe how they can be extended to optimize for average precision.

2.1 LambdaRank

Information retrieval metrics are not smooth as they mainly depend on the ranks of documents, which are discontinuous. In order to overcome the problem of optimizing non-smooth IR metrics, LambdaRank uses the approach of defining the gradient of the target evaluation metric only at the positions needed.

Given a pair of documents, the virtual gradients (λ functions) used in LambdaRank are obtained by scaling the RankNet [6] cost with the amount of change in the value of the metric obtained by swapping the two documents. Following the same setup, in order to optimize for the average precision metric, we scale the RankNet cost with the amount of change in the value of the corresponding metric when the two documents are swapped. This way of building the gradients in LambdaRank is shown to find the local optima for the target metrics and can be applied to both AP and NDCG [10].

2.2 SoftRank

The main idea used in SoftRank for optimizing non-smooth IR metrics is based on defining smooth versions of information retrieval metrics by assuming that the score s_j of each document j is a value generated according to a Gaussian distribution with mean equal to s_j and shared smoothing variance σ_s. Based on this, Taylor et al. [14] then define π_{ij}, the probability that document i will beat document j and the rank distribution $p_j(r)$, the probability that document j will be at rank r. These distributions can then be used to define smooth versions of IR metrics as expectations over these rank distributions.

Given these definitions, we now extend SoftRank to optimize for average precision by defining SoftAP, the *expected* average precision with respect to these distributions and compute the gradient of SoftAP.

In order to define soft average precision, consider the random experiment corresponding to average precision, as defined by Yilmaz and Aslam [15]:

1. Pick a relevant document at random. Let the rank of this document be r.
2. Pick a document that is ranked at or above r, at random.
3. Output the relevance of this document.

Given the distribution π_{ij}, the probability that document i will beat document j, SoftAP can then be defined as:

$$SoftAP = \frac{1}{R}\sum_{j=1}^{N} rel(j)\cdot\frac{rel(j)+\sum_{i=1,i\neq j}^{N}\pi_{ij}rel(i)}{\sum_{i=1,i\neq j}^{N}\pi_{ij}+1} \quad (1)$$

where R is the number of relevant documents in the query and N is the total number of documents in the collection.

Using the same approach as in SoftRank and using the chain rule for obtaining derivatives, the derivative of $SoftAP$ with respect to the score of document $\bar{s}_m$ can be written as

$$\frac{\partial SoftAP}{\partial \bar{s}_m} = \sum_{j=1}^{N}\sum_{k=1}^{N}\frac{\partial AP}{\partial \pi_{jk}}\cdot\frac{\partial \pi_{jk}}{\partial \bar{s}_m}+\frac{\partial SoftAP}{\partial \pi_{kj}}\cdot\frac{\partial \pi_{kj}}{\partial \bar{s}_m}$$

$$\frac{\partial SoftAP}{\partial \pi_{kj}} = \frac{rel(j)}{R}\cdot\frac{rel(k)}{\sum_{i=1,i\neq j}^{N}\pi_{ij}+1} - \frac{rel(j)}{R}\cdot\frac{rel(j)+\sum_{i=1,i\neq j}^{N}\pi_{ij}rel(i)}{\left(\sum_{i=1,i\neq j}^{N}\pi_{ij}+1\right)^2}$$

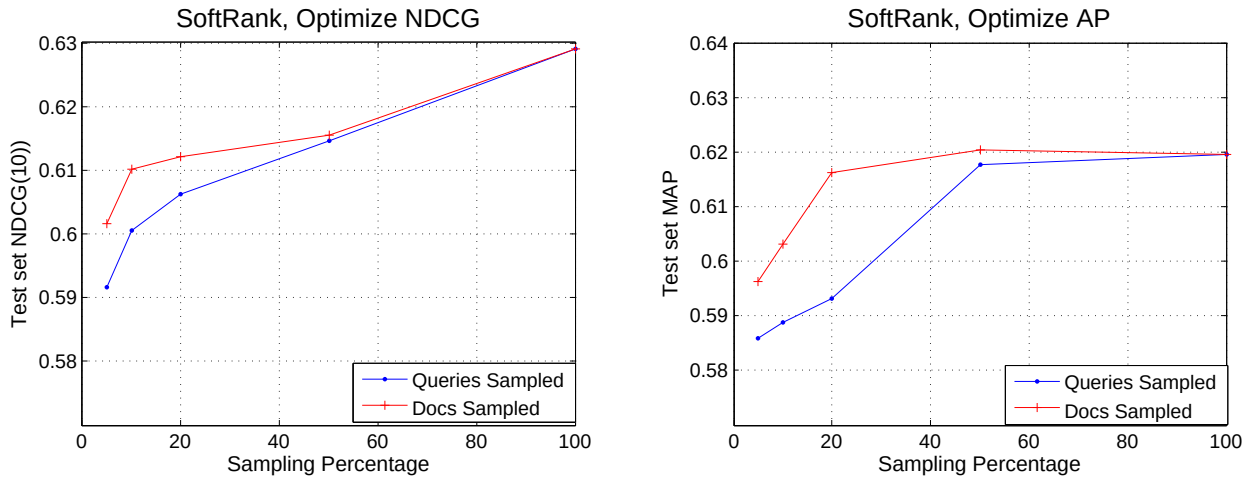

Figure 1: Test set NDCG(10) and MAP values, training on $p\%$ sampled training sets formed by sampling queries vs. sampling documents using SoftRank as the learning algorithm.

and $\frac{\partial \pi_{kj}}{\partial \bar{s}_m}$ can be computed as described by Taylor et al. [14].

