Inf Retrieval (2010) 13:271–290
DOI 10.1007/s10791-009-9116-x

On the choice of effectiveness measures for learning to rank

Emine Yilmaz · Stephen Robertson

Received: 24 April 2009 / Accepted: 28 August 2009 / Published online: 16 September 2009

Abstract Most current machine learning methods for building search engines are based on the assumption that there is a target evaluation metric that evaluates the quality of the search engine with respect to an end user and the engine should be trained to optimize for that metric. Treating the target evaluation metric as a given, many different approaches (e.g. LambdaRank, SoftRank, RankingSVM, etc.) have been proposed to develop methods for optimizing for retrieval metrics. Target metrics used in optimization act as bottlenecks that summarize the training data and it is known that some evaluation metrics are more informative than others. In this paper, we consider the effect of the target evaluation metric on learning to rank. In particular, we question the current assumption that retrieval systems should be designed to directly optimize for a metric that is assumed to evaluate user satisfaction. We show that even if user satisfaction can be measured by a metric X, optimizing the engine on a training set for a more informative metric Y may result in a better test performance according to X (as compared to optimizing the engine directly for X on the training set). We analyze the situations as to when there is a significant difference in the two cases in terms of the amount of available training data and the number of dimensions of the feature space.

Keywords Evaluation · Evaluation metrics · Learning to rank · Training · Empirical risk minimization

1 Introduction

We consider the problem of optimizing the performance of a retrieval system with respect to a particular evaluation measure. Given an evaluation measure *M* that is assumed to

E. Yilmaz (✉) · S. Robertson
Microsoft Research Cambridge, 7 JJ Thomson Ave, Cambridge CB3 0FB, UK
e-mail: eminey@microsoft.com

S. Robertson
e-mail: ser@microsoft.com

evaluate the quality of the search engine, search engines are build so that they optimize their performance with respect to this measure.

In the field of information retrieval, many different evaluation metrics have been proposed and used. Most of these metrics have different purposes and are known to evaluate different aspects of retrieval effectiveness. These evaluation metrics are commonly considered in two categories.

Some metrics mainly focus on the quality of the top end of the ranking, assuming that the users mainly care about the top end of the ranking (e.g. top 10 documents in the case of web search) and ignore the rest. These metrics are often referred to as user-oriented metrics (e.g. precision at rank 10, ndcg at rank 10).

Measures that evaluate the overall quality of an entire ranked list are often described as system-oriented. Actually, such measures may be associated with a user model, perhaps representing a population of users rather than a single user (Robertson 2007). However, this may not be their main motivation, but they may be intended primarily to represent quality in a reliable, informative and discriminatory way. The criteria of reliability, informativeness, and discriminative power may conflict with a strong user motivation.

The main assumption in building search engines is that if a metric M evaluates the utility of the search engine to an end user, then the search engine should be trained to optimize for that particular metric (Burges et al. 2006; Yue et al. 2007; Taylor et al. 2008). This approach is based on the principle of *empirical risk minimization* which dictates that in order to optimize the algorithm according to a criterion M, one should train the algorithm to minimize (or maximize) M using the training data.

Recently, Aslam et al. (2005) proposed a framework to analyze measures of retrieval performance in terms of how informative they are and showed that some evaluation metrics (e.g. metrics that evaluate the overall quality such as average precision) are more informative than others (e.g. metrics that mostly evaluate the quality of the top end of the ranking). Similarly, Webber et al. (2008) showed that some 'complex' evaluation metrics (metrics that evaluate the overall quality of a ranking) are better than some 'simpler' metrics (metrics that evaluate the quality of the top end) in terms of predicting the value of 'simple' metrics on other topics.

In this paper, we consider the problem of optimization of search engine performance and question the current approach based on empirical risk minimization. Based on the results on informativeness and predictive power of evaluation metrics, we hypothesize that some more informative (or predictive) metrics may be better than others for learning to rank purposes. In particular, we show that even if the user satisfaction can be measured by a metric X, optimizing the search engine for a more informative metric Y in the training set may result in better performance in terms of X in the test set, as opposed to directly optimizing the engine for X in the training set.

In the sections that follow, we first describe some of the commonly used evaluation metrics and give an analysis of why optimizing for a more informative metric may be better for learning purposes. Then, we extend the current learning to rank methods LamdaRank (Burges et al. 2006) and SoftRank (Taylor et al. 2008) to optimize for different IR metrics and validate our hypothesis. We further analyze the effect of the amount of training data and the number of dimensions of the feature space on the validity of this hypothesis and conclude that since obtaining training data is an expensive procedure and often only a limited amount of data is available, one should better optimize for a more informative metric. Finally, we explain this behavior by showing how the local optimum of a more informative metric compares with the local optimum of a less informative metric, both in the test set and the training set.

2 Evaluation metrics

Out of the many evaluation metrics proposed in IR, precision at cutoff *k* (PC(k)), average precision (AP), and NDCG are three of the most commonly used metrics.

Assuming that the user is only interested in top *k* documents, PC(k) is defined as the proportion of relevant documents up to rank *k*. A typical value for *k* is 10, based on web search engines.

AP can be defined as the average of precision values at each relevant document. If a system did not retrieve all relevant documents, these documents are assumed to be retrieved at infinity (hence, the precision value associated with each of these documents is zero).

PC(k) and AP are based on the assumption that relevance judgments are binary (i.e., a document can be either relevant or nonrelevant), an assumption clearly not true. Instead, Järvelin and Kekäläinen (2000) proposed NDCG, a measure that can encorporate graded relevance judgments. NDCG is defined as the sum of discounted cumulative gains divided by maximum such value. Assuming the user only cares about the top end of the ranking, NDCG at rank *k* (NDCG(k)) is also often used.

3 Training and informativeness of metrics

When search engines are optimized for an objective evaluation metric based on empirical risk minimization, the evaluation metric used during training acts as a bottleneck that summarizes the available training data. At each training epoch, given the relevance of the documents in the training set and the ranked list of documents retrieved by the search engine for that epoch, the only information the learning algorithm has access to is the value of the evaluation metric, the algorithm does not have access to the relevance of individual documents. So, the ranking algorithm be changed on the basis of the change in the value of the metric.[1]

Given this fact about evaluation metrics and optimization, which metric is then a good metric to optimize for? According to the principle of empirical risk minimization, if the metric we ultimately care about (the test metric) is “X”, then we should also be optimizing for X using the training data. Intuitively, one would then think that if we care about precision-at-cutoff *k*, then this is what we should be optimizing for.

Our hypothesis is that this argument is not valid and that since evaluation metrics act as bottlenecks that summarize the training data, it is actually better to choose a more *informative* metric to optimize on the training data. This may be better in the sense of giving better test results (test metric + test data) than optimizing the test metric on the training data. In the following sections, we will first describe why some metrics may be more informative than others. We use the maximum entropy framework (Aslam et al. 2005) to compare different evaluation metrics, AP and NDCG, in terms of their informativeness. We then validate our hypothesis using two different learning to rank algorithms.

