A Robust Ranking Methodology based on Diverse Calibration of AdaBoost

Róbert Busa-Fekete, Balázs Kégl, Tamas Elteto, György Szarvas

A Robust Ranking Methodology based on Diverse Calibration of AdaBoost

Róbert Busa-Fekete[1,2], Balázs Kégl[1,3], Tamás Éltető[3], and György Szarvas[2,4]

[1] Linear Accelerator Laboratory (LAL), University of Paris-Sud, CNRS Orsay, 91898, France
[2] Research Group on Artificial Intelligence of the Hungarian Academy of Sciences and University of Szeged, Aradi vértanúk tere 1., H-6720 Szeged, Hungary
[3] Computer Science Laboratory (LRI), University of Paris-Sud, CNRS and INRIA-Saclay, 91405 Orsay, France
[4] Ubiquitous Knowledge Processing (UKP) Lab, Computer Science Department Technische Universität Darmstadt, D-64289 Darmstadt, Germany

Abstract. In *subset ranking*, the goal is to learn a ranking function that approximates a gold standard partial ordering of a set of objects (in our case, relevance labels of a set of documents retrieved for the same query). In this paper we introduce a learning to rank approach to subset ranking based on multi-class classification. Our technique can be summarized in three major steps. First, a multi-class classification model (AdaBoost.MH) is trained to predict the relevance label of each object. Second, the trained model is calibrated using various calibration techniques to obtain diverse class probability estimates. Finally, the Bayes-scoring function (which optimizes the popular Information Retrieval performance measure NDCG), is approximated through mixing these estimates into an ultimate scoring function. An important novelty of our approach is that many different methods are applied to estimate the same probability distribution, and all these hypotheses are combined into an improved model. It is well known that mixing different conditional distributions according to a prior is usually more efficient than selecting one "optimal" distribution. Accordingly, using all the calibration techniques, our approach does not require the estimation of the best suited calibration method and is therefore less prone to overfitting. In an experimental study, our method outperformed many standard ranking algorithms on the LETOR benchmark datasets, most of which are based on significantly more complex learning to rank algorithms than ours.

Keywords: Learning-to-rank, AdaBoost, Class Probability Calibration

1 Introduction

In the past, the result lists in Information Retrieval were ranked by probabilistic models, such as the BM25 measure [16], based on a small number of attributes (the frequency of query terms in the document, in the collection, etc.). The parameters of these models were usually set empirically. As the number of useful features increase, these manually crafted models become increasingly laborious

to configure. Alternatively, one can use as many (possibly redundant) attributes as possible, and employ Machine Learning techniques to induce a ranking model. This approach alleviates the human effort needed to design the ranking function, and also provides a natural way to directly optimize the retrieval performance for any particular application and evaluation metric. As a result, Learning to Rank has gained considerable research interest in the past decade.

Machine Learning based ranking systems are traditionally classified into three categories. In the simplest *pointwise* approach, the instances are first assigned a relevance score using classical regression or classification techniques, and then ranked by posterior scores obtained using the trained model [12]. In the *pairwise* approach, the order of pairs of instances is treated as a binary label and learned by a classification method [8]. Finally, in the most complex *listwise* approach, the fully ranked lists are learned by a tailor-made learning method which aims to optimize a ranking-specific evaluation metric during the learning process [18].

In web page ranking or subset ranking [7] the training data is given in the form of query-document-relevance label triplets. The relevance label of a training instance indicates the usefulness of the document to its corresponding query, and the ranking for a particular query is usually evaluated using the (normalized) Discounted Cumulative Gain ((N)DCG) or the Expected Reciprocal Rank (ERR) [5] measures. It is rather difficult to extend classical learning methods to directly optimize these evaluation metrics. However, since the DCG can be bounded by the $0-1$ loss [12], the traditional classification error can be considered as a surrogate function of DCG to be minimized.

Calibrating the output of a particular learning method, such as Support Vector Machines or AdaBoost, is crucial in applications with quality measures different from the $0-1$ error [14]. Our approach is based on the calibration of a multi-class classification model, AdaBoost [9]. In our setup, the class labels are assumed to be random variables and the goal is the estimation of the probability distribution of the class labels given a feature vector. An important novelty in our approach is that instead of using a single calibration technique, we apply several methods to estimate the same probability distribution and, in a final step, we combine these estimates. Both the Bayesian paradigm and the Minimum Description Length principle [15] suggest that it is usually more efficient to mix different conditional distributions according to a prior than to select one "optimal" distribution.

We use both *regression-based calibration (RBC)* and *class probability-based calibration (CPC)* to transform the output scores of AdaBoost into relevance label estimates that are comparable to each other. In the case of RBC, the real-valued scores are obtained by a regression function fit to the output of AdaBoost as the independent variable and the relevance labels as the dependent variable. These scores are then used to rank the objects. In the case of CPC, the posterior probability distribution is used to approximate the so-called Bayes-scoring function [7], which is shown to optimize the expected DCG in a probabilistic setup.

The proper choice of the prior on the set of conditional distributions obtained by the calibration of AdaBoost is an important decision in practice. In this paper, we use an exponential scheme based on the quality of the rankings implied by the conditional distributions (via their corresponding conditional ranking functions) which is theoretically more well-founded than the uniformly weighted aggregation used by McRank [12].

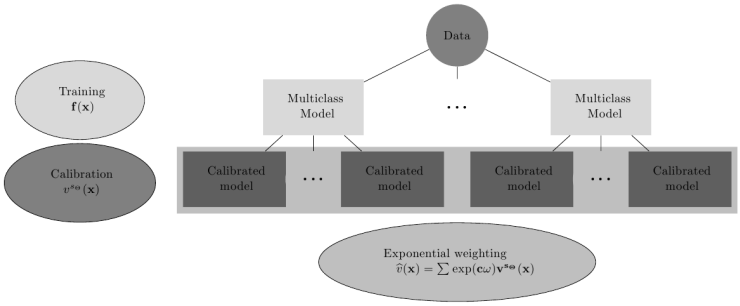


Fig. 1. The schematic overview of our approach. In the first level a multi-class method (AdaBoost.MH) is trained using different hyperparameter settings. Then we calibrate the multi-class models in many ways to obtain diverse scoring functions. In the last step we simply aggregate the scoring functions using an exponential weighting.