2.3 Deep versus Shallow Judgments in Learning to Rank

In order to test the effect of reducing the number of queries vs. the number of judged documents per query, we use data obtained from a commercial search engine. The dataset contains 382 features and is split into train, validation and test sets with 2K, 1K, and 2K queries, respectively. The average number of judged documents in the training set is 350 per query, the number highly varying across different queries. The documents in the dataset are assigned judgments with 5 grades, between 0 (nonrelevant) to 4 (most relevant). In order to compute average precision, we convert these judgments to binary by assuming that documents with relevance>0 are all relevant and the rest are nonrelevant. We use the commonly used log discount with the exponential gain function to compute NDCG.

To test the effect of judging more queries vs. more documents per query, we form different training sets by (1) sampling $p\%$ of queries, keeping the number of documents per query fixed to the maximum available and (2) sampling $p\%$ of documents per query, keeping the number of queries fixed. We then train the LambdaRank and SoftRank algorithms using these different training sets using NDCG and AP as the objective metrics and compute the test set AP and NDCG(10) values. In order to avoid the variance in results due to sampling, we repeat the procedure 5 times and report the average NDCG(10) and AP values over 5 different samples.

Figure 1 and Figure 2 show the result of this experiment for different sampling percentages. The x axis in the figures shows the sampling percentage; the line with the plus marks corresponds to sampling documents and the dotted line corresponds to sampling queries. It can be seen that given a fixed sampling percentage (thus a fixed judgment budget), sampling queries for the training set results in worse performance than sampling documents per query. The behavior is consistent for both learning algorithms and both evaluation metrics.

These results suggest that our training set should contain as many queries as possible with a few judged documents. The question, then, is how to pick these judged documents on per query basis and how to best utilize the limited number of available judgments per query. In the following sections, we mainly focus on this problem and propose a method based on statistical estimation that can be used to utilize the unjudged documents as well as the limited judged documents for more effective learning.

First, we compare two of the most commonly used methods for document selection, depth-k pooling and sampling and show that especially when very limited number of documents are judged on per query basis, sampling (with a predefined sampling distribution) is better than depth pooling. We then describe the statistical estimation method and show that this method results in further improvements in the test set performance when compared to sampling.

3. LEARNING TO RANK WITH DEPTH POOLING VS. SAMPLING

In the previous section we assumed that the reduced judgment sets are formed by randomly sampling documents from a bigger collection of documents. In reality, the most commonly used method for reducing judgment effort is depth-k pooling where top k documents from different systems are

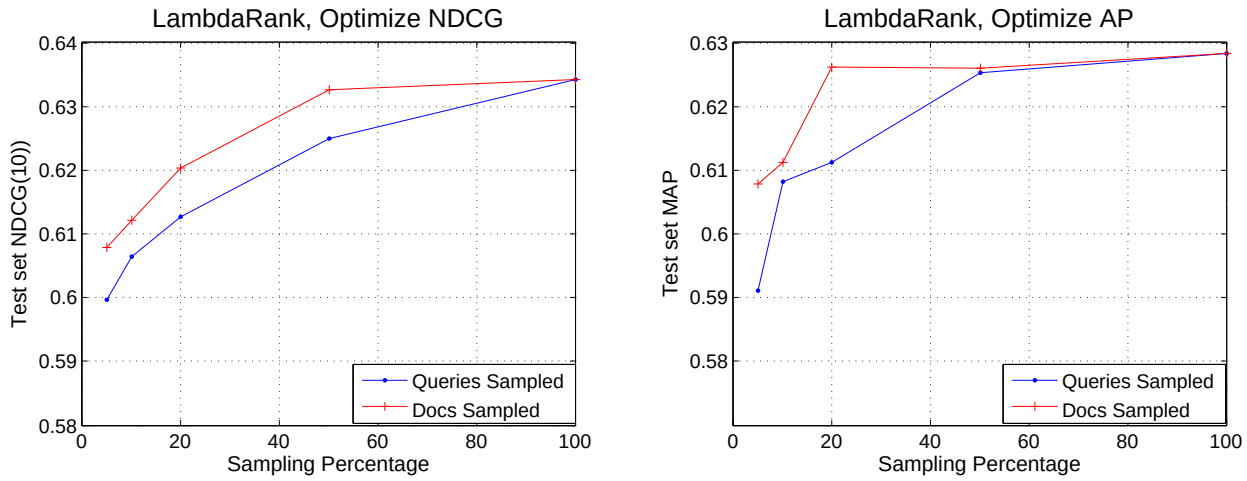

Figure 2: Test set NDCG(10) and MAP values, training on $p\%$ sampled training sets formed by sampling queries vs. sampling documents using LambdaRank as the learning algorithm.