3.1 Why are some metrics more informative?

One metric can be more informative than another due to two main reasons: (1) some metrics ignore some changes in the quality of the ranking, and (2) some metrics give to much weight to some parts of the ranking due to the properties of the discount function used.

[1] Note that this is specific to the learning algorithms that directly optimize for an objective metric, ignoring the relevance of individual documents as this is the common practice for many learning to rank algorithms.

Most metrics in information retrieval are based on ranks and relevances of documents and their values only change if two documents with different relevance values are flipped. As pointed out by Robertson and Zaragoza (2007), metrics may ignore some flips, but if they respond at all, all reasonable metrics behave in the same way: positive for a good flip (relevant moved above non-relevant), and vice versa. If used for optimization, they therefore tend to guide the ranker in a similar direction.

In general, multi-graded measures are more informative than binary measures since they respond to any flips between documents of different relevance grade. Furthermore, given two binary (or multi-graded) measures, one measure may still be more informative than another for three main reasons: (1) some measures respond only to flips in some specific part of the ranking, usually the top end; (2) even if two measures depend on the same part of the ranking, one may be insensitive to some flips within this part, and (3) even if two measures are sensitive to the same flips, one may be more informative than another.

3.1.1 Loss of information due to ignoring flips

Suppose we know that the user only cares about the top 10 documents and never looks at the documents beyond this rank, and that the metric that evaluates the satisfaction of the user is precision at cutoff 10. Now consider optimizing the search engine for this metric. Since the metric only depends on the top 10 documents, during training, the learning algorithm will try to collect the best set top 10 documents relative to the remaining documents. However, at each training step, the desired change will be based only on the selection of the top 10 documents as compared to the remainder. The relative ranking of documents beyond rank 10 will be ignored.

Furthermore, the ranking of documents 1–10 will also be ignored by PC(10). Note, on the other hand, that NDCG(10), even though it uses exactly the same documents in the training data as PC(10), does take account of flips within ranks 1–10 (even if the relevance judgments were binary).

If we could assume that we had enough training data, and further that all possible situations that might arise in the real world (or indeed in any held-out test data) were adequately represented in the training data, then ignoring these flips would not be a problem. However, it is quite likely that the test set will contain examples of documents which are similar to those ignored in training, but which affect important parts of the ranking in the test.

As an example, consider a collection with $N = 20$ documents, a query with $R = 5$ relevant documents, and two search engines. Assume that the first retrieves the five relevant documents at ranks 1, 2, 11, 12, 13 and the second engine retrieves them at ranks 8, 9, 18, 19, 20. According to PC(10), the two are equally effective, though by many other measures (as well as common sense), the first is better. One might argue this is not a problem; if the user examines all documents in the top 10, then it does not really matter where these documents are. However, in the test set, there might be other documents that have similar properties to the top 7 nonrelevant documents retrieved by the second engine, hence the second engine would retrieve many more nonrelevant documents before retrieving the relevant documents. Thus, it would have worse performance in terms of PC(10) in the test set.

3.1.2 Loss of information due to discount function

Apart from the loss of information due to insensitivity to some flips, some information can also be lost due to properties of evaluation metrics. In general, some evaluation metrics are more informative than others even if they depend on the same amount of information as they give different weights to some flips (they may weights flips in some part of the ranking much more than the flips in another part of the ranking). In the following section, we will show how such a situation might arise by comparing NDCG and AP in terms of their informativeness.

3.2 Informativeness of metrics

Informativeness of an evaluation metric is related to how well it summarizes the training data, i.e., the relevance of documents retrieved by the learning algorithm (or search engine) for a given query. If a metric is highly informative, then given the value of the metric, the learning algorithm should be able to infer the relevance of individual documents retrieved.

Based on this intuition, Aslam et al. (2005) proposed a framework that can be used to analyze retrieval measures in terms of their informativeness. The maximum entropy framework is based on the assumption that the quality of a list of documents retrieved in response to a given query is strictly a function of the relevance of the documents retrieved within that list. The question that naturally arises is how well does a measure capture the distribution of relevance over the output list? In other words, given the value of a measure, for a particular system on a particular query, how accurately can one predict the relevance of documents retrieved by the system? In order to compute how well a measure predicts the relevance of the returned documents, the maximum entropy framework is based on forming a distribution over all possible sequences of relevance and finding the maximum entropy distribution over these lists given the value of the measure. Using the maximum entropy distribution, it is guaranteed that no extra information is used other than the value of the given measure. Thus, using only the information given by the metric, a distribution over the sequence of relevant/nonrelevant documents of length N is formed. Assuming that the distribution over lists is a product distribution, i.e. the probability of a particular sequence of relevance is the product of the probability of different relevance grades at each rank, the probability of relevance of a document at a particular rank (probability-at-rank distribution) given the value of a metric can then be inferred.

To quantify how informative a metric is, two main criteria are used (Aslam et al. 2005): (1) how do the precision-recall curves obtained from the inferred probability-at-rank distribution compare with the actual precision-recall curves of the system, and (2) how well can the probability-at-rank distribution predict other metrics? If a measure is highly informative, then one should be able to accurately infer the relevance of individual documents, and hence given the value of the metric, these inferences should also be accurate.

We would like to emphasize that informativeness of an evaluation metric is mainly related how well it summarizes the relevance of documents in the ranked list. Hence, these two criteria mainly aim at evaluating how the inferred probabilities of relevance of each document compare with their actual relevance. It is easy to show that given the precision-recall curve of a system as well as the total number of relevant documents in the collection (R), one can infer the relevance of all the documents retrieved by the system. Therefore, precision recall curves perfectly summarize the relevance of documents retrieved. Hence, if a measure can accurately predict the probability of relevance of documents retrieved by a system, then the inferred precision-recall curve of the system given the value of the metric should be very similar to the actual precision-recall curve. Similarly, if a metric can predict the probability of relevance of documents accurately, then given the value of the evaluation metric one can also predict other evaluation metrics accurately.

In their experiments, Aslam et al. mainly focused on comparing the informativeness of evaluation metrics for evaluation purposes. They compared binary evaluation metrics such as average precision, precision at cutoff 10 and R-precision (precision at cutoff R, where R is the total number of relevant documents) and showed that average precision is more informative than the other two metrics. In this paper, we extend the maximum entropy framework to multi-graded evaluation metrics and use the maximum entropy framework for comparing the informativeness of evaluation metrics used during optimization. Based on the results from this framework, we show that the informativeness of the objective evaluation metric used for optimization in learning to rank is highly important and that if you have a limited amount of available training data, it is better to optimize for a more informative evaluation metric.

3.3 Which evaluation metric should we optimize for?

Suppose the (test) performance of our search engine will be evaluated on 2 different objective metrics, PC(10) or NDCG(10). Which metric should we then optimize for?