Figure 1 provides a structural overview of our system. Our approach belongs to the simplest, pointwise category of learning-to-rank models. It is based on a series of standard techniques (i) multiclass classification, ii) output score calibration and iii) exponentially weighted forecaster to combine the various hypotheses. As opposed to previous studies, we found our approach to be competitive to the standard methods (AdaRank, ListNET, rankSVM, and rankBoost) of the theoretically more complex pairwise and listwise approaches.

We attribute this surprising result to the presence of label noise in the ranking task and the robustness of our approach to this noise. Label noise is inherent in web page ranking for multiple reasons. First, the relevance labels depend on human decisions, and so they are subjective and noisy. Second, the common feature representations account only for simple keyword matches (such as query term frequency in the document) or query-independent measures (such as PageRank) that are unable to capture query-document relations with more complex semantics like the use of synonyms, analogy, etc. Query-document pairs that are not characterized well by the features can be considered as noise from the perspective of the learning algorithm. Our method suites especially well to practical problems with label noise due to the robustness guaranteed by the meta-ensemble step that combines a wide variety of hypotheses. As our results demonstrate, it compares favorably to theoretically more complex approaches.

The paper is organized as follows. In Section 2 we provide a brief overview of the related works. Section 3 describes the formal setup. Section 4 is devoted to the description of calibration technique. The ensemble scheme is presented

in Section 5. In Section 6 we investigate the theoretical properties of our CPC approach. Our experimental results are presented in Section 7 and we draw conclusions in Section 8.

2 Related work

Among the plethora of ranking algorithms, our approach is the closest to the McRank algorithm [12]. We both use a multi-class classification algorithm at the core (they use gradient boosting whereas we apply AdaBoost.MH). The major novelties in our approach are that we use product base classifiers besides the popular decision tree base classifiers and apply several different calibration approaches. Both elements add more diversity to our models that we exploit by a final meta-ensemble technique. In addition, MCRank's implementation is inefficient in the sense that the number of decision trees trained in each boosting iteration is as large as the number of different classes in the dataset.

Even though McRank is not considered a state-of-the-art method itself, its importance is unquestionable. It can be viewed as a milestone which proved the raison d'etre of classification based learning-to-rank methods. It attracted the attention of researchers working on learning-to-rank to classification-based ranking algorithms. The most remarkable method motivated by McRank is LambdaMart [19], which adapts the MART algorithm to the subset ranking problem. In the Yahoo! Learning-to-rank Challenge this method achieved the best performance in the first track [4].

In the Yahoo! challenge [4], a general conclusion was that listwise and pairwise methods achieved the best scores in general, but tailor-made pointwise approaches also achieved very competitive results. In particular, the approach presented here is based on our previous work [1]. The main contributions of this work are that we evaluate a state of the art multiclass classification based approach on publicly available benchmark datasets, and that we present a novel calibration approach, namely sigmoid-based class probability calibration (CPC), which is theoretically better grounded than regression-based calibration. We also provide an upper bound on the difference between the DCG value of the Bayes optimal score function and the DCG value achieved by its estimate using CPC.

In the LambdaMART [19] paper there is a second interesting contribution, namely a linear combination scheme for two rankers with an $O(n^2)$ algorithm where n is the number of documents. This method is simply based on a line search optimization among the convex combination of two rankers. This ranking combination is then used for adjusting the weights of weak learners. This combination method has the appealing property that it gives optimal convex combination of two rankers. However, it is not obvious how to extend it for more than two rankers, so it is not directly applicable to our setting.

3 Definition of the ranking problem

In this section we briefly summarize the part of [7] relevant to our approach, and, at the same time, we introduce the notation that will be used in the rest of the paper.

Let us assume that we are given a set of *query objects* $\boldsymbol{Q} = \{Q^1, \ldots, Q^M\}$. For a query object Q^k we will define the set of feature vectors

$$D^k = \{\boldsymbol{x}_1^k, \ldots, \boldsymbol{x}_j^k, \ldots, \boldsymbol{x}_{m^k}^k\},$$

where $\boldsymbol{x}_j^k$ are the real valued feature vectors that encode the set of documents retrieved for Q^k. The upper index will always refer to the query index. When it is not confusing, we will omit the query index and simply write x_j for the jth document of a given query.

The *relevance grade* of $\boldsymbol{x}_j^k$ is denoted by y_j^k. The set of possible relevance grades is $\mathfrak{Y} = \{\gamma_1, \ldots, \gamma_K\}$, usually referred to as *relevance labels*, i.e. integer numbers up to a threshold: $\ell = 1, \ldots, K$. In this case, the relation between the relevance grades and relevance labels is $y_j^k = 2^{\ell_j^k} + 1$, where ℓ_j^k is the relevance label for response j given a query Q^k.

The goal of the ranker is to output a permutation $J = [j_1, \ldots, j_m]$ over the integers $(1, \ldots, m)$. The Discounted Cumulative Gain (DCG) is defined as

$$\mathrm{DCG}(J, [y_{j_i}]) = \sum_{i=1}^{m} c_i y_{j_i}, \tag{1}$$

where c_i is the discount factor of the i^{th} document in the permutation. The most commonly used discount factor is $c_i = \frac{1}{\log(1+i)}$. One can also define the normalized DCG (NDCG) score by dividing (1) with the DCG score of the best permutation.

We will consider y_j^k as a random variable with discrete probability distribution $P[y_j^k = \gamma | \boldsymbol{x}_j^k] = p^*_{y_j^k|\boldsymbol{x}_j^k}(\gamma)$ over the relevance grades for document j and query Q^k. The *Bayes-scoring function* is defined as

$$v^*(\boldsymbol{x}_j^k) = \mathbb{E}_{p^*_{y_j^k|\boldsymbol{x}_j^k}}\left\{y_j^k\right\} = \sum_{\gamma \in \mathfrak{Y}} \gamma p^*_{y_j^k|\boldsymbol{x}_j^k}(\gamma).$$

Since y_j^K is a random variable, we can define the expected DCG for any permutation $J = [j_1, \ldots, j_{m^k}]$ as

$$\mathrm{DCG}(J, [y_{j_i}^k]) = \sum_{i=1}^{m} c_i \mathbb{E}_{p^*_{y_{j_i}^k|\boldsymbol{x}_{j_i}^k}}\left\{y_{j_i}^k\right\} = \sum_{i=1}^{m} c_i v^*(\boldsymbol{x}_{j_i}^k).$$