combined and judged for relevance. The remaining documents are often assumed to be nonrelevant, or may be ignored for training purposes. In the previous section, we did not have access to the systems that were used to create the initial pools. Hence, depth-k pooling could not be employed. In the following sections, we instead use TREC data as we have access to the ranked lists of systems that can be used to create the depth-k pools. [2]

In our experiments, we use the TREC collection that was used by Aslam et al. [2] to compare the effect of different document selection methodologies for learning to rank. The document corpus, the queries and the relevance judgments in this collection are obtained from TREC 6, 7 and 8 Adhoc retrieval track. Collectively, the dataset contains 150 queries. A number of retrieval systems were run over these queries and for each query the depth-100 pools of the returned documents were judged as relevant or nonrelevant.

The *effectively* complete dataset we use in our experiments contain the documents from the depth-100 pools. The features used are shown in Table 3 and they are a subset of the LETOR3.0 features [13]. The description of these features along with their exact formulas can be found in the LETOR3.0 documentation [13]. Each feature is computed over the document text (without the title) and over the document text combined with the title, resulting in 22 features in total. The data is then partitioned into five parts in order to conduct five-fold cross validation. For each fold, three parts are used for training, one part for validation and one part for testing.

Using this collection, Aslam et al. [2] compared different document selection methodologies for learning to rank and showed that using as few as 2% of the judged documents (corresponding to the total number of documents depth-1 pool, where the total number of judged documents on average per query is 34) similar results to training using all the documents can be obtained. These results suggest that it may be possible and more cost effective to further reduce the judgment effort per query.

In our experiments, we focus on depth-k (where $k \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$) pooling and sampling as techniques that can be used to reduce judgment effort. The sampling method is based on sampling documents from the complete depth pool given a predefined sampling distribution. The sampling distribution used is based on a prior of relevance induced by the average precision measure, where each document is selected with probability roughly proportional with its likelihood of relevance [3]. [3]

	Features
1.	BM25
2.	LogBM25_Feature
3.	LM_ABS_Feature
4.	LM_DIR_Feature
5.	LM_JM_Feature
6.	LogNormalizedTF_Feature
7.	SumLogTF_Feature
8.	TF_Feature
9.	TF_IDF_Feature
10.	LogTF_IDF_V2_Feature
11.	NormalizedTF_Feature

Table 1: Feature Set

[2]As opposed to the dataset from the commercial search engine, TREC data has much fewer queries. Hence, it would not be feasible to use the TREC data for the experiments comparing deep versus shallow judgments.

[3]The conclusions in this section apply for the particular sam-

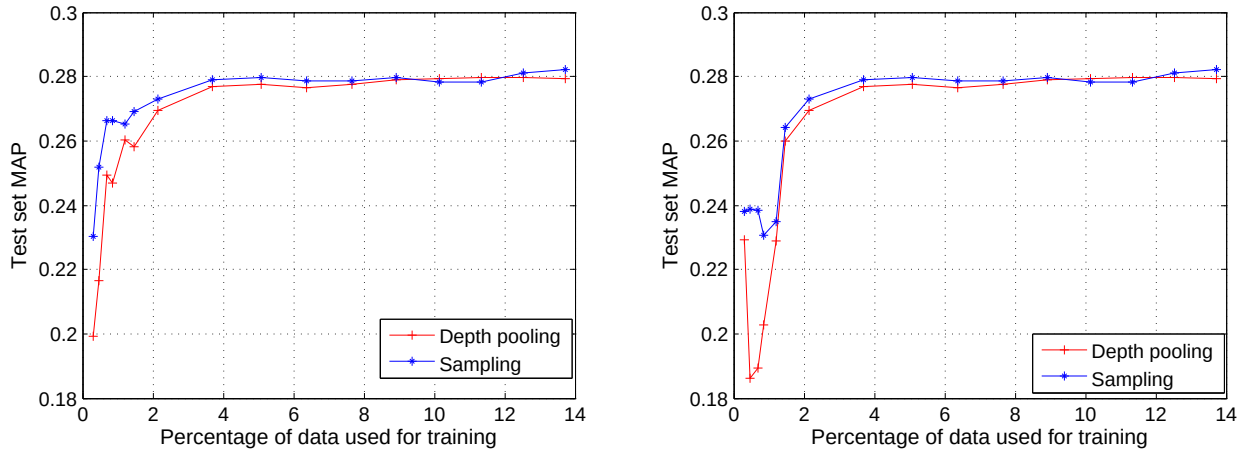

Figure 3: Depth pooling vs Sampling when unjudged documents are (left) removed from training data (right) assumed to be nonrelevant.

Since depth-1 pooling is the smallest depth pool one can achieve given the number of systems that contribute to the pool, one possibility of reducing the judgment effort further is to reduce the number of systems that contribute to the depth pool. Hence, to create even fewer judgments than the 2% of judgments, we randomly sample p% ($p \in \{5, 10, 20, 30, 40, 50\}$) of *systems* and get the documents in depth-1 pools of these systems. In order to be fair in comparison, when the sampling technique is used, we sample equivalent number of judgments from the depth-100 pools of the p% systems that were used to create the corresponding depth-1 pools.