Based on the results from Aslam et al. (2005), average precision is more informative than precision at cutoff 10 as it gives a lot of information about the relevance of documents retrieved by the search engine, i.e., given the value of the metric one can more accurately predict the relevance of individual documents retrieved, as compared to precision at cutoff 10. As explained before, this is due to the fact that PC(10) is insensitive to some changes in the ranking. Similarly, NDCG is also more informative than PC(10) as it responds to many changes in the ranked list. Therefore, we hypothesize that if the test metric is PC(10), it is better to optimize the search engine for average precision or NDCG as compared to PC(10).

Suppose our test metric is again PC(10), is it then better to optimize for NDCG or average precision? Note that NDCG is more sensitive to changes in the ranking as it is a multi-graded metric whereas average precision is binary. Furthermore, NDCG is more correlated with precision at 10 as compared to average precision.

Figure 1 shows the correlations between average precision and PC(10), as opposed to correlations between NDCG and PC(10) for TRECs 9 and 10. TRECs 9 and 10 contain judgments with 3 relevance grades, nonrelevant (0), relevant (1), and highly relevant (2). Each dot in the plots correspond to the average performance of a system over 50 queries. The plots contain the root mean squared error (RMS) (how correlated are the values?) and the Kendall's τ (how correlated are the rankings of systems?) statistics. It can be seen that NDCG metric is more correlated with PC(10) than the average precision metric.

Since (1) NDCG responds to more flips than average precision, and (2) NDCG is more correlated with PC(10), one might expect optimizing for NDCG to result in better test set performance in terms of PC(10), as opposed to optimizing for average precision. However, according to the maximum entropy framework, we will now show that average precision is a more informative metric than NDCG for summarizing the binary relevance of individual documents and that given the value of average precision, one can more accurately predict the relevance of documents retrieved by the system.

In the following section, we will use the maximum entropy framework to compare average precision and NDCG to quantify how informative each metric is regarding the binary relevance of documents in a ranked list, and we will show that average precision is actually more informative than NDCG. Therefore, we conclude that when limited amount of training data is available, if the goal is to optimize for PC(10), optimizing for average precision should be preferred.

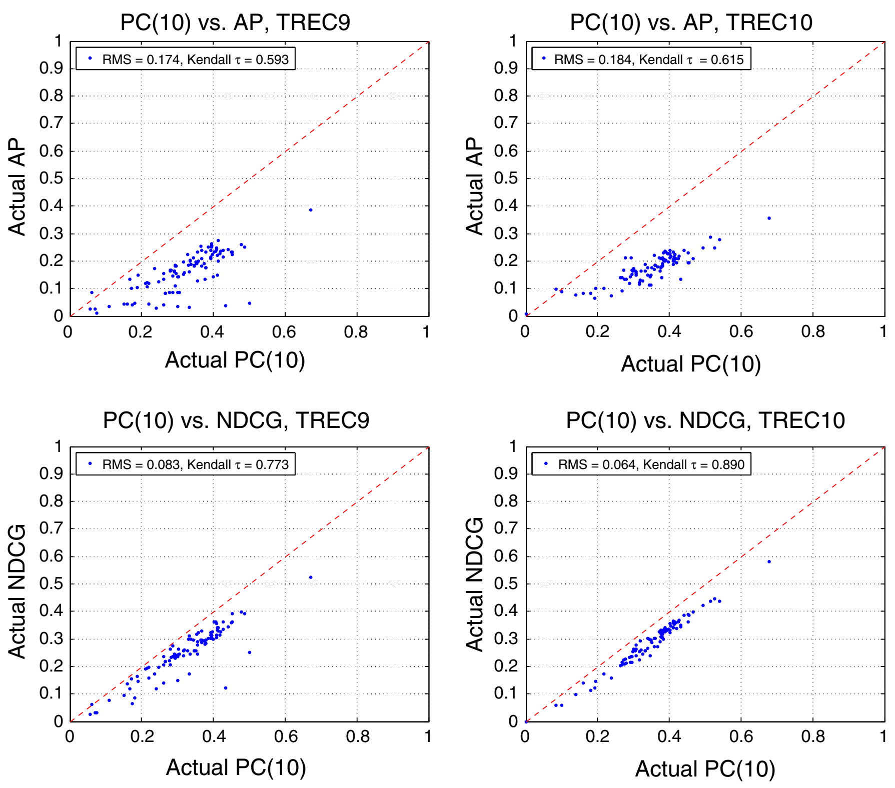

Fig. 1 Correlations of precision at 10 with (*top*) average precision and (*bottom*) NDCG for TRECs 9 and 10

3.3.1 Maximum entropy method for comparing AP and NDCG

Since our goal is to infer the relevance of individual documents given the value of the metric, according to the maximum entropy framework, we find the maximum entropy probability of relevance distribution subject to the value of the evaluation metric as a constraint. This requires us to define the expectations of evaluation measures given the probability of relevance at rank distribution.

Let N be the total number of documents retrieved by a system, R be the total number of relevant documents in the query and $P(rel_n = 1)$ is the probability that the document at rank n is relevant, then the expected value of average precision can be written as (Aslam et al. 2005):

$$E[AP] = \frac{1}{R}\sum_{n=1}^{N}\left(\frac{P(rel_n = 1)}{n}\left(1 + \sum_{j=1}^{n-1} P(rel_j = 1)\right)\right)$$

The maximum entropy method was originally proposed to compare binary evaluation metrics. We further extended the maximum entropy method to encorporate multi-graded evaluation metrics such as NDCG. In the above setup, let $\xi \in 0, 1, \ldots c$ be a relevance grade and let $Pr(rel_n = \xi)$ be the probability that the relevance of document at rank n is ξ. Let $g(\xi)$ be the gain value associated with relevance grade ξ in the NDCG computation. Then, the expected value of NDCG can be defined as:

Maximize: $H(\mathbf{P}) = \sum_{n=1}^{N} H(P_n)$

Subject to:

1. $\frac{1}{R}\sum_{n=1}^{N}\left(\frac{P(rel_n = 1)}{n}\left(1 + \sum_{j=1}^{n-1} P(rel_j = 1)\right)\right) = ap$
2. $\sum_{n=1}^{N} P(rel_n = 1) = R_{ret}$
3. $\sum_{n=1}^{N} P(rel_n = 1) = 1$

Maximize: $H(\mathbf{P}) = \sum_{n=1}^{N} H(P_n)$

Subject to:

1. $\sum_{n=1}^{N}\sum_{\xi=0}^{c} \frac{g(\xi)\cdot P(rel_n = \xi)}{lg(n+1)} / (\text{optDCG}) = ndcg$
2. $\sum_{n=1}^{N} P(rel_n = \xi) = R_\xi \qquad \forall \xi : 1 \le \xi \le c$
3. $\sum_{\xi=0}^{c} P(rel_n = \xi) = 1 \qquad \forall n : 1 \le n \le N$

Fig. 2 Maximum entropy setup for AP and NDCG, respectively

$$E[NDCG] = \sum_{n=1}^{N}\sum_{\xi=0}^{c}\frac{g(\xi)\cdot P(rel_n = \xi)}{lg(n+1)}\Big/(\text{optDCG})$$

The maximum entropy formulations for average precision and NDCG are shown in Fig. 2. The second constraint in the formulations guarantees that there are a given number of relevant documents in total (in the case of average precision), or there are a given number of documents from each relevance grade (in the case of NDCG), and the third constraint guarantees that we have a distribution.

