Generalized BROOF-L2R: A General Framework for Learning to Rank Based on Boosting and Random Forests *

Clebson C. A. de Sá Marcos A. Gonçalves Daniel X. Sousa Thiago Salles
Federal University of Minas Gerais
Computer Science Department
Belo Horizonte, Brazil
{clebsonc, mgoncalv, danielxs, tsalles}@dcc.ufmg.br

ABSTRACT

The task of retrieving information that really matters to the users is considered hard when taking into consideration the current and increasingly amount of available information. To improve the effectiveness of this information seeking task, systems have relied on the combination of many predictors by means of machine learning methods, a task also known as learning to rank (L2R). The most effective learning methods for this task are based on ensembles of tress (e.g., Random Forests) and/or boosting techniques (e.g., RankBoost, MART, LambdaMART). In this paper, we propose a general framework that smoothly combines ensembles of additive trees, specifically Random Forests, with Boosting in a original way for the task of L2R. In particular, we exploit out-of-bag samples as well as a selective weight updating strategy (according to the out-of-bag samples) to effectively enhance the ranking performance. We instantiate such a general framework by considering different loss functions, different ways of weighting the weak learners as well as different types of weak learners. In our experiments our rankers were able to outperform all state-of-the-art baselines in all considered datasets, using just a small percentage of the original training set and faster convergence rates.

Keywords

Learning to Ranking; Random Forests; Boosting

1. INTRODUCTION

Today, we live in an era of massive available information, with a never-seen-before (and increasing) rate of information production. It is not surprising that such a scenario imposes hard to tackle challenges. For example, the availability of massive amounts of data is not of great help if one is not able to effectively access relevant information that satisfies her information needs. Retrieval systems, such as search engines, question and answer, and expert search systems serve exactly this purpose: given an information need, expressed in the form of a query, and a set of possible information units (e.g., documents), the main goal is to provide an ordered list of information units according to their relevance with relation to the query. The desideratum is to increase the likelihood of satisfying an user's information need in an effective manner, which translates to maintain the truly relevant results on top of the less relevant ones.

One of the key aspects that influence retrieval systems is how they determine the relative relevance among candidate results in order to produce a ranked list based on their relevance with regard to some information need, posed in the form of a query. The quality of those rankings is thus paramount to guarantee efficient and effective access to relevant information (and, hopefully, the satisfaction of the user's information needs). Several approaches to generate such ranked lists do exist, being traditionally performed by the specification of a function that is able to relate some user's query to the set of known (indexed) information units. Usually, ranking functions consider several features, such as those that rely on the relatedness between query and possible results (e.g., BM25, edit distance, similarities in vector space models) or on link analysis information (e.g., PageRank, HITS). Such features must be somehow combined to provide accurate relevance scores (and, thus, a properly ranked list of results).

Unfortunately, to specify and tune ranking functions turns out to be a major problem, specially when the number of features becomes large, with non-trivial interactions. This motivates the use of supervised machine learning techniques to devise such functions, since machine learning techniques are effective to combine multiple pieces of evidence towards optimizing some goal. This is the direction pursued by Learning to Rank (L2R) techniques, the primary focus of this work.

More specifically, based on a set of query-document pairs with known relevance judgments, the goal is to learn a function $f(d, q)$ that is able to accurately devise the relevance scores for a document d, with respect to a query q. Due to its importance, several approaches for L2R have been proposed in the literature. Ensemble methods, such as RankBoost [7], AdaRank [32] and Random Forests [1] (and the variations thereof, such as [11]), are deemed to be the techniques of choice for L2R, achieving higher effectiveness in published benchmarks when compared to other algorithms [11, 3]. Both RankBoost and AdaBoost are based on boosting [26], an

*This work was partially funded by projects InWeb (grant MCT/CNPq 573871/2008- 6) and MASWeb (grant FAPEMIG/PRONEX APQ-01400-14), and by the authors' individual grants from CNPq, FAPEMIG, Capes and Google Inc.

SIGIR '16, July 17–21, 2016, Pisa, Italy.

DOI: http://dx.doi.org/10.1145/2911451.2911540

iterative meta-algorithm that combines a series of weak-learners in order to come up with a strong final learner, focusing on hard-to-classify regions of the input space as the iterations go by. The strategies based on Random Forest rely on the combination of several decision trees, learned using bootstrapped samples of the training set, together with additional sources of randomization (such as random feature selection) to produce decorrelated-correlated trees—a requirement to guarantee its effectiveness.

In this work, we propose a general framework for L2R, named Generalized BROOF-L2R that explores the advantages of boosting and Random Forests, by combining them in a non-trivial fashion. More specifically, at each iteration of the boosting algorithm, a Random Forest model is learned, considering training examples sampled according to a probability distribution. Such probability distribution is updated at the end of each iteration, in order to force the subsequent learners to focus on hard to classify regions of the input space. In particular, the use of RF models as weak learners has its own advantages, since it is capable of providing robust estimates of expected error through out of bag error estimates and, by means of selectively updating the weights of out of bag samples, one can effectively slow down the tendency of boosting strategies to overfit (a well known phenomenon that becomes critical as the noise level of the dataset being analyzed increases).

As we shall detail in the next sections, the key aspects of the proposed Generalized BROOF-L2R have to do with how to update the probability distribution and how such update should be performed, as well as the underlying ranker to be used to produce the final set of results. In this work, we discuss a set of possible instances of the proposed general framework, in order to highlight the behavior and potential of the proposed L2R solution. In fact, the instances that makes use of out-of-bag samples and optimizes through gradient descent [12] over the residues is able to achieve the strongest results, in terms of Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG), with significant improvements over the explored adversary algorithms, considering 5 traditional benchmark datasets. Our alternative instances were also able to achieve competitive (or superior) results when compared to the baselines. Moreover, as our experimental evaluation shows, our approaches based on the proposed general framework are able to produce top-notch results with substantially less training samples when compared to the baselines. Such data efficiency is key to guarantee practical feasibility as obtaining labeled data is still a costly process.

To summarize, the contributions of this work are threefold. We provide a general framework for L2R that is able to combine two strong methods (boosting and Random Forests) in an original way, which can be specialized in several ways and produce highly effective L2R solutions. We propose and discuss a set of alternative instantiations of such a framework, in order to highlight the behavior and effectiveness of each possible choice. Finally, we advance the state of the art in L2R by means of some instantiations of our proposed framework that are able to outperform top-notch solutions, according to an extensive benchmark evaluation considering five datasets and seven L2R baseline algorithms.

Roadmap: Section 2 discusses related work. Section 3 presents our proposed Generalized BROOF-L2R framework, as well as outlines our proposed set of possible instantiations of the proposed framework. We clarify our experimental setup and discuss the obtained results in Section 4. Finally, Section 5 concludes and highlights some future work.

2. RELATED WORK

Learn to Rank (L2R) [17] is the focus of active developments due to its cross-industry and society importance. Here, we review some relevant work on this topic, positioning our work in the literature.

L2R attempts to improve traditional strategies for ranking query results according to some relevance criteria, by exploring supervised machine learning algorithms to combine various relevance related features into a more effective ranking function, based on a set of queries and associated documents with relevance judgments. L2R have been successfully applied to a variety of tasks, such as Question and Answer [28], Recommender [29, 16] and Document Retrieval [14] systems.