Since the TREC collection has binary relevance judgments, we use Average Precision as the evaluation metric, mainly focusing on SoftRank as the learning algorithm as it has the advantage of directly optimizing the objective metric.

Figure 3 shows the effect of reducing the number of judgments when sampling and depth-k pooling are used for selecting the judged documents. The x axis in the plot shows the percentage of pool judged to create the training data (keeping the test set fixed) and the y axis shows the test set MAP values when the two strategies are used for selecting documents. The training sets containing less than 2% of the documents are created by sampling systems as described earlier. For the plot on the left, we only use the judged documents in training and remove the unjudged documents from the training sets; whereas for the plot on the right we assume that unjudged documents are nonrelevant (a commonly used assumption in depth pooling).

The plots show that when all the systems are used (up to 2% of the collection), reducing the number of documents per query has no significant effect on the test set performance. This is consistent with the results of Aslam et al [3]. Applying sign-rank test with $p = 0.05$ shows that when the training set is further reduced by sampling systems, then the test set performance tends to get significantly worse. Furthermore, in both plots it can be seen that sampling method outperforms depth pooling. The differences are statistically significant when very small amount of training data is available.

When the two plots are compared, it can be seen that making no assumptions regarding the relevance of unjudged documents (left plot) tends to give better test set results than assuming that unjudged documents are nonrelevant. This is consistent with the results from evaluation as evaluation measures tend to be more robust to incomplete judgments when the unjudged documents are removed from the collection [4, 15].

Note that reducing the number of documents by reducing the number of systems that contribute to the pool has a slightly different effect than reducing the depth used to create the depth-k pools. Having different systems contribute to the pools is important as they introduce a variety of documents to be used in training. Hence, by reducing the number of systems that contribute to the pool, we are in effect not only decreasing the number of judged documents but also possibly the variety of documents in the training data.

The two plots show that including a variety of systems to create the pools have a significant impact in the quality of training sets. This is particularly important for researchers who would like to build training sets outside of TREC as they often do not have access to a high variety of systems. There are many important questions that need to be answered regarding this: What is the minimal number of systems that should be used when the training data is created? Which systems are better than others for constructing better

pling distribution used. Different conclusions can be reached if different sampling distribution was used.

training data? These questions are all highly important; and in the future we plan to further investigate these questions.

We have shown that sampling could be a better alternative to depth pooling especially when very limited number of judgments are available. In our experiments, the sampling distribution that was used to create the relevance judgments was completely ignored. We will now show how one can utilize both judged and unjudged documents in the collection by making use of the sampling distribution that was used to create the judged documents, resulting in better effectiveness as compared to using sampled documents alone.

4. LEARNING TO RANK WITH PARTIALLY JUDGED COLLECTIONS

Recently, Aslam et al. [3] proposed a new statistical method that can be used to produce unbiased values of evaluation metrics with respect to a partially judged collection of documents given *any* sampling distribution that was used to create the partial (incomplete) relevance judgments. The main ideas of the method can be summarized as follows:

1. For a measure you need to estimate, define a probability distribution and a random variable such that the expectation of the random variable with respect to the probability distribution is the value of the measure.

2. Since the value of the measure can be seen as an expectation of a random variable, one can use sampling to estimate the value of the measure.

3. Usually, the relevance judgments are generated according to *some* distribution that does not necessarily match the distribution necessary to estimate the desired expectation. Hence, one needs to estimate the expected value of a random variable with respect to one distribution using a sample drawn according to a different distribution. Scaling factors from the method of importance sampling [1] can be used to correct for the difference between two distributions.

The method is applicable for estimating *any* evaluation measure that can be defined as an expectation. Aslam et al. showed how the above approach can be used to compute unbiased estimates of average precision.

We will now show how the corresponding method can be used for learning to rank when only a limited number of judged documents are available. We assume that the training data contains some limited number of judged documents that are generated according to some sampling distribution from a bigger pool (complete pool) of documents. Training is then done over the complete pool and the judged documents are used to estimate the values of the objective metrics (as well as the gradients of these metrics) if the complete pool was judged. Hence, the proposed technique can utilize both judged and unjudged documents during training.

In the following sections, we first briefly describe how the above statistical method is used to compute unbiased estimates of average precision given limited judgments and based on ideas from the statistical method we then extend SoftRank to encorporate the judged as well as unjudged documents by utilizing the sampling distribution that was used to generate the available judgments.

4.1 Estimating Average Precision

Average precision is defined as the sum of precisions at relevant documents, divided by the total number of relevant documents in the collection (R). Let SP be the sum of precisions at relevant documents. Aslam et al. compute the estimate of AP ($\hat{AP}$) by first computing the estimate of SP ($\hat{SP}$) and dividing this value by the estimate of R ($\hat{R}$) [4].