We compare the informativeness of AP and NDCG using data from TRECs 9 and 10. Using the setup described above, we first infer the probability of relevance of each document given the value of each metric and then calculate the maximum entropy precision-recall curves. As NDCG is a multi-graded metric, the maximum entropy method gives us the probability of a document being in each relevance grade. Since our goal is to compare AP and NDCG in terms of how well they summarize the probability of relevance of each document, we convert the inferences obtained from NDCG to binary by summing these probabilities for each grade $c > 0$.

Once the inferred probability of relevance values are obtained given the value of each evaluation metric, we also infer the precision-recall curve of a system given the value of each evaluation metric and compare the inferences with the actual precision-recall curve of a system.

The mean RMS and mean absolute error (MAE) between the inferred and the actual precision-recall curves, calculated at the points where recall changes, is shown in Table 1. It can be seen that the inferences obtained from AP are always better than the inferences obtained from NDCG. Hence, one can conclude that average precision is actually more informative than NDCG in terms of summarizing the binary relevance of each document.

In order to further compare the informativeness of the two metrics, we also compare how well these two metrics can predict other evaluation metrics. Given the probability of

Table 1 RMS and MAE between the actual precision-recall curves and inferred precision-recall curves given the value of NDCG and AP

	TREC9		TREC10	
	RMS	MAE	RMS	MAE
NDCG	0.2034	0.1686	0.1751	0.1214
AP	0.1666	0.1134	0.1620	0.1086

relevance distributions obtained given each metric, we infer the value of PC(10) and compare the inferred PC(10) values with the actual PC(10) values. Figure 3 shows how the inferred values of precision at 10 given each metric compare with the actual value of precision at 10. It can be seen that average precision actually contains much more information about PC(10) as compared to NDCG (in contrast to the original correlations as shown in Fig. 1) since average precision can accurately predict the probability of relevance of documents. Note that for TREC10, the inferences from NDCG look better in terms of RMS error when compared to AP (even though the correlations seem higher visually); however, this is mainly caused by the two outlier systems.

Based on these results, we claim that in order to obtain a better test set value with respect to a binary evaluation measure (such as PC(10)), one should better optimize for average precision as opposed to NDCG since average precision is more informative regarding the binary relevance of.

Although optimizing for a different metric other than the test metric may seem counter intuitive, similar conclusions were reached in machine learning when optimizing classifiers for error rate. Error rate has similar problems to precision at cutoff *k* in the sense that it does not much discriminate between different ranked lists. Ling et al. (2003) have shown that even if what one cares about is error rate, training to optimize for the area under ROC curve produces better classifiers in terms of their error rate than classifiers trained to directly minimize error rate.

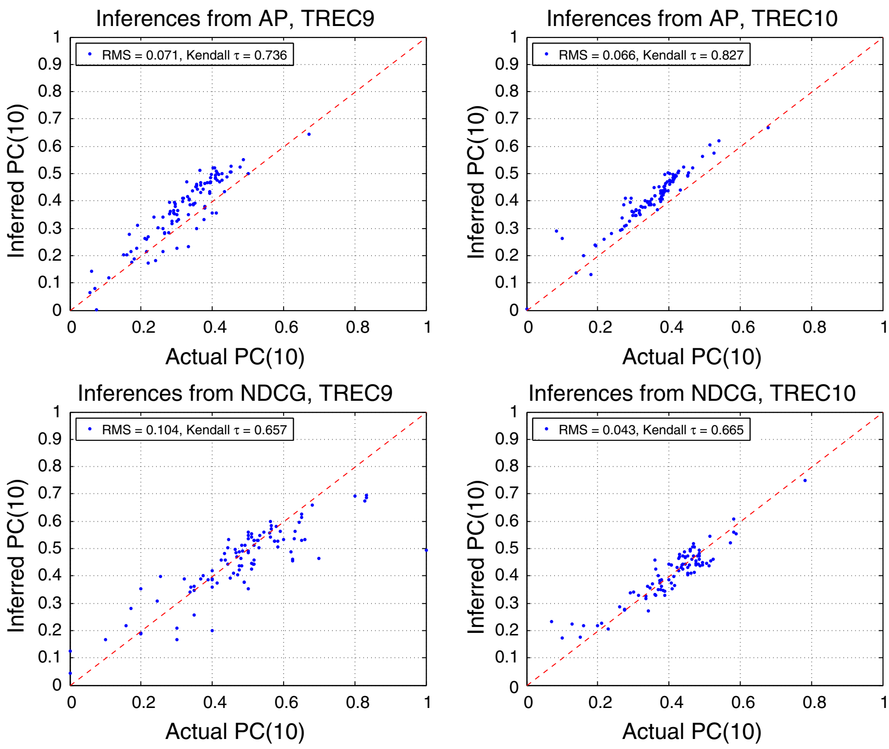

Fig. 3 Actual PC10 values compared to inferred PC10 values given the value of (*up*) average precision, and (*bottom*) NDCG, for TRECs 9 and 10

Recently, Le and Smola (2007) showed that optimizing for Winner Takes it All (WTA) metric results in worse test performance in terms of WTA, as opposed to optimizing for NDCG at rank k. However, the authors do not provide a detailed analysis and explanation of the phenomenon. Their main explanation is based on the argument that WTA is a less structured metric. Furthermore, they still use NDCG at rank k for optimization, as opposed to computing NDCG of the entire ranking.

The common assumption in the learning to rank community is that regardless of the informativeness of the metrics and the amount of available data training data, the objective metric used in optimization should be the target test metric. For example, Xu and Li (2007) optimize for NDCG at rank 5, Joachims (2005) mentions that the search engines should be optimized for the target test metric and uses target metrics such as the F-score, R-precision (referred to as precision-recall break-even point), etc. for optimization. Similarly, Burges et al. (2006) mention that in an ideal situation, the optimization cost should be the target cost.

4 Optimizing different metrics of retrieval effectiveness

In order to compare the effect of evaluation metrics used in optimization and test our hypothesis, we use two algorithms that are mainly designed to optimize directly for evaluation metrics. The two main algorithms that we use are SoftRank (Taylor et al. 2008) and LambdaRank (Burges et al. 2006), which are shown to find local optima for the target evaluation metrics (Liu et al. 2008; Donmez et al. 2008). Given that both algorithms are shown to find the local optimum with respect to an objective metric, our results are likely to apply for any other learning to rank algorithm that can find the local optimum for the objective metrics.

Both LambdaRank and SoftRank were originally designed to optimize for NDCG. In the following section, we give a basic description of both methods and describe how they can be extended to optimize for average precision and precision at cutoff k. In all our experiments, we use NDCG with the most commonly used $\log_2(r+1)$ discount, where r is the rank and the exponential gain $2^{rel(i)} - 1$, where $rel(i)$ is the relevance of document i (Taylor et al. 2008; Burges et al. 2006).