Solutions specifically tailored to improve document retrieval have been extensively studied in the past years [4, 19]. In general, there are three major L2R approaches: the pointwise, pairwise and listwise approaches. Pointwise L2R algorithms are probably the simplest (yet successful) approaches, directly translating the ranking problem to a classification/regression one. In this case, the training set for the supervised learning algorithm consists of pairs $\langle q_i, (x_{i,j}, y_{i,j})\rangle$ of queries q_i and a list of associated documents $x_{i,j}$, each one with its relevance judgment $y_{i,j}$. In this case, each triple $\langle q_i, x_{i,j}, y_{i,j}\rangle$ is considered to be a single training example. The goal is to learn a classifier/regressor model capable of accurately predicting the relevance score of a document x', with relation to a query q_i, thus producing a partial ordering over documents. Pairwise algorithms, on the other hand, transform the ranking problem into a pairwise classification/regression problem. In this case, learning algorithms are used to predict orders of document pairs, thus exploring more ground-truth information than the pointwise approaches. Unlike both mentioned strategies, the listwise approaches essentially treat $\langle q_i, (x_{i,j}, y_{i,j})_j\rangle$ as a single training instance (that is, considering a ranked list of documents for a query q_i as a single training example), capturing more information from the training set (namely, group structure) than the previous alternatives. Of course, being able to better capture training data information when learning a ranking function comes with a price: usually, pairwise and mainly listwise approaches are harder to train, since they require more sophisticated (e.g. query-level) loss functions [17].

In terms of the state-of-the-art in L2R, methods based on Random Forests (RFs) and boosting were shown to be strong solutions according to already published benchmarks [22, 11, 3]. More specifically, RFs (and the variations thereof [11]) as well as boosting algorithms such as Gradient Boosted Regression Trees (GBRT) [9] and LambdaMART [33], are considered by many [3, 22, 18] to be the state of the art in L2R tasks. This work is based on both RFs and boosting strategies. Thus, in the following we briefly review some previous literature on them.

The RF algorithm was proposed in [1] as a variation of bagging of low-correlated decision/regression trees, built with a series of random procedures, such as bootstrapping of training data and random attribute selection. The popularity of RFs is highlighted by their successful application in several domains, such as tag recommendation [3], object segmentation [27], human pose recognition [30] and L2R [3, 22],

to name a few. Thus, it is natural to expect several extensions to it, in order to improve its effectiveness even more. One such extension is the extremely randomized trees (ERT) model [10] and its application to L2R [11]. The ultimate goal of ERTs is to reduce the correlation between the trees composing the ensemble, a requirement to guarantee high effectiveness of RF models. This is achieved by modifying the RF algorithm in, essentially, two aspects: each tree is learned considering the entire training set, instead of bootstrapped samples. Furthermore, in order to determine the decision splits after the random attribute selection, instead of selecting a cut-point that optimizes node purity, ERTs simply select a random cut-point threshold. This ultimately reduces tree correlation, potentially improving generalization capability of the learned model. As a final remark, such RF based models can be regarded as nonlinear pointwise approaches for L2R.

Boosting strategies have also been shown to produce state of the art results on L2R tasks, with GBRT [9] (a.k.a, MART[1] (Multiple Additive Regression Trees) and Lambda-MART [33] as the two perhaps most widely used strategies. Both algorithms are additive ensembles of regression trees. GBRT learns a ranking function by approximating the root mean squared error (RMSE) on the training set through gradient descent. As with typical boosting algorithms, the goal of GBRT is to focus on regions of the input space where predicting the correct relevance score is a hard task. Since this algorithm aims at approximating the RMSE on the training data, it can be regarded as a pointwise approach. The Lambda-MART algorithm, on the other hand, is a listwise approach that directly optimizes the ranked list of documents according to some retrieval measure, such as NDCG (instead of simply approximating the RMSE of the training documents relevance scores in isolation). To this end, Lambda-MART learns a ranking function that generates a list of relevant documents to a query that is as close as possible to the correct rank. As GBRT, it is based on gradient descent to optimize such metric.

Due to the successful application of RFs and boosting in machine learning tasks (such as classification and L2R), some authors propose to use both strategies in order to come up with better learned models. For example, in [22] GBRTs and RFs are independently explored in order to learn better ranking functions. More specifically, the GBRT model is initialized with the residues of the RF algorithm, followed by the traditional iterations of a GBRT model. The main motivation behind this approach is that RFs are less prone to overfitting, being ideal to initialize the GBRT algorithm instead of the usual uniform initialization. According to the reported benchmark, such strategy was shown to be superior to the GBRT algorithm.

Unlike [22], in [21] the authors propose an enhanced RF model for classification by boosting the decision trees composing the ensemble. In this case, each tree is learned with training examples weighted by w_i, resembling boosting by re-weighting. In particular, training instances with higher weights influence more when determining the decision nodes (and cut-point threshold definition). Furthermore, each tree is evaluated according to this weighted training set, which enables the ensemble to focus on hard-to-predict regions. The observed effect of such combination is the ability to come up with high quality models with substantially reduced training sets. As we shall detail, our proposed framework is tailored for the L2R task and, instead of introducing boosting into random forests, we apply boosting to several RF models, which act as weak learners.

Differently from the aforementioned previous work, we base ourselves in a recent development for text classification, namely, the BROOF algorithm [25]. In BROOF algorithm, RF and boosting strategies are tightly coupled in order to exploit their unique advantages: by exploiting out of bag error estimates as well as selectively updating training weights according to out of bag samples, the BROOF model is able to focus on hard-to-classify regions of the input space, without being compromised by the boosting tendency to overfit. This ultimately leads to competitive results when compared to state of the art algorithms. In here, we generalize such approach specifically for L2R tasks in order to come up with better ranking functions: the Generalized BROOF-L2R. As we shall see, this general framework is flexible enough so that it can be instantiated in several ways, exploiting distinct characteristics of the ranking tasks being addressed. In special, with this general framework we are able to achieve state of the art results, with rankers superior to the top notch algorithms proposed so far in all evaluated cases.

3. GENERALIZED BROOF-L2R

In this section, we detail our proposed Generalized BROOF-L2R framework. Briefly speaking, this framework allows the definition of learners based on the combination of Random Forests and the Boosting meta-algorithm, in a non-trivial fashion. As we shall see, this framework establishes a set of operations to be performed during the boosting iterations, in a well defined order of application. The goal is to drive the weak learners towards hard to predict regions of the underlying data representation, in order to come up with an optimized additive combination of weak learners to form the final predictor. The extension points of the proposed framework can produce a heterogeneous set of instantiations that typically produces very competitive results for L2R. In the following, we present the generalized framework for L2R, as well as some pointwise instantiations. We stress that the set of instantiations discussed here is far from exhaustive, being possible to elaborate even better possibilities in future work.

3.1 Framework Description

Based on the BROOF algorithm, proposed in [25] to solve text classification tasks, we here extend the proposed ideas in order to exploit the combination of Random Forests and Boosting for the specific task of L2R. However, instead of directly adapting the original algorithm to a single L2R method, we here generalize it into an extensible framework that is flexible enough to permit a series of possible instantiations. The proposed framework, named Generalized BROOF-L2R is an additive model composed of several Random Forest models, which act as weak-learners. Each fitted model influences the final decision proportionally to its accuracy, focusing — as the boosting iterations go by — on ever more complex regions of the input space, in order to drive down the expected error. As usual in a boosting strategy, two aspects play a key role: (i) the influence β_t of each learner in the fitted additive model, and (ii) the strategy to update the sample distribution $w_{i,j}$ in each iteration t of the boosting meta-algorithm.