Abusing the notation used by Aslam et al., let N be the total number of documents in the complete collection and let $rel(i)$ be the binary relevance of the document at rank i such that $rel(i) = 0$ when document at rank i is nonrelevant and $rel(i) = 1$ if it is relevant. Then, SP can be written as:

$$\begin{aligned} SP &= \sum_{i=1}^{N} rel(i) \cdot PC(i) \\ &= \sum_{i=1}^{N} rel(i) \sum_{j=1}^{i} rel(j)/i = \sum_{1 \le j \le i \le N} \frac{1}{i} \cdot rel(i) \cdot rel(j) \end{aligned}$$

The approach is based on viewing the above formula as an expectation over pairs of ranks (i, j) by viewing $rel(i) \cdot rel(j)$ as a random variable and the associated weight $1/i$ as the associated probability distribution (appropriately normalized so that the sum equals to 1). Let X be a random variable corresponding to the product of relevances $rel(i) \cdot rel(j)$ and let JD be a probability distribution $JD(i, j) = \frac{1/i}{H_N}$, where $H_N = \sum_{i=1}^{N} \sum_{j=1}^{i} 1/i = N$ is the normalization factor used to convert the weights into a probability distribution.

Given this definition, SP can be computed as $SP = N \cdot E[X]$, where the expectation is computed with respect to the joint distribution JD. Based on this, if U is a multiset of pairs drawn according to JD, SP can be estimated as:

$$\widehat{SP} = N \cdot \frac{1}{|U|} \sum_{(i,j) \in U} rel(i) \cdot rel(j)$$

Usually, the relevance judgments are generated according to *some* distribution that does not necessarily match the distribution necessary to estimate the desired expectation. In the case of training for example, the ranked lists returned in response to a query often change during each training epoch. Hence, during each epoch, there would be a different sampling distribution required to compute the above expectation.

Ideally, given a fixed sampling distribution and the relevance judgments generated according to this sampling distribution, we would like to use this distribution to compute the above expectation regardless of the distribution required. Hence, we would like to compute the expected value above using a sample drawn according to a different distribution than the one necessary to estimate the desired expectations. In order to do this, Aslam et al. employ *scaling factors* that are used to correct for the differences between the actual and the required sampling distributions.

Let $M(i)$ be the effective sampling distribution that is used to sample the individual judged documents and let $JD(i, j)$ be the joint distribution that is required to estimate

[4]Note that even since the expectation of a fraction is not equal to the fraction of the expectations, the estimates obtained this way may be slightly biased. However, the bias introduced due to this fact was shown not to affect the estimates so much.

the proper expectation, [5] where $JD(i,j)$ is now defined over documents i and j as opposed to ranks (i.e., $JD(i,j)$ is the joint distribution over documents i and j). Then, SP can be estimated as:

$$\widehat{SP} = N \cdot \frac{1}{|U|} \sum_{(i,j)\in U} rel(i) \cdot rel(j) \cdot SF(i,j)$$

where $SF(i,j)$ is the scaling factor that corresponds to the ratio between the required and the effective sampling distributions. Let K be the total number of sampled documents. Then, the scaling factors can be computed as:

$$SF_s(i,j) = \frac{JD(i,j)}{I(i,j)}. \quad (2)$$

and I(i,j) can be computed as:

$$I(i,j) = \frac{K-1}{K} M(i)M(j) \text{ if } i \neq j$$
$$I(i,i) = \frac{1}{K} M(i) \left(1 + (K-1)M(i) + (1-M(i))^{K-1}\right).$$

Based on this idea, it is easy to show that if S is the set of judged documents sampled according to distribution M, the total number of relevant documents in the collection, R can be estimated as $\widehat{R} = \frac{N}{|S|} \sum_{i\in S} rel(i) \cdot SF(i)$, where $SF(i) = \frac{1/N}{M(i)}$.

4.2 SoftRank with Incomplete Judgments

In the previous section, we described the method by Aslam et al. [3] that can be used to compute unbiased estimates of evaluation metrics with respect to a *complete* collection given a limited number of relevance judgments and the sampling distribution that was used to generate these relevance judgments. Although we focus on average precision as the evaluation metric, the method can be used to compute unbiased estimates of *any* evaluation metric that can be defined as an expectation. Furthermore, the estimates produced by the method are guaranteed to be unbiased regardless of the sampling distribution used.

In this section, we will show how the method can be used for training purposes using SoftRank as the learning algorithm and AP as the evaluation metric. Due to the properties of the method, same approach is applicable for optimizing *any* evaluation metric given *any* sampling distribution.

As shown in Equation 1, given a probability distribution π_{ij}, the probability that document i will beat (will be ranked above) document j SoftAP can be computed as:

$$SoftAP = \frac{1}{R} \sum_{j=1}^{N} rel(j) \cdot \frac{rel(j) + \sum_{i=1,i\neq j}^{N} \pi_{ij} rel(i)}{\sum_{i=1,i\neq j}^{N} \pi_{ij} + 1}$$

Using the approach described above to estimate the value of average precision, given a document collection D, a set of documents S sampled from D nd the associated sampling distribution M, one can show that SoftAP can be estimated as follows:

[5]The method proposed by Aslam et al. can also be used when the systems do not retrieve all the documents in the complete collection. Since during training the learning algorithm returns all the documents in the training set, this is not an issue when the method is used for training purposes. Hence, for the purposes of this paper, we assume that all the documents in the collection are retrieved.