4.1 SoftRank

Information retrieval metrics are not smooth as they mainly depend on the ranks of documents, which are discontinuous. The main idea used in SoftRank 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. (2008) 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 define SoftPC and SoftAP, the *expected* precision-at-cutoff and average precision with respect to these distributions and compute the gradients of these metrics.

4.1.1 SoftRank for optimizing PC(k) (SoftPC)

Given the rank distribution as defined above, expected value of precision at cutoff k with respect to this distribution can be written as:

$$PC(k) = \frac{\sum_{j=1}^{N} rel(j) \cdot P(r_j \leq k)}{k}$$

where N is the total number of documents in the collection, *rel*(*j*) is the binary relevance of document j, and $P(r_j \leq k)$ is the probability that the document j will be among the top k documents retrieved by the search engine, which can be computed as $P(r_j \leq k) = \sum_{r=1}^{k} p_j(r)$.

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

$$\frac{\partial PC(k)}{\partial \bar{s}_m} = \frac{\partial PC(k)}{\partial p_m(r)} \cdot \frac{\partial p_m(r)}{\partial \bar{s}_m}$$

where

$$\frac{\partial PC(k)}{\partial p_m(r)} = \begin{cases} \frac{rel(m)}{k} & \text{if } r \leq k \\ 0 & \text{o.w.} \end{cases}$$

and $\frac{\partial p_m(r)}{\partial \bar{s}_m}$ can be computed as described by Taylor et al. (2008).

4.1.2 SoftRank for optimizing AP (SoftAP)

In order to define soft average precision, consider the random experiment corresponding to average precision, as defined by Yilmaz and Aslam (2008):

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:

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

where R is the number of relevant documents in the query.

Using the same approach as before, he derivative of AP with respect to the score of document $\bar{s}_m$ can be written as

$$\frac{\partial AP}{\partial \bar{s}_m} = \sum_{i=1}^{N}\sum_{j=1}^{N} \frac{\partial AP}{\partial \pi_{ij}} \cdot \frac{\partial \pi_{ij}}{\partial \bar{s}_m} + \frac{\partial AP}{\partial \pi_{ji}} \cdot \frac{\partial \pi_{ji}}{\partial \bar{s}_m}$$

$$\frac{\partial AP}{\partial \pi_{ij}} = \frac{rel(j)}{R}\left[\frac{rel(i)}{\sum_{i=1, i\neq j}^{N} \pi_{ij} + 1} - \frac{rel(j) + \sum_{i=1, i\neq j}^{N} \pi_{ij} rel(i)}{\left(\sum_{i=1, i\neq j}^{N} \pi_{ij} + 1\right)^2}\right]$$

and $\frac{\partial \pi_{ij}}{\partial \bar{s}_m}$ can again be computed as described by Taylor et al. (2008).

4.2 LambdaRank

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 points needed.

Given a pair of documents, the virtual gradients (λ functions) used in LambdaRank are obtained by scaling the RankNet (Burges et al. 2005) cost with the amount of change in the value of the metric obtained by swapping the two documents (Burges et al. 2006).

Following the same setup, in order to optimize for the precision or 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 gradients in LambdaRank is shown to find the local optima for the target metrics (Donmez et al. 2008).

4.2.1 *LambdaRank for optimizing PC(k)*

The discount function corresponding to precision at cutoff k can be written as:

$$D(r) = \begin{cases} \frac{1}{k} & \text{if } r \leq k \\ 0 & \text{o.w.} \end{cases}$$

where $PC(k) = \sum_{j=1}^{N} rel(j) \cdot D(r_j)$, where r_j is the rank of document j.

Then, when document i and j are swapped, the change in precision value is equal to $(rel(i) - rel(j)) \cdot (D(r_i) - D(r_j))$. When weighted by the RankNet cost, the λ function for precision is:

$$\lambda = \frac{1}{1 + exp(s_i - s_j)} \cdot (rel(i) - rel(j)) \cdot (D(r_i) - D(r_j))$$

4.2.2 *LambdaRank for optimizing AP*

Average precision is computed as average of the precisions at relevant documents and the precision at each relevant document depends on the relevance of all documents above that document. Therefore, swapping two documents may affect the precision at all documents that are ranked in between the two documents. Hence, the difference in the average precision value when two documents are swapped is not as obvious as the change in the value of NDCG or precision at cutoff k.

Consider two documents j and i at ranks r_1 and r_2. When the two documents are swapped so that document j is now at rank r_2 and document i at rank r_1, assuming $r_1 < r_2$, the change in the average precision value can be written as:

$$\Delta AP = \frac{1}{R} \cdot \left[\Delta P(r_1) + \sum_{r \in (r_1, r_2)} \Delta P(r) + \Delta P(r_2) \right]$$

where $\Delta P(i)$ corresponds to the change of precision at rank i when the two documents are swapped. Then the change in precision value at rank r_1 and r_2 can be computed as

$$\begin{aligned} \Delta P(r_1) = & \frac{rel(i)}{r_1} \left(rel(i) + \sum_{r=1}^{r_1 - 1} rel(r) \right) \\ & - \frac{rel(j)}{r_1} \left(rel(j) + \sum_{r=1}^{r_1 - 1} rel(r) \right) \\ \Delta P(r_2) = & \frac{(rel(j) - rel(i)) \cdot \left(\sum_{r=1, r \neq r_1}^{r_2 - 1} rel(r) + 1 \right)}{r_2} \end{aligned}$$

For all ranks r such that $r_1 < r < r_2$, the change in precision value is

$$\Delta P(r) = \frac{rel(r)}{r}(rel(i) - rel(j))$$

Once these values are computed, the lambda values can be computed as

$$\lambda = \frac{1}{1 + exp(s_i - s_j)} \cdot \Delta AP$$

5 Experimental results

Given that we have extended the current learning methods to optimize for different evaluation metrics, we now use these algorithms to validate our hypothesis. In this section, we first show that when the number of features is large, training for a more informative metric produces significantly better results than training for a less informative metric. We further investigate the effect of the number of features and the amount of available training data on the validity of the hypothesis and finally analyze how the local optimum of a more informative metric compares with the local optimum of a less informative one, both in the test set and the training set.

In order to test our hypothesis, we run each algorithm with data obtained from a commercial web search engine. The web data contains 382 features and is split into train, validation and test sets. We use training data of various sizes since the effect of the informativeness of a metric on a learning to rank algorithm depends on the amount of available training data. We use validation and test sets with 1K, and 2K queries, respectively. The documents in the datasets are assigned relevance judgments with 5 grades, between 0 (nonrelevant) to 4 (most relevant). In order to compute the binary measures, we convert these judgments to binary by assuming that documents with relevance > 0 are all relevant and the rest are nonrelevant.[2]

In our experiments, similar to the setup used in LambdaRank and SoftRank, we use a two layer net with ten hidden nodes. For various learning rates, we run each algorithm for 300 epochs and pick the best epoch according to the validation set.