[1]From now on, we will use MART and GBRT as synonyms.

The basic structure of the framework is outlined in Algorithm 1, together with a brief explanation of what we call its extension points—the general functions exploited by the framework to determine how the optimization process works. There are 5 general functions whose purpose is to specify the weight distribution update process, the error estimation and the underlying input representation. Particularly, the use of the Random Forest classifier as a weak learner extends the range of possible instantiations of the framework, since it enables us to come up with better error rate estimates and a more selective approach to update the examples' weights, through the use of the so-called out-of-bag samples.

Let $Q_{trn} = \{(q_i, \{x_{i,j}, y_{i,j}\})|_{j=1}^{m_i}\}$ be the training set, descriptively the set of documents $x_{i,j}$, with associated graded relevance judgment $y_{i,j}$ with relation to a query q_i. Initially, associate a weight $w_{i,j}$ with each training example $x_{i,j}$ according to the general function InitializeWeights. For each boosting iteration t, the input data representation may be updated, through the general function UpdateExamples. This general function can considerably extend the range of possible implementations of the framework, allowing us for example, to instantiate a Gradient Boosting Machine algorithm [20, 12]. Then, a Random Forest regressor model RF_t is learned considering this data representation.

In order to evaluate the generalization capabilities of RF_t, predict $\hat{y}$ for a set of training documents given by ValidationSet. The output of this step is paramount to guide the optimization process towards hard to classify regions of the input space. Although being of great importance to boosting effectiveness, this focus on hard to classify regions of the input space may also be harmful to the optimization process, specially when dealing with noisy data. As noted by [8, 13], boosting tends to increase the weights of few hard-to-classify examples (e.g., noisy ones). Thus, the decision boundary may only be suitable for those noisy regions of the input space while not necessarily general enough for general examples. In order to offer a greater robustness against such a drawback, our framework exposes an intermediary step related to how the examples weights get updates as the boosting iterations go by. The general function ValidationSet serves the purpose of specifying which training examples should be used during error estimation and weights update. The main goal here is to provide some mechanism to slowdown overfitting as well as provide more robust estimates of error weight (to capture the generalization power of each weak learner and to determine how they should influence the final predictor).

The selected training examples are then used to compute both the error rate of the model and the influence β_t of the weak learner on the final model, through ComputeLearnerWeights. Finally, the training examples' weight distribution is updated by UpdateExampleWeights. This update process should, ideally, take into account the generalization capability of the current weak learner RF_t, as well as how hard is to correctly predict the ranked lists of the validation examples. Validation examples whose outcome is hard to predict by an accurate learner should influence more in the following boosting iterations. An early stopping strategy is adopted, terminating the boosting iterations if the current learner has an estimated error rate greater than 0.5. The final prediction rule is then given by an additive combination of the weak-learners RF_t, weighted by β_t.

Instantiation	Description	
	Extension Point	Variation
$\text{BROOF}_{\text{absolute}}$	InitializeWeights	**Uniform**
	UpdateExamples	**Identity**
	ValidationSet	**OOB**
	ComputeLearnerWeights	**Absolute**
	UpdateExampleWeights	**OOB**
$\text{BROOF}_{\text{median}}$	InitializeWeights	**Uniform**
	UpdateExamples	**Identity**
	ValidationSet	**OOB**
	ComputeLearnerWeights	**Median**
	UpdateExampleWeights	**OOB**
$\text{BROOF}_{\text{height}}$	InitializeWeights	**Uniform**
	UpdateExamples	**Identity**
	ValidationSet	**OOB**
	ComputeLearnerWeights	**Height**
	UpdateExampleWeights	**OOB**
$\text{BROOF}_{\text{gradient}}$	InitializeWeights	**Uniform**
	UpdateExamples	**Residue**
	ValidationSet	**OOB**
	ComputeLearnerWeights	**Constant**
	UpdateExampleWeights	**Constant**

Table 1: Generalized BROOF-L2R: Possible instantiations.

3.2 Possible Instantiations

In this section, we describe a set of possible instantiations of the proposed framework. Due to space limitations, we here focus on four possible instantiations, stressing that this is far from being an exhaustive list of possibilities. In fact, we consider some representative alternatives that highlight the flexibility of the proposed framework to produce L2R solutions that typically produces very competitive results.

In order to induce a L2R algorithm based on the Generalized BROOF-L2R framework, one needs to specify the 5 generic functions discussed earlier. Our proposed instantiations can be found in Table 1. In that table, we specify which alternative was chosen for each generic function, providing details on how they are implemented.

As it can be observed, $\text{BROOF}_{\text{absolute}}$, $\text{BROOF}_{\text{median}}$ and $\text{BROOF}_{\text{height}}$ rely on out-of-bag samples in order to drive the boosting meta-algorithm further on hard to predict regions of the input space. Such samples are explored when estimating the weak-learner's error rate through out-of-bag estimates. Recall that in boosting, the usual way of assessing the errors is to use the training to measure the error. This is too optimistic, since the same data that was used to train the model is used as a measure of error. By using the out-of-bag samples we are able to produce better error estimates, since the out-of-bag are an independent set of samples that was left apart during the construction of the model. Thus, it is able to better approximate the expected error rate of the learner and is a more reliable measure than the usual training error rate [1].

In addition, the out-of-bag errors estimates are used to identify the weights' distribution that should be applied on following iterations of the boosting procedure; allowing the model to focus on hard to predict regions of the input space. We hypothesize that such selective update strategy can slowdown the algorithm's tendency to overfit. The major difference between them relates on how each weak-learner influence on the final predictor. The proposed instantiations can be found outlined in Algorithms 2 to 4. More specifically, we considered the absolute regression loss, $|y_{i,j} - \hat{y}_{i,j}|$, computed for the out-of-bag samples. We call

Algorithm 1 Generalized BROOF-L2R: Pseudocode

function FIT($Q_{trn} = \{(q_i, \{x_{i,j}, y_{i,j}\}|_{j=1}^{m_i}\}$, max_iter=$M$, num_trees=$N$, shrinkage=$\eta$)
  $w_{i,j}$ =INITIALIZEWEIGHTS(Q_{trn})
  $x'_{i,j} \leftarrow y_{i,j}$
  for each $t = 1$ **to** M **do**
    $x'_{i,j} \leftarrow$UPDATEEXAMPLES($Q_{trn}, x'_{i,j}$)
    $RF_t \leftarrow RF_{Regressor}$.FIT($\{(x'_{i,j}, y_{i,j})\}, N$)
    $\{(\hat{y}^t_{i,j}, y^t_{i,j})\} \leftarrow$VALIDATIONSET($RF_t, Q_{trn}$)
    $\langle e^t_{i,j}, \beta_t \rangle \leftarrow$COMPUTELEARNERWEIGHTS($RF_t, \{(\hat{y}^t_{i,j}, y^t_{i,j})\}$)
    if $\sum_{i,j} e^t_{i,j} w_{i,j} \geq 0.5$ **then**
      break
    end if
    $w_{i,j} \leftarrow$UPDATEEXAMPLEWEIGHTS($e^t_{i,j}, \beta_t, \{(\hat{y}^t_{i,j}, y^t_{i,j})\}$)
  end for
  return $\{(RF_t, \beta_t)\}|_{t=1}^{M}$
end function