Let PC_j be

$$PC_j = \frac{rel(j) \cdot SF(j,j) + \sum_{i\in S, i\neq j} \pi_{ij} rel(i) SF(i,j)}{\left(\sum_{i\in D, i\neq j}^{N} \pi_{ij} + 1\right)}$$

Then, $\widehat{SoftAP}$ can be written as:

$$\widehat{SoftAP} = \frac{1}{\widehat{R}} \sum_{j\in S} rel(j) \cdot PC_j$$

, where $SF(i,j)$ can be computed as in Equation 2.

Note that even though the numerator of the above formula only depends on the sampled documents, the denominator depends on all the documents in the collection D. Therefore, the unjudged documents also have an effect in the computation even though their relevances are not used.

The gradient of $\widehat{SoftAP}$ with respect to the distribution π_{kj} can then be computed as follows:

If both documents k and j are sampled, then the gradient with respect to π_{kj} is:

$$\frac{\partial \widehat{SoftAP}}{\partial \pi_{kj}} = \frac{rel(j)}{\widehat{R}} \cdot \left[\pi_{kj} \frac{rel(k) \cdot SF(k,j)}{\sum_{i\in D, i\neq j} \pi_{ij} + 1} - \frac{PC_j}{\sum_{i\in D, i\neq j} \pi_{ij} + 1} \right]$$

If either document k or document j is not sampled, then

$$\frac{\partial \widehat{SoftAP}}{\partial \pi_{kj}} = \begin{cases} 0 & \text{if } j \notin S \\ -\frac{rel(j)}{\widehat{R}} \cdot \frac{PC_j}{\sum_{i\in D, i\neq j} \pi_{ij} + 1} & \text{if } j \in S,\ k \notin S \end{cases}$$

and $\frac{\partial \pi_{kj}}{\partial \bar{s}_m}$ can again be computed as described by Taylor et al. [14].

Given these definitions, we can now employ SoftRank to train a ranking function over a partially judged collection (where the partial judgments are generated using a known sampling distribution), while optimizing for the estimated value of an evaluation metric with respect to the complete collection.

4.3 Results

In this section, we compare the quality of the resulting ranking function trained by SoftRank over a partially judged collection using the afore-described approach of measure estimation, with the quality of the ranking function when optimized by the mere use of the incompletely judged data.

In our experiments here, we use the same sample of judgments (pools) that were generated according to the sampling distribution described in Section 3 so that the results we obtain are comparable to both the sampling and depth pooling methods. We then employ the above technique for training using the partially judged collection by utilizing the estimated value of the AP with respect to the complete collection as the objective metric (referred to as *Estimated AP*). Since directly using the sampled judgments to compute the value of the objective metric without considering the sampling distribution (referred to as *Sampled Pool AP*) was previously shown to outperform depth pooling, here we only compare the two sampling based approaches, *Estimated AP* versus *Sampled Pool AP*.

Figure 4 illustrates how the two methods compare. As before, the left hand side plot corresponds to the case where unjudged documents are not considered in training based on Sampled Pool AP, while in the right-hand side plot unjudged documents are used in training assumed to be non-relevant.

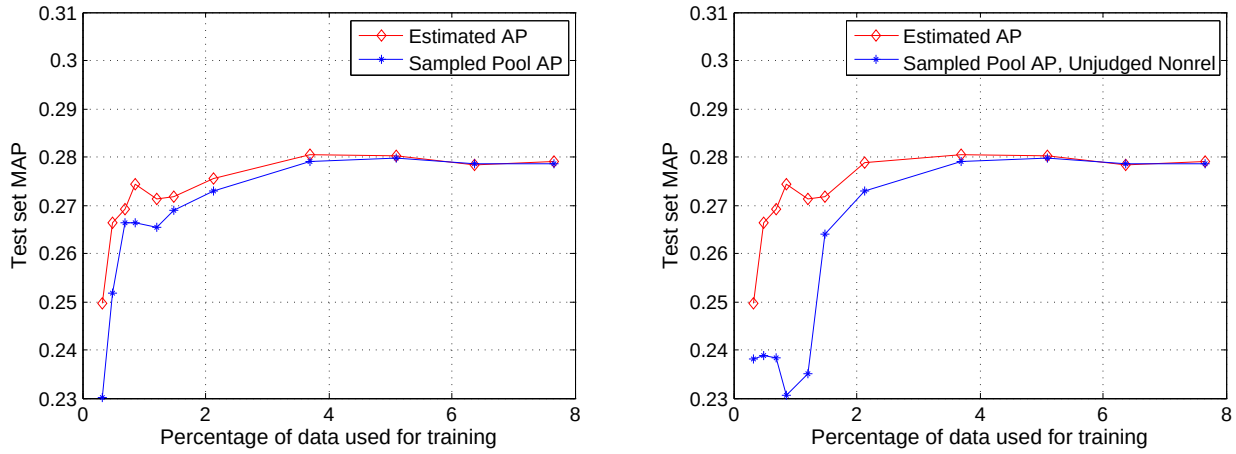

Figure 4: The effect of using the estimated value of average precision given the judged documents and the sampling distribution vs. (left) using only the sampled documents, and (right) using only the sampled documents while assuming unjudged document are nonrelevant.