Let the optimal Bayes permutation $J^{*k} = [j_1^{*k}, \ldots, j_{m^k}^{*k}]$ over the documents of query Q^k be the one which maximizes the expected DCG-value, that is,

$$J^{*k} = \arg\max_J \mathrm{DCG}(J, [y_{j_i^k}]).$$

According to Theorem 1 of [7], J^{*k} has the property that if $c_i > c_{i'}$ then for the Bayes-scoring function it holds that $v^*(\boldsymbol{x}_{j_i^{*k}}) > v^*(\boldsymbol{x}_{j_{i'}^{*k}})$. Our goal is to estimate $p^*_{y_j^k|\boldsymbol{x}_j^k}(\gamma)$ by $p^{\mathcal{A}}_{y_j^k|\boldsymbol{x}_j^k}(\gamma)$, which defines the following scoring function

$$v^{\mathcal{A}}(\boldsymbol{x}_j^k) = \mathbb{E}_{p^{\mathcal{A}}_{y_j^k|\boldsymbol{x}_j^k}}\left\{y_j^k\right\} = \sum_{\gamma \in \mathfrak{Y}} \gamma p^{\mathcal{A}}_{y_j^k|\boldsymbol{x}_j^k}(\gamma), \tag{2}$$

where the label $\mathcal{A}$ will refer to the method that generates the probability estimates.

4 The calibration of multi-class classification models

Our basic modeling tool is multi-class AdaBoost.MH introduced by [17]. The input training set is the set of feature vectors $\mathbf{X} = \left(\mathbf{x}_1^1, \ldots, \mathbf{x}_{m_1}^1, \ldots, \mathbf{x}_1^M, \ldots, \mathbf{x}_{m_M}^M\right)$ and a set of labels $\boldsymbol{Z} = \left(\mathbf{z}_1^1, \ldots, \mathbf{z}_{m_1}^1, \ldots, \mathbf{z}_1^M, \ldots, \mathbf{z}_{m_M}^M\right)$. Each feature vector $\mathbf{x}_k^j \in \mathbb{R}^d$ encodes a (query, document) pair. Each label vector $\mathbf{z}_j^k \in \{+1, -1\}^K$ encodes the relevance label using a one-out-of-K scheme, that is, $z_{j,\ell}^k = 1$ if $\ell_j^k = \ell$ and -1 otherwise. We used two well-boostable base learners, i.e. *decision trees* and *decision products* [11]. Instead of using uniform weighting for training instances, we up-weighted relevant instances exponentially proportionally to their relevance, so, for example, an instance $\mathbf{x}_k^j$ with relevance $\ell_k^j = 3$ was twice as important in the global training cost than an instance with relevance $\ell_k^j = 2$, and four times as important than an instance with relevance $\ell_k^j = 1$. Formally, the initial (unnormalized) weight of ℓth label of the $\mathbf{x}_k^j$th instance is

$$w_{j,\ell}^k = \begin{cases} 2^{\ell_j^k} & \text{if } \ell_j^k = \ell, \\ 2^{\ell_j^k}/(K-1) & \text{otherwise.} \end{cases}$$

The weights are then normalized to sum to 1. This weighting scheme was motivated by the evaluation metric: the weight of an instance in the NDCG score is exponentially proportional to the relevance label of the instance itself.

AdaBoost.MH outputs a strong classifier $\mathbf{f}^{(T)}(\boldsymbol{x}) = \sum_{t=1}^T \alpha^{(t)} \mathbf{h}^{(t)}(\boldsymbol{x})$, where $\mathbf{h}^{(t)}(\boldsymbol{x})$ is a $\{-1, +1\}^K$-valued base classifier (K is number of relevance label values), and $\alpha^{(t)}$ is its weight. In multi-class classification the elements of $\mathbf{f}^{(T)}(\boldsymbol{x})$ are treated as posterior scores corresponding to the labels, and the predicted label is $\widehat{\ell}_j^k = \arg\max_{\ell=1,\ldots,K} f_\ell^{(T)}(\boldsymbol{x}_j^k)$. When posterior probability estimates are required, in the simplest case the output vector can be shifted into $[0,1]^K$ using

$$\mathbf{f}'^{(T)}(\boldsymbol{x}) = \frac{1}{2}\left[1 + \frac{\mathbf{f}^{(T)}(\boldsymbol{x})}{\sum_{t=1}^T \alpha^{(t)}}\right],$$

and then the posterior probabilities can be obtained by simple normalization

$$p_{y_j^k|\boldsymbol{x}_j^k}^{\text{standard}}(\gamma_\ell) = \frac{f'^{(T)}_\ell(\boldsymbol{x}_j^k)}{\sum_{\ell'=1}^K f'^{(T)}_{\ell'}(\boldsymbol{x}_j^k)}. \tag{3}$$

4.1 Regression based pointwise calibration

An obvious way to calibrate the AdaBoost output is to re-learn the relevance grades by applying a regression method. Using the raw K-dimensional output of AdaBoost we can obtain relevance grade estimates in the form of $y_j^k \approx$

$g\left(\mathbf{f}^{(T)}(\boldsymbol{x}_j^k)\right)$. where $g : \mathbb{R}^K \to \mathbb{R}$ is the regression function. We shall refer to this scheme as *regression-based calibration* (RBC).

In our experiments we used five different regression methods: Gaussian process regression, logistic regression, linear regression, neural network regression, and polynomial regression of degree between 2 and 5. From a practical point of view, the relevance grade estimates provided by the different regression methods for an unseen test document are on a different scale, so these values cannot be aggregated in a direct way. In Section 5 we describe a way we normalized these values.

4.2 Class Probability Calibration and its implementation

Let us recall that the output of AdaBoost.MH is $\mathbf{f}\left(\boldsymbol{x}_i^k\right) = \left[f_\ell\left(\boldsymbol{x}_i^k\right)\right]_{\ell=1,\ldots,K}$. Then, the class-probability-based sigmoidal calibration for label ℓ is

$$p^{s_\Theta}_{y_i^k|\boldsymbol{x}_i^k}(\gamma_\ell) = \frac{s_\Theta(f_\ell(\boldsymbol{x}_i^k))}{\sum_{\ell'=i}^{K} s_\Theta(f_{\ell'}(\boldsymbol{x}_i^k))} \tag{4}$$

where the sigmoid function can be written as $s_{\Theta=\{a,b\}}(x) = \frac{1}{1+\exp(-a(x-b))}$. The parameters of the sigmoid function can be tuned by minimizing a so-called *target calibration function (TCF)* $L^{\mathcal{A}}(\Theta, \mathbf{f})$. Generally speaking, $L^{\mathcal{A}}(\Theta, \mathbf{f})$ is a loss function calculated using the relevance label probability distribution estimates defined in (4). $L^{\mathcal{A}}$ is parametrized by the parameter Θ of sigmoid function and the multi-class classifier $\mathbf{f}$ output by AdaBoost. $L^{\mathcal{A}}(\Theta, \mathbf{f})$ is also naturally a function of the validation data set (which is not necessarily the same as the training set), but we will omit this dependency for easing the notation.