We consider PC(10) and NDCG(10) as the test metrics that evaluate user satisfaction. We also report the AP value as the test metric for comparison purposes. Tables 2 and 3 show the result of training, testing and validating using different measures on a dataset with 2K training queries using LambdaRank and SoftRank as the learning algorithms, respectively. By convention, we report the results in the scale of 0–100 for each metric.

The rows of the tables correspond to training for a different metric, average precision, precision at cutoff 10, NDCG at 10, and NDCG of the entire list. The columns correspond to the test metric. When training (optimizing) the search engine for a measure *X* and using a test measure *Y*, we use *Y* as the validation metric to pick the best epoch using the validation set.[3]

The $^\bullet$ mark indicates statistically significant differences (with 95% confidence) compared to training, validating and testing using the test metric itself and $^\circ$ corresponds to significant difference when training for AP and NDCG are compared, given a test metric. In order to avoid reaching wrong conclusions due to using a particular significance test, we

[2] We tried varying the conversion by thresholding at different grades, but the conclusions did not change.

[3] We tried using the training metric as the validation metric as well, but did not observe significant differences in the results.

Table 2 LambdaRank results on 2K web data

Testmetric	AP	PC(10)	NDCG(10)
TrainAP	**62.91**$^\circ$	**54.96**$^\bullet$	**63.03**$^\bullet$
Train PC(10)	62.28	54.44	62.24
Train NDCG	62.82	54.92$^\bullet$	62.98$^\bullet$
Train NDCG(10)	62.30	54.72$^\bullet$	62.41

Table 3 SoftRank results on 2K Web data

Testmetric	AP	PC(10)	NDCG(10)
Train AP	**61.95**$^\circ$	**54.21**$^{\bullet\circ}$	61.29
Train PC(10)	59.99	52.73	61.81
Train NDCG	61.30	53.27$^\bullet$	**62.90**$^{\bullet\circ}$
Train NDCG(10)	60.82	52.77	62.37

mark a difference as statistically significant if it is significant with 95% confidence according to both Wilcoxon sign-rank, sign test and *t*-test (Sanderson et al. 2005).

It can be seen from the tables that even if the search engine will be ultimately tested on PC(10) or NDCG(10), it is better to train the search engine for a more informative metric such as AP or NDCG. This result is consistent for both learning algorithms. Note that the improvements obtained by optimizing for a more informative metric are similar in terms of value to the improvement achieved by LambdaRank over RankNet (Burges et al. 2006) or LambdaRank over SoftRank (Taylor et al. 2008) on a similar dataset obtained from a commercial search engine. Since LambdaRank is known to be better than RankNet (or SoftRank), these results suggest that the informativeness of the objective metric may be as important as the quality of the learning algorithm. k over SoftRank on a similar dataset obtained from a commercial search engine as it is difficult to achieve these improvements.

When the two measures that depend on the entire ranking are compared (AP and NDCG), it can be seen that when SoftRank method is using for optimization, training for average precision produces significantly better results for average precision and PC(10), compared to training for NDCG. When LambdaRank is used, average precision still produces better values for the two binary evaluation metrics (AP and PC(10)). However, the differences are not significant when the test metric is PC(10). This result is consistent with the conclusions obtained through the maximum entropy framework in the previous section that suggests that average precision is actually more informative than NDCG in terms of summarizing the binary relevance of documents. These results also suggest that properties of average precision should be further explored and a multi-graded version of the measure be designed.

When NDCG(10) is compared with precision at cutoff 10, it can be seen that NDCG(10) consistently outperforms PC(10), as expected.

5.1 Number of features, amount of training data and informativeness

In this section, our goal is to answer two main questions: (1) if the number of features is small (hence, the algorithm does not need as much information for learning), and (2) if much more training data were available, does optimizing for a more informative metric still result in a significant improvement over optimizing for a less informative one?

Taylor et al. (2006), recently showed that when there are very few features (e.g. 2), a much smaller amount of training data is enough for learning and increasing the amount of training data does not result in any significant improvement. As the number of features increase, the learning algorithm needs more training data to perform well.

Similarly, when the number of features is large, using a more informative metric in training should result in a bigger improvement in test set performance over a less informative metric. When the number of features is small, since the algorithm does not need as much information, training using a less/more informative metric will probably not significantly affect the test set performance.

Note that this hypothesis is also related to the amount of available training data versus the number of features. If the number of features is large, then one would need more training data to get a similar performance when using a non-informative metric versus when using an informative metric. Furthermore, even if the number of features is large, if infinite amount of training data were available, then the informativeness of the objective metric may not make significant differences.

In order to check the tradeoff between the number of dimensions, informativeness of a metric and the amount of training data, we use the data from the 2K training set and vary the number of features (382 vs. 9 vs. 2, as in Taylor et al. (2006)) and the amount of available training data (by sampling queries from this training set) and optimize the search engine for different target metrics. Since LambdaRank and SoftRank behave in a similar way, we mainly focus on LambdaRank as the learning algorithm.

For comparison purposes, we focus on the effect of training for average precision assuming the test metric is PC(10) compared to training for PC(10) and the effect of training for NDCG assuming the test metric is NDCG(10) compared to training for NDCG(10).

Figure 4 shows the result of this experiment. The left plot in the figure shows the test set PC(10) value when training for AP versus PC(10) when the number of features varies. Similarly, the right plot shows the test set NDCG(10) value when training for NDCG versus NDCG(10) when the number of features varies. The *x* axis in both plots show the amount of training data used in optimization as a percentage.

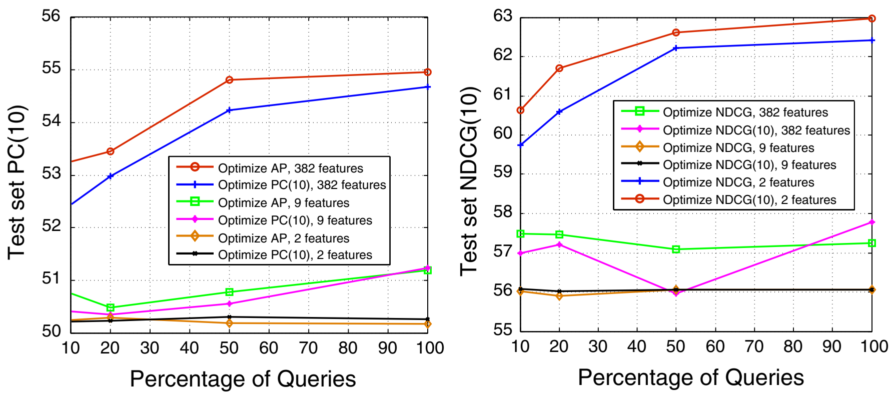

Fig. 4 (*Left*) Test set PC(10) values when optimizing for AP versus PC(10) (*Right*) Test set NDCG(10) values when optimizing for NDCG versus NDCG(10), varying the number of queries used in training and the number of features

In both plots, when the number of features is small (i.e. 2), for all amounts of training data, training on AP (or NDCG) gives similar test performance to training on PC(10) (or NDCG(10)), and there is no statistically significant differences in the results.