The Estimated AP approach does not need to make any such assumption since it uses the unbiased estimator over the complete pool and thus the curves corresponding to Estimated AP are exactly the same in both plots. It can be seen that optimizing for Estimated AP consistently outperforms optimizing for Sampled Pool AP, and the difference is statistically significant for smaller sampling percentages ($p \leq 0.05$).

To get a better insight over the results illustrated in Figure 4 and further demonstrate the advantages of the Estimated AP during training, we investigate how the value of actual average precision (Actual AP) changes as compared to the value of the objective metric during each training epoch. Actual average precision is the average precision value that would be computed if the training was terminated at a specific epoch and the learning algorithm was run on the training set with complete judgments.

Figure 5 shows how the value of Actual AP changes as compared to the computed value of the objective metric during each training epoch. The sampled judgments were created by sampling 2% of the complete pool. The left, middle and right plots in the figure show the change in the value of Actual AP when (a) Sampled Pool AP, (b) Sampled Pool AP assuming unjudged documents are nonrelevant, and (c) Estimated AP are used as the objective metrics, respectively. For comparison purposes, the plots also show the change in the value of the corresponding objective metric. It can be seen that in general the value of Actual AP and the objective metrics tend to change in a similar way, with Estimated AP being the closest approximation to Actual AP.

In order to better examine both the direction of change and the extend of change between Actual AP and the different objectives, Figure 6 shows the scatter plot of the objective metrics and the actual AP values from Figure 5. The plots also contain the RMS error (how correlated are the values?) and the Kendall's τ (how correlated are the rankings induced by the values?) statistic between the values of objective metrics and Actual AP.

Note that achieving a high Kendall's τ value as compared to actual AP is particularly important as it shows that an increase (or decrease) in the value of the objective metric also results in an increase (or decrease) in the value of Actual AP. This would suggest that the objective metric guides the ranker in the same direction as Actual AP during gradient ascent. It can be seen using Estimated AP with as few as 2% of judgments as the objective metric in optimization is very similar to using actual AP with the complete judgments as the objective. Note that the Kendall's τ values are directly correlated with the test set performance obtained using each of the metrics as the objective, with Estimated AP being the best, followed by Sampled Pool AP (Figure4, left plot), followed by Sampled Pool AP assuming unjudged documents are nonrelevant (Figure4, right plot).

Therefore, Estimated AP is not just a surrogate of the Actual AP but an unbiased estimator that can lead the optimization towards the right direction in an efficient and effective way and thus leads to better performance ranking functions.

5. CONCLUSIONS

Learning to rank lies in the core of information retrieval. Much research in learning to rank has been devoted to developing sophisticated learning to rank algorithms, treating the training set as a given. Obtaining relevance judgments to construct the training sets is an expensive procedure and one is often faced with the problem of learning to rank when

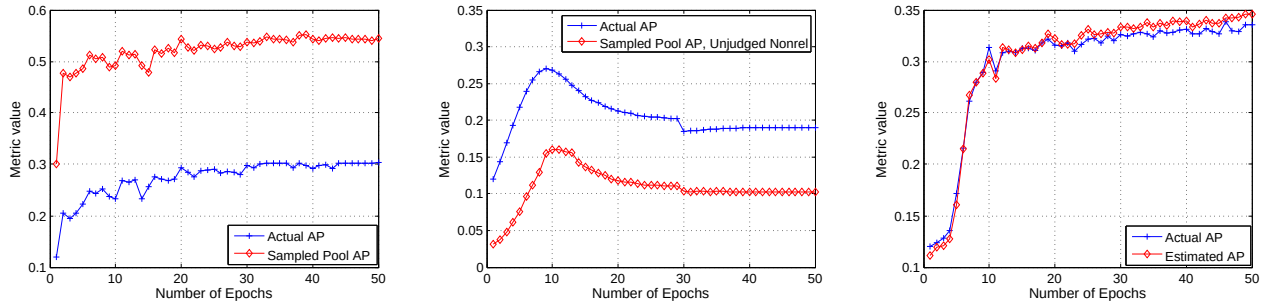

Figure 5: Change in value of Actual AP when optimizing for (left) Sampled Pool AP, (middle) Sampled Pool AP assuming unjudged documents are nonrelevant, and (right) Estimated AP as compared to the value of the corresponding target metric during each training epoch.