Given a TCF $L^{\mathcal{A}}$ and a multi-class classifier $\mathbf{f}$, our goal is to find the optimal calibration parameters

$$\Theta^{\mathcal{A},\mathbf{f}} = \arg\min_{\Theta} L^{\mathcal{A}}(\Theta, \mathbf{f}).$$

The output of this calibration step is a probability distribution $p^{\mathcal{A},\mathbf{f}}_{y_i^k|\boldsymbol{x}_i^k}(\cdot)$ on the relevance grades for each document in each query and a Bayes-scoring function $v^{\mathcal{A},\mathbf{f}}(\cdot)$ defined in (2). We will refer to this scheme as *class probability-based calibration* (CPC). The upper index $\mathcal{A}$ refers to the type of the particular TCF, so the *ensemble* of probability distributions is indexed by the type $\mathcal{A}$ and the multi-class classifier $\mathbf{f}$ output by AdaBoost.

Given the ensemble of the probability distributions $p^{\mathcal{A},\mathbf{f}}_{y_i^k|\boldsymbol{x}_i^k}(\cdot)$ and an appropriately chosen prior $\pi(\mathcal{A}, \mathbf{f})$, we follow a Bayesian approach and calculate a posterior conditional distribution by

$$p^{\text{posterior}}_{y_i^k|\boldsymbol{x}_i^k}(\cdot) = \sum_{\mathcal{A},\mathbf{f}} \pi(\mathcal{A}, \mathbf{f}) p^{\mathcal{A},\mathbf{f}}_{y_i^k|\boldsymbol{x}_i^k}(\cdot).$$

Then we obtain a "posterior" estimate for the Bayes-scoring function

$$v^{\text{posterior}}(\boldsymbol{x}_i^k) = \sum_{\ell=1}^{K} \gamma_\ell p^{\text{posterior}}_{y_i^k|\boldsymbol{x}_i^k}(\gamma_\ell) = \sum_{\ell=1}^{K} \gamma_\ell \sum_{\mathcal{A},\mathbf{f}} \pi(\mathcal{A}, \mathbf{f}) p^{\mathcal{A},\mathbf{f}}_{y_i^k|\boldsymbol{x}_i^k}(\boldsymbol{x}_i^k).$$

which can be written as

$$v^{\text{posterior}}(\boldsymbol{x}_i^k) = \sum_{\mathcal{A},\mathbf{f}} \pi(\mathcal{A},\mathbf{f}) \sum_{\ell=1}^{K} \gamma_\ell p_{y_i^k|\boldsymbol{x}_i^k}^{\mathcal{A},\mathbf{f}}(\boldsymbol{x}_i^k) = \sum_{\mathcal{A},\mathbf{f}} \pi(\mathcal{A},\mathbf{f}) v^{\mathcal{A},\mathbf{f}}(\boldsymbol{x}_i^k). \quad (5)$$

The proper selection of the prior $\pi(\cdot,\cdot)$ can further increase the quality of the posterior estimation. In Section 5 we will describe a reasonable prior definition borrowed from the theory of experts.

In the simplest case, the TCF can be

$$L^{\text{LS}}(\Theta,\mathbf{f}) = \sum_{k=1}^{M}\sum_{i=1}^{m_k} -\log \frac{s_\Theta(f_{\ell_i^k}(\boldsymbol{x}_i^k))}{\sum_{\ell'=1}^{K} s_\Theta(f_{\ell'}(\boldsymbol{x}_i^k))}.$$

We refer to this function as the *log-sigmoid TCF*. The motivation of the log-sigmoid TCF is that the resulting probability distribution minimizes the relative entropy

$$D(p^*||p) = -\sum_{k=1}^{M}\sum_{i=1}^{m_k}\sum_{\ell} p^*_{y_i^k|\boldsymbol{x}_i^k}(\gamma_\ell)\log p_{y_i^k|\boldsymbol{x}_i^k}(\gamma_\ell) + p^*_{y_i^k|\boldsymbol{x}_i^k}(\gamma_\ell)\log p^*_{y_i^k|\boldsymbol{x}_i^k}(\gamma_\ell),$$

between the Bayes optimal probability distribution p^* and p. In practice distributions being less (or more) uniform over the labels might be preferred. This preference can be expressed by introducing the entropy weighted version of the log-sigmoid TCF, that can be written as

$$L_C^{\text{EWLS}}(\Theta) = \sum_{k=1}^{M}\sum_{i=1}^{m_k} -\log \frac{s_\Theta(f_{\ell_i^k}(\boldsymbol{x}_i^k))}{\sum_{\ell'=1}^{K} s_\Theta(f_{\ell'}(\boldsymbol{x}_i^k))} \times H_M\left(\frac{s_\Theta\left(f_1\left(\boldsymbol{x}_i^k\right)\right)}{\sum_{\ell'=i}^{K} s_\Theta(f_{\ell'}(\boldsymbol{x}_i^k))},\ldots,\frac{s_\Theta\left(f_K\left(\boldsymbol{x}_i^k\right)\right)}{\sum_{\ell'=i}^{K} s_\Theta(f_{\ell'}(\boldsymbol{x}_i^k))}\right)^C,$$

where $H_M(p_1,\ldots,p_K) = \sum_{\ell=1}^{K}\left[p_\ell\left(-\log p_\ell\right)\right]$, and C is a hyperparameter. The minimization in LS and EWLS TCF can be considered as an attempt to minimize a cost function, the sum of the negative logarithms of the class probabilities. Usually, there is a cost function for misclassification associated to the learning task. This cost function can be used for defining the *expected loss TCF*

$$L^{\text{EL}}(\Theta) = \sum_{k=1}^{M}\sum_{i=1}^{m_k}\sum_{\ell=1}^{K} \frac{\mathcal{L}\left(\ell,\ell_i^k\right) s_\Theta(f_\ell(\boldsymbol{x}_i^k))}{\sum_{\ell'=1}^{K} s_\Theta(f_{\ell'}(\boldsymbol{x}_i^k))},$$

where ℓ_i^k is the correct label of $\boldsymbol{x}_i^k$, and $\mathcal{L}(\ell,\ell_i^k)$ expresses the loss if ℓ is predicted instead of the correct label. We used the standard square loss ($\mathcal{L}^2$) setup, so $\mathcal{L}(\ell,\ell') = (\ell-\ell')^2$.