Figure 4 shows that when we use 9 features, except when 100% training data is used, training on AP (or NDCG) results in significantly better test set PC(10) (or NDCG(10)) values when compared to training on PC(10) (or NDCG(10)). When 100% training data is used, there is no significant difference in the two cases.

When all the features are used (382 features), training for AP (or NDCG) consistently significantly outperforms training on PC(10) (or NDCG(10)), for all amounts of training data. Note that in general the informativeness of the optimization metric tends to matter more when less training data is available, as expected.

In order to validate our conclusions that the informativeness of the optimization metric is not important when there as a small number of features, we also ran our experiments on the OHSUMED dataset. This dataset contains 20 features most of which are highly correlated with each other, hence the effective number of features is much fewer than 20. Given this fact, the optimization metric is expected not to make a significant difference and our results validate this. Similar conclusions were reached when we used global optimization techniques (brute force search) with 2 features.

In order to further check the effect of the amount of training data versus the informativeness of the objective metric, we repeated the same set of experiments using training data with 30K queries with 382 features from a commercial search engine. We used the same validation/test set as in the previous experiments with 2K training data. Table 4 shows the result of this experiment. Even though the amount of training data is much larger in this case, the results obtained are still similar to the ones obtained using 2K dataset; training on the more informative metric resulting in a better test set performance for the target metric than training for the target metric itself. Note that all the above experiments are still performed using a limited amount of training data. Although training data with 30K queries is quite a large set, if we had even more training data available, we might reach different conclusions (Donmez et al. 2008). However, since obtaining relevance judgments is an expensive procedure, we almost always have a limited amount of training data available (especially for non-commercial usage). In the literature, many different methods have been proposed for learning with unlabeled data. Our results suggest that one way of decreasing the amount of judgment effort for obtaining training data is to use more informative metrics during training, especially when the number of features is large.

5.2 Local optimality of training versus target metrics

A natural question one might ask is how optimizing for a metric *X* compares with optimizing for another metric *Y* in terms of the optimality of *Y* in the training and test sets.

Figure 5 shows the learning curves when optimizing for each metric. These plots illustrate how the value of the test metric changes while training for another metric. For

Table 4 Lambdarank results on 30K Web data

Testmetric	AP	PC(10)	NDCG(10)
Train AP	**64.36**$^{\bullet\circ}$	**56.40**$^{\bullet\circ}$	64.94
Train PC(10)	63.98	56.12	64.65
Train NDCG	64.14	55.96	**65.55**$^{\bullet\circ}$
Train NDCG(10)	63.77	55.95	65.31

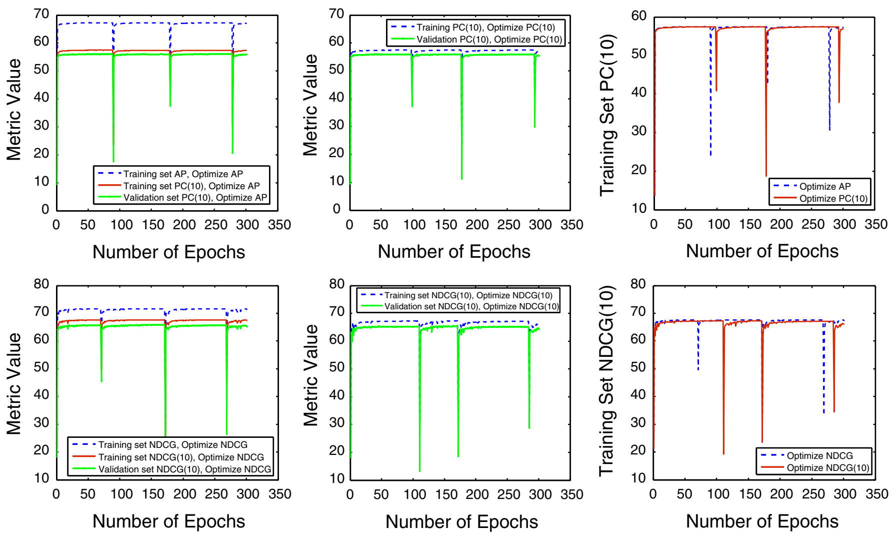

Fig. 5 (*Top*) Training and validation curves when optimizing for AP and PC(10). (*Bottom*) Training and validation curves when optimizing for NDCG and NDCG(10)

comparison purposes, we again mainly focus on the effect of training for average precision assuming the test metric is PC(10) (top left plot) compared to training for PC(10) (top middle plot) and the effect of training for NDCG assuming the test metric is NDCG(10) (bottom left plot) compared to training for NDCG(10) (bottom middle plot).

The dotted line (blue) at each plot shows how the value of the optimization metric changes in the training set and the solid line (red) shows how the value of the test metric changes in the same set at each training iteration (epoch). The dotted solid line (green) shows the value of the test metric in the validation set. We restart the optimization if we do not observe any improvement in the training set performance for 16 consecutive iterations. The sudden drops in the validation/training set values of the metrics are due to these restarts.

It can be seen that when optimizing for NDCG (or AP), the NDCG(10) (or PC(10)) values follow a very similar trend in the training set.

In Fig. 5 in the top right plot, we compare how the value of PC(10) changes when the learning algorithm is optimized for AP versus PC(10) during training and in the bottom right plot change in the NDCG(10) value when the algorithm is optimized for NDCG versus NDCG(10). These plots also show that when optimizing for AP (or NDCG), the learning algorithm is behaving in a very similar way to optimizing for PC(10) (or NDCG(10)) in terms of the values of these test metrics.

Intuitively, optimizing for NDCG (or AP) of the entire ranked list should implicitly correspond to optimizing for NDCG(10) (or PC(10)). A natural question to ask is ‘Is the local optimum for these more informative metrics also a local optimum for the less informative ones?’. If this is the case, then this also suggests that one should better optimize for these overall metrics, since the metrics that depend on the top end of the ranking will be implicitly optimized. In order to check whether the local optimum of a metric that evaluates the overall quality of a ranking (AP, NDCG) is also the local optimum for a metric that evaluates the top end of the ranking (PC(10), NDCG(10)), we use the Monte-Carlo test with one side error used by Donmez et al. (2008) to prove that Lambdarank finds the local optimum of an IR metric.

The Monte-Carlo test of local optimality can be described as follows: Given $\mathbf{w}$, the optimal weights found by the optimization algorithm, sample points from a unit sphere around $\mathbf{w}$. In order to do so, first form an n-dimensional vector by sampling each dimension of the vector from a Gaussian distribution with 0 mean unit variance and project it on to the unit sphere. Repeat this process k times, obtaining k vectors $\mathbf{v_1}, \mathbf{v_2}, \ldots \mathbf{v_k}$. Alter the optimal weights $\mathbf{w}$ by adding $\eta \cdot \mathbf{v_i}$ to $\mathbf{w}$ for small $\eta > 0$ and test whether there is any improvement on the value of the objective metric. The η value corresponding to the step size is varied ($\eta \in \{0.1, 0.2, \ldots, 1\}$).