Function	Description
INITIALIZEWEIGHTS	Initial weights associated to each example, ressambling boosting by re-weighting. **Uniform:** Equal weights for each example, $w_{i,j} = \frac{1}{\sum_{i,j} m_{i,j}}$. **Random:** Randomly initialized weights, $w_{i,j}$ =RANDOM(), $0 \leq w_{i,j} \leq 1$.
UPDATEEXAMPLES	Determines the underlying representation of the input data, directly defining what the algorithm should optimize for. **Identity:** Maintains the original representation of input data, $x'_{i,j} = x_{i,j}$. **Residue:** Optimizes for the residues: $x'_{i,j} = \begin{cases} x'_{i,j} - \eta \hat{y}^{t-1}_{i,j} & \text{if } t > 1 \\ y_{i,j} & \text{otherwise} \end{cases}$, where η is a shrinkage factor.
VALIDATIONSET	Determines which training data will be considered during weight update and error rate estimation, with direct influence on the algorithm robustness to overfitting. **OOB:** The set of out of bag examples OOB_t related to RF_t. **Train:** The entire training set Q_{trn}.
COMPUTELEARNERWEIGHTS	Determines how to compute the influence of the current weak learner on the final predictor. **Absolute:** $\beta_t = \eta \frac{\epsilon}{1-\epsilon}$, where $\epsilon = \sum_{i,j} e^t_{i,j} w_{i,j}$, $e^t_{i,j} = \|y_{i,j} - \hat{y}_{i,j}\|$ and η is a shrinkage factor. **Median:** Similarly to the above variant, $\beta_t = \eta \frac{\epsilon}{1-\epsilon}$ and $\epsilon = \sum_{i,j} e^t_{i,j} w_{i,j}$. However, the errors are given by $e^t_{i,j} = \|\text{MEDIAN}(R_{\hat{y}_{i,j}}) - \hat{y}_{i,j}\|$ where R_i denotes the list of predictions $\hat{y}_{i,j}$ associated to documents whose real relevance score is i. **Height:** Similarly to the variants above, both $\beta_t = \eta \frac{\epsilon}{1-\epsilon}$ and $\epsilon = \sum_{i,j} e^t_{i,j} w_{i,j}$. Unlike them, $e^t_{i,j} = \begin{cases} \#\text{ irrelevant documents above } x_{i,j} & \text{if } x_{i,j} \text{is relevant} \\ \#\text{ relevant documents below } x_{i,j} & \text{otherwise} \end{cases}$, in the ordered list of results. **Constant:** Produces constant coefficients, $\beta_t = \eta$.
UPDATEEXAMPLEWEIGHTS	Specifies how to update the training examples weights to be used in the next iteration. **OOB:** Updates the weights associated to the out of bag samples according to β_t and the difficulty involved in predicting the samples' outcomes. More specifically, $w_{i,j} = w_{i,j} \beta_t^{1-e^t_{i,j}}$ **Train:** Updates the weights associated to the entire training set. Similarly to the above variant, the update strategy considers both the coefficient β_t and the error $e^t_{i,j}$. **Constant:** Keeps the same weights during the boosting iterations, $w_{i,j} = w_{i,j}$.

this variant $\text{BROOF}_{\text{absolute}}$. We also considered two other alternatives, that rely on the position of documents in the predicted ranked lists. One alternative, named $\text{BROOF-L2R}_{\text{median}}$, relies on the intuition that documents with the same relevance judgment should be as nearer as possible to each other on the current ranked list. We thus consider as loss $|\text{MEDIAN}(R_{\hat{y}_{i,j}}) - \hat{y}_{i,j}|$ where R_i denotes the list of predictions $\hat{y}_{i,j}$ associated to documents whose real relevance score is i. The second alternative, named $\text{BROOF}_{\text{height}}$, is inspired on ideas of [5]. We define the height of a document $x_{i,j}$ as the total number of irrelevant documents ranked higher then $x_{i,j}$ if $x_{i,j}$ is relevant, or the total number of relevant documents ranked below $x_{i,j}$ if it is an irrelevant one.

Finally, in order to illustrate the generality of our proposed framework, we provide a fourth instantiation, $\text{BROOF}_{\text{gradient}}$, that resembles the gradient boosting machines (GBM), that optimizes through gradient descent [12] over the residues. More specifically, by a suitable combination of alternative implementations for each general function outlined in Algorithm 1, one can come up with an algorithm that could be named Gradient Boosted Random Forests (GBRF). This is achieved by considering an alternative representation of input data, that optimizes for the residues, such as $y - \hat{y}$, instead of the original input representation, updating them according to the negative gradient of the cost function (in this case, RMSE). Such alternative is outlined in Algorithm 5.

As we shall see in our experimental evaluation (Section 5), our proposed instantiations achieve very strong results compared to seven state-of-the-art baselines algorithms in five representative datasets. In particular, $\text{BROOF}_{\text{absolute}}$ and $\text{BROOF}_{\text{gradient}}$ were shown to be the strongest algorithms, obtaining significant improvements over the best baselines.

4. EXPERIMENTAL EVALUATION

We conducted extensive experiments in well-known L2R benchmarks. In the following, we describe the characteristics of the used datasets, the baseline algorithms, the experimental protocol/setup and the experimental results.

4.1 Datasets

The corpus we use are freely available online for scientific purposes. Such datasets can be divided into two groups

Algorithm 2 BROOF$_{\text{absolute}}$: Pseudocode

function FIT($\{(q_i, \{x_{i,j}, y_{i,j}\}|_{j=1}^{m_i}\}$, M, N, η)
  $w_{i,j} = \frac{1}{\sum_{i,j} m_{i,j}}$
  $x'_{i,j} \leftarrow y_{i,j}$
  for each $t = 1$ **to** M **do**
    $x'_{i,j} \leftarrow x_{i,j}$
    $RF_t \leftarrow RF_{Regressor}$.FIT($\{(x'_{i,j}, y_{i,j})\}, N$)
    $\{(\hat{y}^t_{i,j}, y^t_{i,j})\} \leftarrow RF_t$.OOB
    $e^t_{i,j} \leftarrow |y_{i,j} - \hat{y}_{i,j}|$
    $\epsilon \leftarrow \sum_{i,j} e^t_{i,j} w_{i,j}$
    $\beta_t \leftarrow \eta \frac{\epsilon}{1-\epsilon}$
    if $\epsilon \geq 0.5$ **break**
    $w_{i,j} \leftarrow w_{i,j} \beta_t^{1-e^t_{i,j}}$
  end for
  return $\{(RF_t, \beta_t)\}|_{t=1}^{M}$
end function