If the labels have some structure, e.g., they are ordinal as in our case, it is possible to calculate an expected label based on a CPC distribution. In this case

we can define the *expected label loss TCF*

$$L^{\mathrm{ELL}}(\Theta) = \sum_{k=1}^{M}\sum_{i=1}^{m_k} \mathcal{L}\left(\sum_{\ell=1}^{K} \frac{\ell s_\Theta(f_\ell(\boldsymbol{x}_i^k))}{\sum_{\ell'=1}^{K} s_\Theta(f_{\ell'}(\boldsymbol{x}_i^k))}, \ell_i^k\right).$$

Here, the goal is to minimize the incurred loss between the expected label and the correct label ℓ_i^k. We used here also $\mathcal{L}^2$ as loss function. Note that the definition of $\mathcal{L}(\cdot,\cdot)$ might need to be updated if a weighted average of labels is to be calculated, as the weighted average might not be a label at all.

Finally, we can apply the idea of SmoothGrad [6] to obtain a TCF. In SmoothGrad a smooth surrogate function is used to optimize the NDCG metric. In particular, the *soft indicator variable* can be written as

$$h_{\Theta,\sigma}(\boldsymbol{x}_i^k, \boldsymbol{x}_{i'}^k) = \frac{\exp\left(-\frac{\left(v^{s_\Theta}(\boldsymbol{x}_i^k) - v^{s_\Theta}(\boldsymbol{x}_{i'}^k)\right)^2}{\sigma}\right)}{\sum_{j=1}^{m^k}\exp\left(-\frac{\left(v^{s_\Theta}(\boldsymbol{x}_j^k) - v^{s_\Theta}(\boldsymbol{x}_{i'}^k)\right)^2}{\sigma}\right)}.$$

Then the TCF can be written in the form:

$$L_\sigma^{\mathrm{SN}}(\Theta) = -\sum_{k=1}^{M}\sum_{i=1}^{m_k}\sum_{t=1}^{m_k} y_i^k c_t h_{\Theta,\sigma}(\boldsymbol{x}_i^k, \boldsymbol{x}_{j_t}^k)$$

where $J = [j_1, \ldots, j_m]$ is the permutation based on the scoring function $v^{s_\Theta}(\cdot)$. The parameter σ controls the smoothness of L_σ^{SN}. That is, the higher σ, the smoother is the function but the bigger is the difference between the NDCG value and the value of surrogate function. If $\sigma \to 0$ then L_σ^{SN} tends to the NDCG value, but, at the same time, to optimize the surrogate function becomes harder. We refer this TCF as the *SN target calibration* function.

Note that in our experience, the more diverse the set of TCFs, the better is the performance of the ultimate scoring function, and we could use other, even fundamentally different TCFs in our system.

5 Ensemble of ensembles

The output of calibration is a set of relevance predictions $v^{\mathcal{A},\mathbf{f}}(\boldsymbol{x}, S)$ for each TCF type $\mathcal{A}$ and AdaBoost output $\mathbf{f}$. Each relevance prediction can be used as a scoring function to rank the query-document pairs represented by the vector $\mathbf{x}_i^k$. Until this point it is a pure pointwise approach except that the smoothed version of NDCG was optimized in SN. To fine-tune the algorithm and to make use of the diversity of our models, we combine them using an exponentially weighted forecaster [3]. The reason of using this particular weighting scheme is twofold. First, it is simple and computationally efficient to tune which is important when we have a large number of models. Second, theoretical guarantees over the cumulative regret of a mixture of experts on individual (model-less) sequences [3] makes the technique robust against overfitting the validation set.

The weights of the models are tuned on the NDCG score of the ranking, giving a slight listwise touch to our approach. Formally, the final scoring function is obtained by using $\pi(X) = \exp(c\omega^{\mathcal{A},\mathbf{f}})$. Plugging it into (5),

$$v^{\text{posterior}}(\mathbf{x}) = \sum_{\mathcal{A},\mathbf{f}} \exp(c\omega^{\mathcal{A},\mathbf{f}}) v^{\mathcal{A},\mathbf{f}}(\boldsymbol{x}), \tag{6}$$

where $\omega^{\mathcal{A},\mathbf{f}}$ is the NDCG_{10} score of the ranking obtained by using $v^{\mathcal{A},\mathbf{f}}(\boldsymbol{x})$. The parameter c controls the dependence of the weights on the NDCG_{10} values.

A similar ensemble method can be applied to outputs calibrated by regression methods. A major difference between the two types of calibration is that the regression-based scores have to be normalized/rescaled before the exponentially weighted ensemble scheme is applied. We simply rescaled the output of the regression models into $[0, 1]$ before using them in the exponential ensemble scheme.

6 DCG Bound for Class Probability Estimation

In (2) we described a way to obtain a scoring function v based on the estimate of the probability distribution of relevance grades. Based on the estimated scoring function v, it is straightforward to obtain a ranking and the associated permutation on the set of documents D.[5] More formally, let $J^v = [j_1^v, \ldots, j_m^v]$ be such that if $j_k^v > j_{k'}^v$ then $v(\boldsymbol{x}_{j_k}) > v(\boldsymbol{x}_{j_{k'}})$.

The following proposition gives an upper bound for the difference between DCG value of the Bayes optimal score function and the DCG value achieved by its estimate in terms of the quality of the relevance probability estimate.
Proposition: Let $p, q \in [1, \infty]$ and $1/p + 1/q = 1$. Then

$$\text{DCG}(J^*, [y_{j_i^*}]) - \text{DCG}(J, [y_{j_i^v}]) \leq$$

$$\leq \left(\sum_{i=1}^{m} \sum_{\gamma \in \mathfrak{Y}} \left| (c_{\widetilde{j}_i^v} - c_{\widetilde{j}_i^*}) \gamma \right|^p \right)^{\frac{1}{p}} \left(\sum_{i=1}^{m} \sum_{\gamma \in \mathfrak{Y}} \left| p_{y_i|\boldsymbol{x}_i}(\gamma) - p^*_{y_i|\boldsymbol{x}_i}(\gamma) \right|^q \right)^{\frac{1}{q}},$$