Donmez et al. conclude that if no improvement on the value of the objective metric is observed more than a small $\varepsilon = 0.003$ when $k = 460$ random directions are used (resulting in 4,600 different alterations of the weight vector considering the step size), then the learned weights represent a local optimum with 99% confidence. Using such an ε value may be arguable for checking local optimality. However, such a value can be used considering the granularity of the IR metrics.

In our experiments, we used the Monte-Carlo test of local optimality to show whether optimizing for a metric Y also optimizes for a target metric X, both in training and testing sets. Table 5 shows the result of this experiment.

Given the optimal weights for a training metric in the training set, we also use 4,600 different alterations of the weight vector as described above and check whether we observe any improvement in the value of the test metric, both in the training set and in the test set. Out of the 4,600 different alterations, we report the number of cases when we see *any* improvement in the value of the test metric and an improvement greater than $\varepsilon = 0.003$. It can be seen that when using PC(10) as the test metric, no improvement $>\varepsilon$ was observed in the training set when training for AP or PC(10). So, according to the criterion by Donmez et al., when training for AP, the algorithm also finds the local optimum for PC(10) (we refer to this as ε-optimality). Similarly, when training for NDCG, the ε-optimum for NDCG(10) is also found.

However, it can be seen that when the weights are altered, there are some cases when we observe an improvement $<\varepsilon$ in the training set. Considering PC(10) as the test metric, the number of cases that we observe any improvement is more when training for AP than when training for PC(10) itself.

When the test set is considered given the optimal weights of the training metric in the training set, the number of cases where there is *any* improvement or an improvement $>\varepsilon$ in the PC(10) value is always smaller when AP is used as the training objective versus when PC(10) is used as the objective. In the case of AP for example, AP seems to be ε-optimal even in the test set. This suggests that even though training for AP is less likely to have found the local optimum for PC(10), as compared to training for PC(10) in the training set,

Table 5 Optimality results for PC(10) when training for AP versus PC(10) and NDCG(10) when training for NDCG and NDCG(10)

Train metric	Test metric	Trainset		Testset	
		>0	>ε	>0	>ε
AP	PC(10)	325	0	271	0
PC(10)	PC(10)	252	0	539	15
NDCG	NDCG(10)	226	0	590	17
NDCG(10)	NDCG(10)	150	0	670	22

it is more likely to have found to local optimum in the test set. This may be related to the fact that PC(10) does not utilize all the training data and since there may be documents in the test set with similar properties to those that were ignored during training, even when the weights are slightly altered, we are more likely to observe an improvement in the test set when PC(10) is used for training. Similar behavior can be seen for NDCG versus NDCG(10).

6 Conclusions

Most commonly used learning to rank algorithms are based on empirical risk minimization which suggests that if a metric M evaluates the quality of a search engine to an end user, then the engine should be trained to optimize for M.

We show that this assumption is not necessarily valid. Evaluation measures used in learning to rank act as bottlenecks that summarize the training data. It has been shown that some measures are more informative than others (Aslam et al. 2005). When the number of features is large, given a limited amount of training data, it is essential that the evaluation measure summarizes the limited data as good as possible. In this case, even when the ultimate metric that the user is interested in is M, one might better optimize for a more informative metric in the training set. We further analyze the local optima of different metrics and show that when training for a more informative metric in the training set, one is more likely to have found the local optimum for the less informative metric in the test set as compared to directly training for the less informative metric.

Our results suggest that the objective metrics used in optimization are not necessarily the metrics that evaluate user satisfaction and that more research should be devoted to developing evaluation measures that are more informative and better suited for learning to rank.

References

Aslam, J. A., Yilmaz, E., & Pavlu V. (2005). The maximum entropy method for analyzing retrieval measures. In Marchionini, G., Moffat A., Tait, J., Baeza-Yates, R., & Ziviani, N., (Eds.), *Proceedings of the 28th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 27–34). ACM Press, August 2005.

Burges, C. J. C., Ragno, R., & Le Q. V. (2006). Learning to rank with nonsmooth cost functions. In Schölkopf, B., Platt, J. C., & Hoffman, T. (Eds.), *NIPS* (pp. 193–200). Cambridge, MA: MIT Press.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., et al. (2005). Learning to rank using gradient descent. In *ICML '05: Proceedings of the 22nd international conference on machine learning* (pp. 89–96). New York, NY, USA: ACM.

Donmez, P., Svore, K., & Burges, C. J. (2008). On the optimality of lambdarank. Technical report, Microsoft Research.

Järvelin, K., & Kekäläinen, J. (2000). 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* (pp. 41–48). New York, NY, USA: ACM.

Joachims, T. (2005). A support vector method for multivariate performance measures. In *ICML '05: Proceedings of the 22nd international conference on Machine learning* (pp 377–384). New York, NY, USA: ACM.

Le, Q. V., & Smola. A. J. (2007). Direct optimization of ranking measures. *CoRR*, abs/0704.3359.

Ling, C. X., Huang, J., & Zhang, H. (2003). Auc: A statistically consistent and more discriminating measure than accuracy. In *IJCAI '03: Proceedings of the 18th international conference on artificial intelligence* (pp. 329–341).

Liu, T.-Y., & He Y. (2008). *Are algorithms directly optimizing ir measures really direct*? Technical report, Microsoft Research.

Robertson, S. (2008). A new interpretation of average precision. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 689–690). New York, NY, USA: ACM.

Robertson, S., & Zaragoza H. (2007). On rank-based effectiveness measures and optimization. *Information Retrieval, 10*(3), 321–339.

Sanderson, M., & Zobel J. (2005). Information retrieval system evaluation: Effort, sensitivity, and reliability. In Baeza-Yates, R. A., Ziviani, N., Marchionini, G., Moffat, A., & Tait, J. (Eds.), *SIGIR* (pp. 162–169). ACM.

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

Taylor, M., Zaragoza, H., Craswell, N., Robertson, S., & Burges, C. (2006). Optimisation methods for ranking functions with multiple parameters. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management* (pp. 585–593). New York, NY, USA: ACM

Webber, W., Moffat, A., Zobel, J., & Sakai, T. (2008). Precision-at-ten considered redundant. In *SIGIR* (pp. 695–696). New York, NY, USA: ACM.

Xu, J., & Li, H. (2007). Adarank: a boosting algorithm for information retrieval. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 391–398). New York, NY, USA: ACM.

Yilmaz, E., & Aslam J. A. (2008). Estimating average precision when judgments are incomplete. *Knowledge and Information Systems, 16*(2), 173–211, August 2008.

Yue, Y., Finleym, T., Radlinski, F., & Joachims, T. (2007). 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 information retrieval* (pp. 271–278). New York, NY, USA: ACM.