Algorithm 3 BROOF$_{\text{median}}$: Pseudocode

function FIT($\{(q_i, \{x_{i,j}, y_{i,j}\}|_{j=1}^{m_i}\}$, M, N, η)
  $w_{i,j} = \frac{1}{\sum_{i,j} m_{i,j}}$
  $x'_{i,j} \leftarrow y_{i,j}$
  for each $t = 1$ **to** M **do**
    $x'_{i,j} \leftarrow x_{i,j}$
    $RF_t \leftarrow RF_{Regressor}$.FIT($\{(x'_{i,j}, y_{i,j})\}, N$)
    $\{(\hat{y}^t_{i,j}, y^t_{i,j})\} \leftarrow RF_t$.OOB
    $e^t_{i,j} \leftarrow$ MEDIAN($pos(y_{i,j}) - pos(\hat{y}_{i,j})$)
    $\epsilon \leftarrow \sum_{i,j} e^t_{i,j} w_{i,j}$
    $\beta_t \leftarrow \eta \frac{\epsilon}{1-\epsilon}$
    if $\epsilon \geq 0.5$ **break**
    $w_{i,j} \leftarrow w_{i,j} \beta_t^{1-e^t_{i,j}}$
  end for
  return $\{(RF_t, \beta_t)\}|_{t=1}^{M}$
end function

considering the relevance judgments and their sizes. The two largest datasets contain ⟨query, document⟩ pairs with five relevance levels, ranging from 0 (completely irrelevant) to 4 (highly relevant). In this group we have one dataset from the "YAHOO! Webscope Learning to Rank Challenge", divided into three partitions for training, validation and test. The second largest dataset, WEB10K, consists of 10,000 queries released by Microsoft. In contrast to the YAHOO! datasets, the Microsoft dataset is partitioned into 5 folds for cross-validation purposes, with 3 partitions used for training, 1 for validation and 1 for test.

The second group of datasets corresponds to well-known LETOR 3.0 Topic distillation tasks, TD2003 and TD2004 (a.k.a., informational queries), of the Web track of the Text Retrieval Conference 2003 and 2004. These datasets contain binary relevance judgments. Similarly to the WEB10K benchmark, these datasets are partitioned into 5 folds to be used in a folded cross-validation procedure.

For comparative purposes, considering that the Microsoft and LETOR datasets were designed for a folded cross-validation procedure, we applied this same strategy to the YAHOO! dataset by merging the original partitions into a single set, and splitting the sorted queries into 5 folds, distributed using the same proportions: 3 folds for training, 1 for validation and 1 for test. We report results for both splits: the original one (called YAHOOV1S2) and the new 5-fold split (called YAHOOV1S2-F5).

4.2 Baselines

In our experiments we consider as baselines freely available implementations of state-of-the-art L2R methods, including AdaRank (with MAP and NDCG as loss functions), Random Forests, SVM^{rank}, MART, LambdaMART and RankBoost. We used the RankLib[2] (under the Lemur project) implementations of RankBoost, MART and LambdaMART. For AdaRank we used the implementation freely available at Microsoft Research[3]. For SVM^{rank}, we used the original implementation of [15][4]. Finally, for Random Forests, we used the implementation available in Scikit-Learn[24] library, which is also the basis of our implementations.

Algorithm 4 BROOF-L2R$_{\text{height}}$: Pseudocode

function FIT($\{(q_i, \{x_{i,j}, y_{i,j}\}|_{j=1}^{m_i}\}$, M, N, η)
  $w_{i,j} = \frac{1}{\sum_{i,j} m_{i,j}}$
  $x'_{i,j} \leftarrow y_{i,j}$
  for each $t = 1$ **to** M **do**
    $x'_{i,j} \leftarrow x_{i,j}$
    $RF_t \leftarrow RF_{Regressor}$.FIT($\{(x'_{i,j}, y_{i,j})\}, N$)
    $\{(\hat{y}^t_{i,j}, y^t_{i,j})\} \leftarrow RF_t$.OOB
    $e^t_{i,j} \leftarrow \begin{cases} \#\text{ irrelevant documents above } x_{i,j} & \text{if } x_{i,j} \text{is relevant} \\ \#\text{ relevant documents below } x_{i,j} & \textit{otherwise} \end{cases}$
    $\epsilon \leftarrow \sum_{i,j} e^t_{i,j} w_{i,j}$
    $\beta_t \leftarrow \eta \frac{\epsilon}{1-\epsilon}$
    if $\epsilon \geq 0.5$ **break**
    $w_{i,j} \leftarrow w_{i,j} \beta_t^{1-e^t_{i,j}}$
  end for
  return $\{(RF_t, \beta_t)\}|_{t=1}^{M}$
end function

Algorithm 5 BROOF$_{\text{gradient}}$: Pseudocode

function FIT($\{(q_i, \{x_{i,j}, y_{i,j}\}|_{j=1}^{m_i}\}$, M, N, η)
  $w_{i,j} = \frac{1}{\sum_{i,j} m_{i,j}}$
  $x'_{i,j} \leftarrow y_{i,j}$
  for each $t = 1$ **to** M **do**
    $x'_{i,j} \leftarrow \begin{cases} x'_{i,j} - \eta \hat{y}^{t-1}_{i,j} & \text{if } t > 1 \\ y_{i,j} & \text{otherwise} \end{cases}$
    $RF_t \leftarrow RF_{Regressor}$.FIT($\{(x'_{i,j}, y_{i,j})\}, N$)
    $\{(\hat{y}^t_{i,j}, y^t_{i,j})\} \leftarrow RF_t$.OOB
    $e^t_{i,j} \leftarrow |y_{i,j} - \eta \hat{y}_{i,j}|$
    $\epsilon \leftarrow \sum_{i,j} e^t_{i,j} w_{i,j}$
    $\beta_t \leftarrow \eta$
    if $\epsilon \geq 0.5$ **break**
    $w_{i,j} \leftarrow w_{i,j}$
  end for
  return $\{(RF_t, \beta_t)\}|_{t=1}^{M}$
end function

4.3 Experimental Protocol and Setup

To validate the performance of our approaches, we use two statistical tests to assess the statistical significance of our results, namely, the Wilcoxon signed-rank test and the paired Student's t-test. We consider the Wilcoxon signed-rank test since it is a non-parametric statistical hypothesis testing procedure that requires no previous knowledge of the samples distribution. In fact, some authors believe that it is one of the best choices for the analysis of two independent samples [6]. However, there is also some discussion in the literature favoring the Student's t-test when comparing L2R methods [23]. Due to the lack of consensus, we perform our

[2]http://sourceforge.net/p/lemur/wiki/RankLib/
[3]http://research.microsoft.com/en-us/downloads/0eae7224-8c9b-4f1e-b515-515c71675d5c/
[4]https://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

analysis with both tests, considering a two-sided hypothesis with significance level of 0.95% in both tests.

The statistical tests are computed over the values for Mean Average Precision (MAP) and the Normalized Discounted Cumulative Gain at the top 10 retrieved documents (hereafter, NDCG@10), the two most important and frequently used performance metrics to evaluate a given permutation of a ranked list using binary and multi-relevance order [31]. To compute these metrics we used the standard evaluation tool available for the LETOR 3.0 benchmark (for binary datasets), as well the tool available for the Microsoft dataset for all multi-label relevance judgment datasets [5]. For MAP, let Q be the set of all queries. These tools simply compute

$$MAP = \sum_{q \in Q} \frac{\text{AVERAGEPRECISION}(q)}{|Q|}.$$

Regarding NDCG, we assume that NDCG@p is 0 (zero) for empty queries, i.e., queries with no relevant documents. Some of the available evaluations tools (e.g., the one from YAHOO!) assume the value of 1 for these cases, which may lead to higher values of NDCG [2]. We chose to standardize this issue, using the same criterion used by most evaluation tools, e.g., those available for the Letor (3.0 and 4.0) and Microsoft datasets, in order to allow fairer comparisons. Accordingly, let IDCG_p be the maximum possible discounted cumulative gain for a given query. These tools implement NDCG@p as follows:

$$NDCG@p = \frac{DCG_p}{IDCG_p}, \text{where } DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{log_2(i+1)}.$$

In terms of algorithm tuning, we follow the usual procedure of tuning the hyper-parameters using training and validation sets. Considering the Random Forest based approaches we vary the number of trees ranging from 10 to 1000. We achieved convergence around 300 trees, We also optimized the percentage of features to be considered as candidates during node splitting, as well as the maximum allowed number of leaf nodes. The optimal values were 0.3 and 100, respectively.