where $\widetilde{j}_i^v$ and $\widetilde{j}_i^*$ are the inverse permutations of j_i^v and j_i^*. The relation between $p_{y_i|\boldsymbol{x}_i}(\cdot)$ and $v(\cdot)$ is defined in (2).
Proof: Following the lines of Theorem 2 of the [7],

$$\begin{aligned}
\text{DCG}(J, [y_{j_i}]) &= \sum_{i=1}^{m} c_i v^*(\boldsymbol{x}_{j_i^v}) = \sum_{i=1}^{m} c_i v(\boldsymbol{x}_{j_i^v}) + \sum_{i=1}^{m} c_i (v^*(\boldsymbol{x}_{j_i^v}) - v(\boldsymbol{x}_{j_i^v})) \\
&\geq \sum_{i=1}^{m} c_i v(\boldsymbol{x}_{j_i^*}) + \sum_{i=1}^{m} c_i (v^*(\boldsymbol{x}_{j_i^v}) - v(\boldsymbol{x}_{j_i^v})) \\
&= \sum_{i=1}^{m} c_i v^*(\boldsymbol{x}_{j_i^*}) + \sum_{i=1}^{m} c_i (v^*(\boldsymbol{x}_{j_i^v}) - v(\boldsymbol{x}_{j_i^v})) + \sum_{i=1}^{m} c_i (v(\boldsymbol{x}_{j_i^*}) - v^*(\boldsymbol{x}_{j_i^*}))
\end{aligned}$$

[5] In this section we omit the indexing over the queries.

$$= \mathrm{DCG}(J^*, [y_{j_i^*}]) + \sum_{i=1}^{m} c_i(v^*(\boldsymbol{x}_{j_i^v}) - v(\boldsymbol{x}_{j_i^v})) + \sum_{i=1}^{m} c_i(v(\boldsymbol{x}_{j_i^*}) - v^*(\boldsymbol{x}_{j_i^*})).$$

Here $\sum_{i=1}^{m} c_i v(\boldsymbol{x}_{j_i^v}) \geq \sum_{i=1}^{m} c_i v(\boldsymbol{x}_{j_i^*})$, because J^v is an optimal permutation for scoring function v. Then,

$$\mathrm{DCG}(J^*, [y_{j_i^*}]) - \mathrm{DCG}(J^v, [y_{j_i^v}]) \leq \sum_{i=1}^{m} (c_{\widetilde{j}_i^v} - c_{\widetilde{j}_i^*})(v(\boldsymbol{x}_i) - v^*(\boldsymbol{x}_i))$$

$$= \sum_{i=1}^{m} \sum_{\gamma \in \mathfrak{Y}} (c_{\widetilde{j}_i^v} - c_{\widetilde{j}_i^*})\gamma(p_{y_i|\boldsymbol{x}_i}(\gamma) - p^*_{y_i|\boldsymbol{x}_i}(\gamma)),$$

where $\widetilde{j}_i^v$ and $\widetilde{j}_i^*$ are the inverse permutations of j_i^v and j_i^*. Then, the Hölder inequality implies, that

$$\sum_{i=1}^{m} \sum_{\gamma \in \mathfrak{Y}} \left| (c_{\widetilde{j}_i^v} - c_{\widetilde{j}_i^*})\gamma(p_{y_i|\boldsymbol{x}_i}(\gamma) - p^*_{y_i|\boldsymbol{x}_i}(\gamma)) \right|$$

$$\leq \left(\sum_{i=1}^{m} \sum_{\gamma \in \mathfrak{Y}} \left| (c_{\widetilde{j}_i^v} - c_{\widetilde{j}_i^*})\gamma \right|^p \right)^{\frac{1}{p}} \left(\sum_{i=1}^{m} \sum_{\gamma \in \mathfrak{Y}} \left| p_{y_i|\boldsymbol{x}_i}(\gamma) - p^*_{y_i|\boldsymbol{x}_i}(\gamma) \right|^q \right)^{\frac{1}{q}} .\square$$

Corollary:

$$\mathrm{DCG}(J^*, [y_{j_i^*}]) - \mathrm{DCG}(J^v, [y_{j_i^v}]) \leq C \cdot \left(\sum_{i=1}^{m} \sum_{\gamma \in \mathfrak{Y}} \left| p_{y_i|\boldsymbol{x}_i}(\gamma) - p^*_{y_i|\boldsymbol{x}_i}(\gamma) \right|^q \right)^{\frac{1}{q}},$$

where

$$C = \max_{\widetilde{j}^v, \widetilde{j}^*} \left(\sum_{i=1}^{m} \sum_{\gamma \in \mathfrak{Y}} \left| (c_{\widetilde{j}_i^v} - c_{\widetilde{j}_i^*})\gamma \right|^p \right)^{\frac{1}{p}},$$

$\widetilde{j}^v$ and $\widetilde{j}^*$ are the permutations of $1, \ldots, m$.

The Corollary shows that as the distance between the "exact" and the estimated conditional distributions over the relevance labels tends to 0, the difference in the DCG values also tends to 0.

7 Experiments

In our experiments we used the Ohsumed dataset taken from LETOR 3.0 and both datasets of LETOR 4.0[6]. We are only interested in datasets that contain more than 2 levels of relevance. On the one hand, this has a technical reason: calibration for binary relevance labels does not make too much sense. On the other hand, we believe that in this case the difference between various learning algorithms is more significant. All LETOR datasets we used contain 3 levels of relevance. We summarize their main statistics in Table 1.

[6] `http://research.microsoft.com/en-us/um/beijing/projects/letor/`

Table 1. The statistics of the datasets we used in our experiments.

	Number of documents	Number of queries	Number of features	Docs. per query
LETOR 3.0/Ohsumed	16140	106	45	152
LETOR 4.0/MQ2007	69623	1692	46	41
LETOR 4.0/MQ2008	15211	784	46	19

For each LETOR dataset there is a 5-fold train/valid/test split given. We used this split except that we divided the official train set by a random $80\%-20\%$ split into training and calibration sets which were used to adjust the parameters of the different calibration methods. We did not apply any feature engineering or preprocessing to the official feature set. The NDCG values we report in this section have been calculated using the provided evaluation tools.