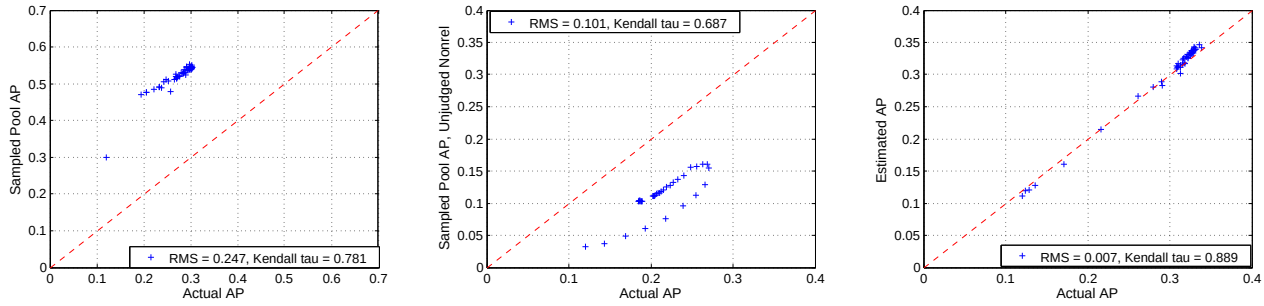

Figure 6: Actual AP vs. the value of the metric used in optimization: (left) Sampled Pool AP, (middle) Sampled Pool AP assuming unjudged documents are nonrelevant, and (right) Estimated AP.

a limited judgment budget is available.

When training with a limited judgment budget, one is often faced with two important problems: (1) How to distribute the limited judgment effort across different queries, and (2) how to best utilize the limited number of judgments available per query. In this paper, we investigated both of these problems.

First, we focused on the problem of distributing the judgment effort across different queries and showed that training sets that contain more queries with shallow judgments per query are more cost effective than training sets that contain less queries but deep (more) judgments per query. These results suggest that training sets should contain queries with very few judged documents. Hence, learning to rank methods should be able to cope with the very limited number of judgments available per query.

In the second part of this paper, we focused on learning to rank when each query is partially judged. First, using SoftRank as the learning algorithm, we compared two commonly used methods for document selection, sampling and depth-k pooling, and showed that given a particular sampling distribution, sampling outperforms depth pooling when the number of judgments in the collection is very limited. Then, we described a technique based on statistical estimation that can be used to utilize both judged and unjudged documents in learning to rank. The proposed technique is based on using statistical estimation techniques to compute unbiased estimates of objective evaluation metrics in learning to rank given the sampling distribution that was used to create the limited relevance judgments. Using this method, we showed that one can obtain significant improvements over utilizing just the judged documents during training.

The approach described in this paper is very general as it is applicable to *any* sampling distribution that was used to create the limited relevance judgments and *any* objective evaluation metric. In this paper, we mainly focused on average precision as the objective metric and SoftRank as the learning algorithm. In the future, we plan to extend different learning algorithms for optimizing different evaluation metrics when only partial judgments are available per query.

6. REFERENCES

[1] E. C. Anderson. Monte carlo methods and importance sampling. Lecture Notes for Statistical Genetics, October 1999.

[2] J. A. Aslam, E. Kanoulas, V. Pavlu, and E. Yilmaz. Document selection methodologies for efficient and effective learning-to-rank. In *SIGIR '09: Proceedings of the 32nd annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, 2009. ACM. To appear.

[3] J. A. Aslam, V. Pavlu, and E. Yilmaz. A statistical method for system evaluation using incomplete judgments. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 541–548, New York, NY, USA, 2006. ACM.

[4] T. Bompada, C.-C. Chang, J. Chen, R. Kumar, and R. Shenoy. On the robustness of relevance measures with incomplete judgments. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 359–366, New York, NY, USA, 2007. ACM.

[5] C. Buckley and E. M. Voorhees. Retrieval evaluation with incomplete information. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 25–32, New York, NY, USA, 2004. ACM.

[6] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 89–96, New York, NY, USA, 2005. ACM.

[7] C. J. C. Burges, R. Ragno, and Q. V. Le. Learning to rank with nonsmooth cost functions. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *NIPS*, pages 193–200. MIT Press, 2006.

[8] B. Carterette, J. Allan, and R. Sitaraman. Minimal test collections for retrieval evaluation. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 268–275, New York, NY, USA, 2006. ACM.

[9] B. Carterette, V. Pavlu, E. Kanoulas, J. A. Aslam, and J. Allan. Evaluation over thousands of queries. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 651–658, New York, NY, USA, 2008. ACM.

[10] P. Donmez, K. Svore, and C. J. Burges. On the optimality of lambdarank. Technical report, Microsoft Research, 2008.

[11] B. He, C. Macdonald, and I. Ounis. Retrieval sensitivity under training using different measures. In *SIGIR '08: Proceedings of the 31th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 67–74. ACM, 2008.

[12] K. Järvelin and J. Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 41–48, New York, NY, USA, 2000. ACM.

[13] T.-Y. Liu, T. Qin, J. Xu, X. Wenying, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval.

[14] M. Taylor, J. Guiver, S. Robertson, and T. Minka. Softrank: optimizing non-smooth rank metrics. In *WSDM '08: Proceedings of the international conference on Web search and web data mining*, pages 77–86, New York, NY, USA, 2008. ACM.

[15] E. Yilmaz and J. A. Aslam. Estimating average precision with incomplete and imperfect judgments. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 102–111, New York, NY, USA, 2006. ACM.

[16] E. Yilmaz and S. Robertson. Deep versus shallow judgments in learning to rank. In *SIGIR '09: Proceedings of the 32th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, 2009. ACM. To Appear.

[17] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in informat ion retrieval*, New York, NY, USA, 2007. ACM Press.