For $\text{BROOF}_{\text{absolute}}$, $\text{BROOF}_{\text{median}}$ and $\text{BROOF}_{\text{height}}$, we limited the number of iterations to 500, reminding that the algorithms have an early stopping criterion that prevents further boosting iterations when the error rate exceeds 0.5. On average, our strategies converge at about 15 iterations on the LETOR datasets, and around 5 to 10 iterations on the multi-relevance judgment datasets. An exception was $\text{BROOF}_{\text{gradient}}$ which converged at about 100 iterations for the largest datasets.

Concerning the SVM^{rank} baseline, we favored the use of a linear kernel considering the fact that we verified in our analysis that a polynomial kernel is infeasible on large scale benchmarks such as WEB10K. The cost parameter C was calibrated using the training and validation sets with the explored values: 0.001, 0.01, 0.1, 1, 10, 100 and 1000. For the boosting methods Mart and LambdaMART, we tuned, always considering the validation set, the number of iterations ranging from one to a hundred, with a step of 1, and then scaling it up to 1000 iterations, with steps of 100. For the shrinkage factor of the predictive models, we tested the values of 0.025, 0.05, 0.075 and 0.1. The best found values for the MART and LambdaMART were ensembles of 1000 trees with shrinkage factor η of 0.1. For the AdaRank_{MAP}, $\text{AdaRank}_{NDCG@5}$ and for the RankBoost algorithm, similar procedures were performed in the validation set to configure the number of iterations.

Finally, we performed 5, 10 and 30 runs of the 5-fold cross validation procedure for WEB10K, YAHOO! and LETOR datasets, respectively. The differences in the number of repetitions are due to the size of the datasets and the need to properly address the variance of the results. The reported results on Tables 2 and 3 are the average of all these runs, being the statistical tests applied to these results.

[5]Reminding that, at the time of the writing of this paper, the evaluation tool used in the YAHOO! competition was not available online.

4.4 Results

In this section we analyze our proposals in terms of effectiveness, comparing them to the 7 explored baseline algorithms on the 5 described datasets. The results are reported on Tables 2 and 3.

We start by considering the MAP metric (Table 2). Briefly, the MAP results show that, overall, our proposed framework outperforms or ties with the strongest baselines in all cases. More specifically, with the TD2003 dataset, $\text{BROOF}_{\text{height}}$ outperformed the strongest baseline (RF) considering both statistical tests, with $\text{BROOF}_{\text{absolute}}$ and $\text{BROOF}_{\text{median}}$ as the winners according to at least one statistical test. In this dataset, $\text{BROOF}_{\text{gradient}}$ was statistically tied with the best baseline. Considering TD2004, $\text{BROOF}_{\text{absolute}}$ was considered the top performer amongst the proposed solutions, being tied with the strongest baseline – RankBoost – in this dataset. Regarding the WEB10K dataset, we can see that $\text{BROOF}_{\text{gradient}}$ was the top performer, according to both statistical tests, being superior to MART, the strongest baseline. Finally, in the YAHOOV1S2 dataset all four proposed algorithms were statistically superior to the strongest baseline (RF) according to both statistical tests, whereas in the YAHOOV1S2-F5 dataset $\text{BROOF}_{\text{gradient}}$ was the best approach. In sum, according to the MAP metric, our results clearly show that the proposed instantiations of the Generalized BROOF framework produced very competitive results as the best algorithm, being superior in the majority of the cases (and tying in the others) – a very significant result.

Turning our attention to the NDCG results, reported on Table 3, a similar behavior can be observed: our proposed instantiations are no worse than the strongest baselines in all cases, being superior in the majority of cases. Considering the TD2003 and TD2004 datasets, our solutions were no worse than any baseline, being statistically tied with the strongest one (RF, in both cases). $\text{BROOF}_{\text{gradient}}$ was the best algorithm in the three remaining datasets, according to both employed statistical tests. Furthermore, $\text{BROOF}_{\text{median}}$ was also superior to the best baseline (MART) in the YAHOOV1S2 dataset (according to the Student's t-test), with $\text{BROOF}_{\text{absolute}}$ and $\text{BROOF}_{\text{height}}$ tied with the MART algorithm. Again, this set of results highlights the effectiveness of the proposed approaches.

We now turn our attention to some behavioral aspects of our algorithms, namely, convergence and learning efficiency. In order to better understand the convergence rate of our proposals, we provide an empirical evaluation of our most effective solution (i.e., $\text{BROOF}_{\text{gradient}}$), by analyzing the obtained NDCG@10 as we vary the number of boosting iterations, contrasting these results with the boosting

	Algorithm	Datasets				
		TD2003	TD2004	WEB10K	YAHOOV1S2	YAHOOV1S2-5F
Baselines	Mart	0.192633	0.193744	0.352491	0.559721	0.568821
	LambdaMart	0.165181	0.169605	0.350263	0.5545	0.563694
	RF	0.278644	0.2522	0.337702	0.563355	0.559019
	RankBoost	0.235189	0.255467	0.316201	0.544887	0.547524
	AdaRank-MAP	0.2003	0.196801	0.294792	0.413846	0.480190
	Adarank-NDCG	0.121672	0.132435	0.304359	0.540243	0.538514
	SVM-Rank	0.257490	0.220392	0.324552	0.544887	0.551333
BROOF	$\text{BROOF}_{\text{absolute}}$	0.288039_{+}	0.263288	0.342437	0.565486^{*}_{+}	0.563729
	$\text{BROOF}_{\text{median}}$	0.282427^{*}	0.259941	0.347665	0.567696^{*}_{+}	0.567284
	$\text{BROOF}_{\text{height}}$	0.285937^{*}_{+}	0.259058	0.340340	0.564774^{*}_{+}	0.557727
	$\text{BROOF}_{\text{gradient}}$	0.280634	0.252342	0.36251^{*}_{+}	0.572918^{*}_{+}	0.57656^{*}_{+}

'$*$': better than the strongest baseline, with statistical significance according to Wilcoxon Test
'$+$': better than the strongest baseline with statistical significance according to Student's t-test
'n': statistically tied results considering both tests

Table 2: Mean Average Precision (MAP): Obtained results.