We compared our algorithm to five state-of-the-art ranking methods whose outputs are available at the LETOR website for each dataset we used:

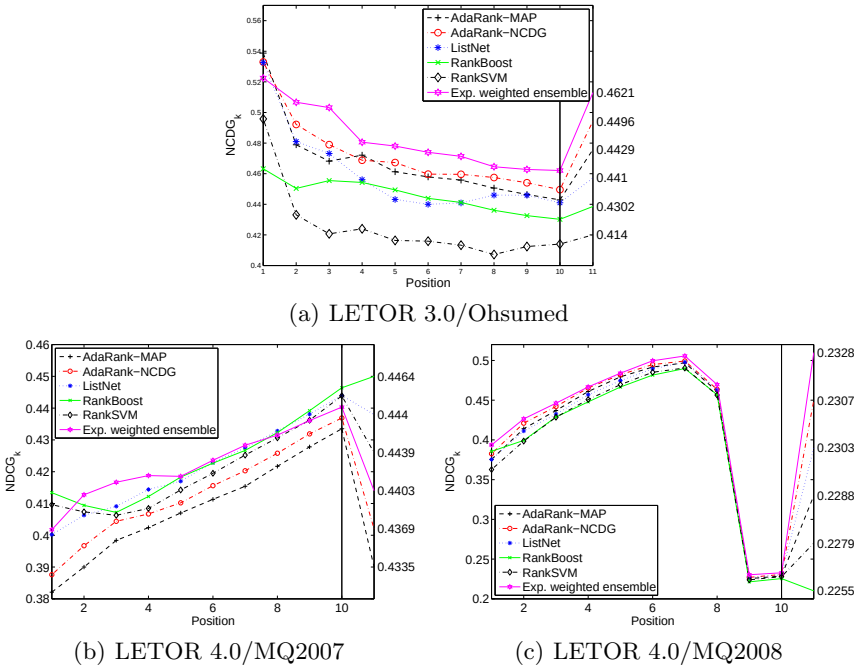
1. AdaRank-MAP [20]: a listwise boosting approach aiming to optimize MAP.
2. AdaRank-NDCG [20]: a listwise boosting approach with the NDCG as objective function, minimized by the AdaBoost mechanism.
3. ListNet [2]: a probabilistic listwise method which employs cross entropy loss as the listwise loss function in gradient descent.
4. RankBoost [8]: a pairwise approach which casts the ranking problem into a binary classification task. In each boosting iteration the weak classifier is chosen based on NDCG instead of error rate.
5. RankSVM [10] a pairwise method based on SVM, also based on binary classification as RankBoost.

To train AdaBoost.MH, we used our open source implementation available at `multiboost.org`. We did not validate the hyperparameter of weak learners. Instead, we calibrated and used all the trained models with various number of tree leafs and product terms. The number of tree leaves ranged from 5 to 45 uniformly with a step size of 5, and the number of product terms from 2 to 10. The training was performed on a grid[7] which allowed us to fully parallelize the training process, and thus it took less than one day to obtain all strong classifiers.

We only tuned the number of iterations T and the base parameter c in the exponential weighting scheme (6) on the validation set. In the exponential weighting combination (6) we set the weights using the NDCG_{10} performances of the calibrated models and c and T were selected based on the performance of $v^{\mathrm{posterior}}(\cdot)$ in terms of NDCG_{10}. The hyperparameter optimization was performed using a simple grid search where c ranged from 0 (corresponding to uniform weighting) to 200 and for T from 10 to 10000. Interestingly, the best number of iterations is very low compared to the ones reported by [11] for classification tasks. For LETOR 3.0 the best number of iterations is $T = 100$ and

[7] http://www.egi.eu

for both LETOR 4.0 datasets $T = 50$. The best base parameter is $c = 100$ for all databases. This value is relatively high considering that it is used in the exponent, but the performances of the best models were relatively close to each other. We used fixed parameters $C = 2$ in the TCN function L_C^{EWLS}, and $\sigma = 0.01$ in L_σ^{SN}.



(a) LETOR 3.0/Ohsumed

(b) LETOR 4.0/MQ2007

(c) LETOR 4.0/MQ2008

Fig. 2. NDCG_k values on the LETOR datasets. We blow up the NDCG_{10} values to see the differences.

7.1 Comparison to standard learning to rank methods

Figure 2 shows the NDCG_k values for different truncation levels k. Our approach consistently outperforms the baseline methods for almost every truncation level on the LETOR 3.0 and MQ2008 datasets. In the case of LETOR 3.0, our method is noticeably better for high truncation levels whereas it outperforms only slightly but consistently the baseline methods on MQ2008. The picture is not so clear for MQ2007 because our approach shows some improvement only for low truncation levels.[8]

[8] The official evaluation tool of LETOR datasets returns with zero NDCG_k value for a given query if k is bigger than the number of relevant documents. This results in poor performance of the ranking methods for $k \geq 9$ in the case of MQ2008. In our ensemble scheme, we used the NDCG_{10} values calculated according to (1) with a truncation level of 10.

Table 2. The NDCG values for various ranking algorithms. In the last three lines the results of our method are shown using only CPC, only RBC and both.

Method Database	Letor 3.0 Ohsumed	Letor 4.0 MQ2007	Letor 4.0 MQ2008
Eval. metric	$NDCG_{10}$	Avg. NDCG	Avg. NDCG
AdaRank-MAP	0.4429	0.4891	0.4915
AdaRank-NDCG	0.4496	0.4914	0.4950
ListNet	0.4410	0.4988	0.4914
RankBoost	0.4302	0.5003	0.4850
RankSVM	0.4140	0.4966	0.4832
Exp. w. ensemble with CPC	0.4621	0.4975	0.4998
Exp. w. ensemble with RBC	0.4493	0.4976	0.5004
Exp. w. ensemble with CPC+RBC	0.4561	0.4974	0.5006
AdaBoost.MH+Decision Tree	0.4164	0.4868	0.4843
AdaBoost.MH+Decision Product	0.4162	0.4785	0.4768

Table 2 shows the average NDCG values for LETOR 4.0 along with $NCDG_{10}$ for LETOR 3.0 (the tool for LETOR 3.0 does not output the average NDCG). We also calculate the performance of our approach where we put only RBC, only CPC, and both calibration ensembles into the pool of score functions used in the aggregation step (three rows in middle of the Table 2). Our approach consistently achieves the best performance among all methods.

We also evaluate the original AdaBoost.MH with decision tree and decision product, i.e. without our calibration and ensemble setup (last two rows in Table 2). The posteriors were calculated according to (3) and here we validated the i) iteration number and ii) the hyperparameters of base learners (the number of leaves and number of terms) in order to select a single best setting. Thus, these runs correspond to a standard classification approach using AdaBoost.MH. These results shows that our ensemble learning scheme where we calibrated the individual classifiers improves the standard AdaBoost.MH ranking setup significantly.