	Algorithm	Datasets				
		TD2003	TD2004	WEB10K	YAHOOV1S2	YAHOOV1S2-5F
Baselines	Mart	0.271274	0.263926	0.4404	0.703757	0.714763
	LambdaMart	0.224536	0.237338	0.445437	0.69619	0.706287
	RF	0.36346	0.350582	0.424498	0.703139	0.702384
	RankBoost	0.31613	0.33399	0.397071	0.682478	0.681796
	AdaRank-MAP	0.271921	0.281035	0.35732	0.51767	0.607867
	AdaRank-NDCG	0.166241	0.182031	0.385761	0.66309	0.664115
	SVM-Rank	0.344177	0.303471	0.399902	0.682478	0.691064
BROOF	$\text{BROOF}_{\text{absolute}}$	0.360802	0.358146	0.434964	0.70633	0.706954
	$\text{BROOF}_{\text{median}}$	0.36798	0.350466	0.436284	0.708538_{+}	0.709148
	$\text{BROOF}_{\text{height}}$	0.368195	0.355356	0.42882	0.70383	0.701985
	$\text{BROOF}_{\text{gradient}}$	0.368695	0.348532	0.456081^{*}_{+}	0.717271^{*}_{+}	0.725129^{*}_{+}

'$*$': better than the strongest baseline, with statistical significance according to Wilcoxon Test
'$+$': better than the strongest baseline, with statistical significance according to Student's t-test
'n': statistically tied results considering both tests

Table 3: Normalized Discounted Cumulative Gain (NDCG@10): Obtained results.

baselines. We here focus on the three largest datasets: YAHOOV1S2, YAHOOV1S2-F5 and WEB10K. Results can be found on Figure 1. As it can be observed, $\text{BROOF}_{\text{gradient}}$ share similar behavior with three explored boosting algorithms, namely, MART, RankBoost and AdaRank-NDCG: the four algorithms show fast convergence rates. The two key differences are: (*i*) our approach is able to achieve significantly better results at the initial boosting iterations and (*ii*) $\text{BROOF}_{\text{gradient}}$ converges to a higher asymptote than the other algorithms. On the other hand, the convergence rate of LambdaMART was significantly slower than the convergence rate of the mentioned algorithms. In sum, $\text{BROOF}_{\text{gradient}}$ enjoys faster convergence rates, with higher NDCG values at the initial boosting iterations and higher asymptote. This is paramount to guarantee practical feasibility of our solution: although high effectiveness is a requirement, achieving such high effectiveness with just a few boosting iterations is key to minimize running time.

Another aspect of direct impact on the practical feasibility of the solutions is to what extent the algorithms are "data efficient". That is, to what extent each algorithm is capable of delivering highly effective rankings with reduced training sets. We evaluate the solutions under this dimension by analyzing each algorithm's learning curve. To this end, we measure the effectiveness of each algorithm as we vary training set size. We randomly sample s% examples from the training set, selected at random. We vary s from 10% to 100%, with steps of 10%. The obtained results can be found on Figure 2. Considering the WEB10K dataset, we can observe a surprising result: $\text{BROOF}_{\text{gradient}}$ is able to outperform all algorithms with just 20% of the training set, even considering the other algorithms trained with larger training sets (including the entire training set). Also, it can be noted that $\text{BROOF}_{\text{absolute}}$ is no worse than the baseline algorithms, even with 10% of the training set. In fact, with about 40% of the training set $\text{BROOF}_{\text{gradient}}$ is able to achieve its maximum effectiveness, whereas for $\text{BROOF}_{\text{absolute}}$ 10% is enough. For the YAHOO datasets, a similar behavior was observed: with about 20% to 30% of the training set our approaches were able to outperform the baseline algorithms (or match, in the case of $\text{BROOF}_{\text{absolute}}$), even considering the baseline algorithms trained with the entire training set. In these datasets, our algorithms were able to achieve maximum effectiveness at 50% to 80% of the training set. Considering the TD2003 and TD2004 datasets, the RF baseline was a bit more competitive to our approaches, exhibiting a similar behavior in terms of effectiveness as the training set size varies. In these datasets, 50% to 60% of the training set was enough to produce the best effectiveness on the TD2003, while 40% was enough to surpass all baselines on TD2004. These findings have also a direct influence on the practical feasibility of our solutions. First, smaller training sets translates to smaller runtimes. Second, obtaining labeled data is critical but also costly. Clearly, being able to produce highly effective models from reduced training sets is an important characteristic of a successful approach.

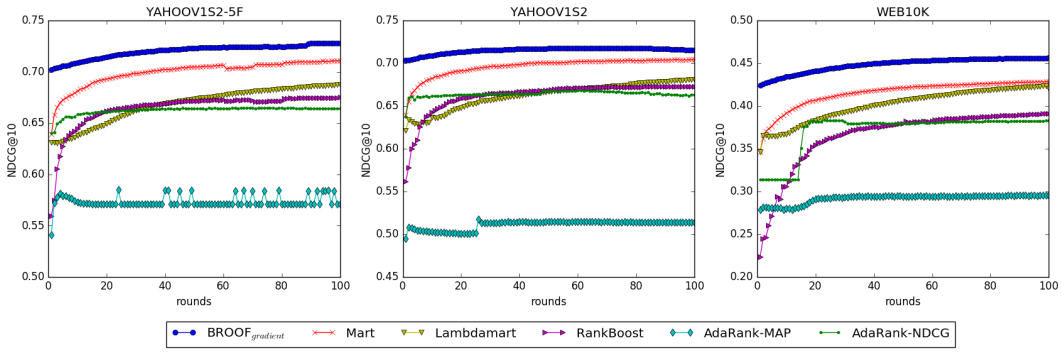

Figure 1: Convergence analysis: NDCG as the number of boosting iterations increases.

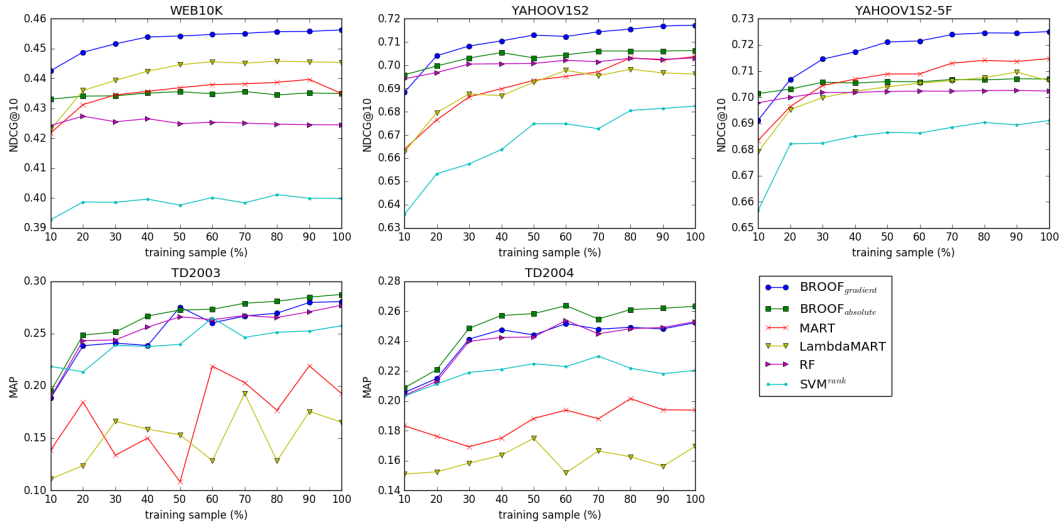

Figure 2: Learning curve analysis for the boosting algorithms.

Finally, we turn our attention to the effect of the use of out-of-bag samples by our approaches. Due to space restrictions, we here focus on BROOF$_{\text{gradient}}$, considering the WEB10K dataset. We analyze the effect of weak-learner error rate estimation through out-of-bag samples by contrasting it with a variant whose generic function ValidationSet equals to **Train**. The effectiveness of BROOF$_{\text{gradient}}$ and the mentioned variation, as the boosting iterations go by, can be found on Figure 3. From that figure, it is clear that the out-of-bag error estimation produces more effective results than the simple training error estimate. In fact, for all boosting iterations, the BROOF$_{\text{gradient}}$ variation with ValidationSet set to **OOB** produces better results than the results obtained with ValidationSet set to **Train**. This highlights the importance of exploiting the out-of-bag error estimates in our proposed framework instantiations. As a final remark, as it can be observed in Figure 3, even the variant that uses the training error rate is able to outperform the explored baselines. This is also an important aspect that highlights the quality of the proposed framework.

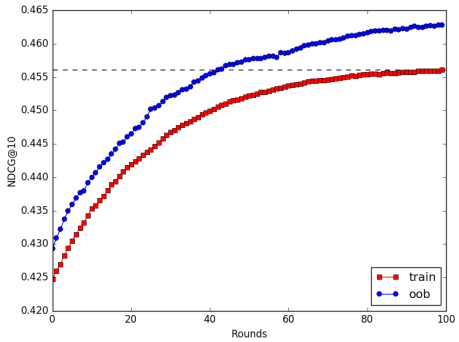

Figure 3: BROOF$_{\text{gradient}}$: Effect of out-of-bag samples versus entire training set.

5. CONCLUSIONS AND FUTURE WORK

In this work, we propose an extensible framework for L2R, called Generalized BROOF-L2R, which smoothly combines two successful strategies for Learning to Rank, namely, Random Forests and Boosting. Such combination, that uses Random Forests models as weak-learners for the boosting algorithm, relies on the use of the out of bag samples produced by the Random Forests to (i) determine the influence of each weak-learner in the final additive model and (ii) update the sample distribution weights by means of a more reliable error rate estimate. In fact, the framework is general enough to provide a rather heterogeneous set of instantiations that, according to our empirical evaluation, are able to

achieve competitive results compared to state-of-the-art algorithms for L2R. We proposed four different instantiations. Three instantiations closely follows the ideas of a recently proposed algorithm for text classification, namely, BROOF. The fourth instantiation is based on gradient descent optimization, resembling gradient boosting machines. In fact, such instantiation can be seen as a gradient boosted random forests model. As our results show, despite the fact that all the four algorithms provide very competitive results, two of them are consistently the top-performers, highlighting the quality and effectiveness of our proposed framework. Also, our proposals have two properties that are paramount to guarantee their practical feasibility, namely, data efficiency and fast convergence rates.

The space of possible instantiations of the proposed general framework for L2R is rather large. This clearly makes room for further investigations regarding such possibilities. In fact, one can come up with improved instantiations of the framework, by means of extending the set of possible implementations for each generic function composing the framework. This is under investigation. We also plan to study a more comprehensive set of instantiations, in order to build a substantially larger catalog of algorithms based on the Generalized BROOF-L2R to better understand the effects of each choice on model effectiveness.

References

[1] L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001.

[2] R. Busa-Fekete, B. Kégl, T. Éltető, and G. Szarvas. Tune and mix: learning to rank using ensembles of calibrated multi-class classifiers. *Machine Learning*, 93(2):261–292, 2013.

[3] S. D. Canuto, F. M. Belém, J. M. Almeida, and M. A. Gonçalves. A comparative study of learning-to-rank techniques for tag recommendation. *JIDM*, 4(3):453–468, 2013.

[4] O. Chapelle and Y. Chang. Yahoo! learning to rank challenge overview. *JMLR - Proceedings Track*, 14:1–24, 2011.

[5] K. Christakopoulou and A. Banerjee. Collaborative ranking with a push at the top. In *WWW*, pages 205–215, 2015.

[6] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, 2006.

[7] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969, 2003.

[8] Y. Freund and R. E. Schapire. Experiments with a New Boosting Algorithm. In *ICML*, pages 148–156, 1996.

[9] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[10] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Mach. Learn.*, 63(1):3–42, 2006.

[11] P. Geurts and G. Louppe. Learning to rank with extremely randomized trees. In *Proc. of the Yahoo! L2R Challenge, held at ICML 2010, Haifa, Israel, June 25, 2010*, volume 14 of *JMLR Proceedings Track*, pages 49–61, 2011.

[12] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, 2009.

[13] R. Jin, Y. Liu, L. Si, J. Carbonell, and A. G. Hauptmann. A new boosting algorithm using input-dependent regularizer. In *ICML*, 2003.

[14] T. Joachims. Optimizing search engines using clickthrough data. In *ACM SIGKDD*, pages 133–142, 2002.

[15] T. Joachims. Training linear svms in linear time. In *ACM SIGKDD*, pages 217–226, 2006.

[16] A. Karatzoglou, L. Baltrunas, and Y. Shi. Learning to rank for recommender systems. In *ACM RecSys*, pages 493–494, 2013.

[17] T.-Y. Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, 2009.

[18] C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, N. Tonellotto, and R. Venturini. Quickscorer: A fast algorithm to rank documents with additive ensembles of regression trees. In *SIGIR*, pages 73–82, 2015.

[19] C. Macdonald, R. L. Santos, and I. Ounis. The whens and hows of learning to rank for web search. *Inf. Retr.*, 16(5):584–628, 2013.

[20] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. In *In Advances in Neural Information Processing Systems*, pages 512–518, 2000.

[21] Y. Mishina, R. Murata, Y. Yamauchi, T. Yamashita, and H. Fujiyoshi. Boosted random forests. *IEICE Transactions*, 98-D(9):1630–1636, 2015.

[22] A. Mohan, Z. Chen, and K. Weinberger. Web-search ranking with initialized gradient boosted regression trees. *JMLR Workshop and Conference Proceedings: Proceedings of the Yahoo! Learning to Rank Challenge*, 14:77–89, 2011.

[23] L. a. F. Park. Confidence Intervals for Information Retrieval Evaluation. *Australiasian Document Computing Symposium*, 2010.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830, 2011.

[25] T. Salles, M. Gonçalves, V. Rodrigues, and L. Rocha. Broof: Exploiting out-of-bag errors, boosting and random forests for effective automated classification. In *SIGIR*, pages 353–362, 2015.

[26] R. E. Schapire and Y. Freund. *Boosting: Foundations and Algorithms*. 2012.

[27] F. Schroff, A. Criminisi, and A. Zisserman. Object class segmentation using random forests. In *British Machine Vision Conf.*, pages 1–10, 2008.

[28] A. Severyn and A. Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *ACM SIGIR*, pages 373–382, 2015.

[29] Y. Shi, M. Larson, and A. Hanjalic. List-wise learning to rank with matrix factorization for collaborative filtering. In *ACM RecSys*, pages 269–272, 2010.

[30] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *CVPR*, pages 1297–1304, 2011.

[31] N. Tax, S. Bockting, and D. Hiemstra. A cross-benchmark comparison of 87 learning to rank. *Information Processing & Management*, 51(6):757–772, 2015.

[32] J. Xu and H. Li. Adarank: A boosting algorithm for information retrieval. In *ACM SIGIR*, pages 391–398, 2007.

[33] Z. E. Xu, K. Q. Weinberger, and O. Chapelle. The greedy miser: Learning under test-time budgets. In *ICML*, 2012.