7.2 The diversity of CPC outputs

To investigate how diverse the score values of different class probability calibrated models are, we compared the scores obtained by the five CPC methods described in Section 4.2 using t-test. We obtained 5 p-values for each CPC pair. Then we applied Fischer's method to get one overall p-value assuming that these 5 p-values are coming form independent statistical tests. Here, we used the output of boosted trees only with the number of tree leaves set to 30.

The results in Figure 3 indicate that for a subset of TCFs, the estimated probability distributions were quite close to each other. Although the TCFs are rather different, it seems that they approximate a similar distribution with just small differences. We believe that one reason for the experienced efficiency of the proposed method is that these small differences within the cluster are due to the estimation noise, so by mixing them, the level of the noise decreases.

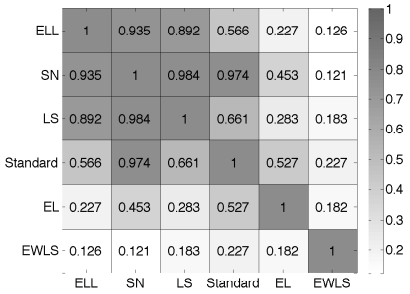


Fig. 3. The p-values for different calibrations obtained by Fischer's method on foldwise p-values of t-test, Letor 4.0/MQ2007. The calibration is calculated according to (3).

8 Conclusions

In this paper we presented a simple learning-to-rank approach based on multi-class classification, model calibration, and the aggregation of scoring functions. We showed that this approach is competitive with more complex methods such as RankSVM or ListNet on three benchmark datasets. We suggested the use of a sigmoid-based class probability calibration which is theoretically better grounded than regression based calibration, and thus we expected it to yield better results. Interestingly, this expectation was confirmed only for the Ohsumed dataset which is the most balanced set in terms of containing a relatively high number of highly relevant documents. This suggests that CPC has an advantage over RBC when all relevance levels are well represented in the data. Nevertheless, the CPC method was strongly competitive on the other two datasets as well, and it also has the advantage of coming with an upper bound on the NDCG measure.

Finally, we found AdaBoost to overfit the NDCG score for low number of iterations during the validation process. This fact indicates the presence of label noise in the learning-to-rank datasets, according to experiments conducted by [13] using artificial data. We note here that noise might come either real noise in the labeling, or from the deficiency of the overly simplistic feature representation which is unable to capture nontrivial semantics between a query and document. As a future work, we plan to investigate the robustness of our method to label noise using synthetic data since this is an important issue in a learning-to-rank application: while noise due to labeling might be reduced simply by improving the consistency of the data, it is less trivial to obtain significantly more complex feature representations. That said, the development of learning-to-rank approaches that are proven to be robust to label noise is of great practical importance.

9 Acknowledgments

This work was supported by the ANR-2010-COSI-002 grant of the French National Research Agency.

References

1. Busa-Fekete, R., Kégl, B., Éltető T., Szarvas, G.: Ranking by calibrated AdaBoost. In: (JMLR W&CP), vol. 14, pp. 37–48 (2011)
2. Cao, Z., Qin, T., Liu, T., Tsai, M., Li, H.: Learning to rank: from pairwise approach to listwise approach. In: Proceedings of the 24rd International Conference on Machine Learning, pp. 129–136 (2007)
3. Cesa-Bianchi, N., Lugosi, G.: Prediction, Learning, and Games. Cambridge University Press, New York, NY, USA (2006)
4. Chapelle, O., Chang, Y.: Yahoo! learning to rank challenge overview. In: Yahoo Learning to Rank Challenge (JMLR W&CP), vol. 14, pp. 1–24. Haifa, Israel (2010)
5. Chapelle, O., Metlzer, D., Zhang, Y., Grinspan, P.: Expected reciprocal rank for graded relevance. In: Proceeding of the 18th ACM conference on Information and knowledge management, pp. 621–630. ACM, New York, NY, USA (2009)
6. Chapelle, O., Wu, M.: Gradient descent optimization of smoothed information retrieval metrics. Information Retrievel **13**(3), 216–235 (2010)
7. Cossock, D., Zhang, T.: Statistical analysis of Bayes optimal subset ranking. IEEE Transactions on Information Theory **54**(11), 5140–5154 (2008)
8. Freund, Y., Iyer, R., Schapire, R.E., Singer, Y.: An efficient boosting algorithm for combining preferences. Journal of Machine Learning Research **4**, 933–969 (2003)
9. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences **55**, 119–139 (1997)
10. Herbrich, R., Graepel, T., Obermayer, K.: Large margin rank boundaries for ordinal regression. In: Smola, Bartlett, Schoelkopf, Schuurmans (eds.) Advances in Large Margin Classifiers, pp. 115–132. MIT Press, Cambridge, MA (2000)
11. Kégl, B., Busa-Fekete, R.: Boosting products of base classifiers. In: International Conference on Machine Learning, vol. 26, pp. 497–504. Montreal, Canada (2009)
12. Li, P., Burges, C., Wu, Q.: McRank: Learning to rank using multiple classification and gradient boosting. In: Advances in Neural Information Processing Systems, vol. 19, pp. 897–904. The MIT Press (2007)
13. Mease, D., Wyner, A.: Evidence contrary to the statistical view of boosting. Journal of Machine Learning Research **9**, 131–156 (2007)
14. Niculescu-Mizil, A., Caruana, R.: Obtaining calibrated probabilities from boosting. In: Proceedings of the 21st International Conference on Uncertainty in Artificial Intelligence, pp. 413–420 (2005)
15. Rissanen, J.: A universal prior for integers and estimation by minimum description length. Annals of Statistics **11**, 416–431 (1983)
16. Robertson, S., Zaragoza, H.: The probabilistic relevance framework: BM25 and beyond. Found. Trends Inf. Retr. **3**, 333–389 (2009)
17. Schapire, R.E., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. Machine Learning **37**(3), 297–336 (1999)
18. Valizadegan, H., Jin, R., Zhang, R., Mao, J.: Learning to rank by optimizing NDCG measure. In: Advances in Neural Information Processing Systems 22, pp. 1883–1891 (2009)
19. Wu, Q., Burges, C.J.C., Svore, K.M., Gao, J.: Adapting boosting for information retrieval measures. Inf. Retr. **13**(3), 254–270 (2010)
20. Xu, J., Li, H.: 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. ACM (2007)