X-CLEAVER: **Learning Ranking Ensembles by Growing and Pruning Trees**

CLAUDIO LUCCHESE, Ca' Foscari University of Venice
FRANCO MARIA NARDINI, ISTI-CNR
SALVATORE ORLANDO, Ca' Foscari University of Venice
RAFFAELE PEREGO, ISTI-CNR
FABRIZIO SILVESTRI, Facebook Search
SALVATORE TRANI, ISTI-CNR

Learning-to-Rank (LtR) solutions are commonly used in large-scale information retrieval systems such as Web search engines, which have to return highly relevant documents in response to user query within fractions of seconds. The most effective LtR algorithms adopt a gradient boosting approach to build additive ensembles of weighted regression trees. Since the required ranking effectiveness is achieved with very large ensembles, the impact on response time and query throughput of these solutions is not negligible. In this paper we propose X-CLEAVER, an iterative meta-algorithm able to build more efficient and effective ranking ensembles. X-CLEAVER interleaves the iterations of a given gradient boosting learning algorithm with pruning and re-weighting phases. First, redundant trees are removed from the given ensemble, then the weights of the remaining trees are fine-tuned by optimizing the desired ranking quality metric. We propose and analyse several pruning strategies and we assess their benefits showing that interleaving pruning and re-weighting phases during learning is more effective than applying a single post-learning optimization step. Experiments conducted using two publicly available LtR datasets show that X-CLEAVER can be successfully exploited on top of several LtR algorithms as it is effective in optimizing the effectiveness of the learnt ensembles thus obtaining more compact forests that hence are much more efficient at scoring time.

This is the accepted authors' version.
Editor's version is avilable at https://doi.org/10.1145/3205453

CCS Concepts: • **Information systems** → **Learning to rank**; *Retrieval efficiency*; • **Computing methodologies** → **Learning to rank**; *Boosting*; *Regularization*;

Additional Key Words and Phrases: Learning to rank, Efficiency, Pruning.

This paper is supported by the MASTER (grant agreement Nº777695), BIGDATAGRAPES (grant agreement Nº780751), SOBIGDATA (grant agreement Nº654024) and ECOMOBILITY (project ID Nº10043082) projects that received funding from the European Union's Horizon 2020 research and innovation programme under the Marie-Slodowska Curie, Information and Communication Technologies, and Interreg Italy-Croatia CBC work programmes.
Author's addresses: F.M. Nardini, R. Perego, S. Trani are with the "Istituto di Scienza e Tecnologie dell'Informazione" (ISTI) of the National Research Council of Italy (CNR), Via Moruzzi 1, 56124 Pisa, Italy. e-mail: {firstname.lastname}@isti.cnr.it. C. Lucchese and S. Orlando are with the "University Ca' Foscari" of Venice, Italy. e-mail: claudio.lucchese@unive.it, orlando@unive.it. F. Silvestri is with "Facebook Search", London, UK. e-mail: fabrizio.silvestri@gmail.com. This work was done while Fabrizio Silvestri was an ISTI-CNR employee.

DOI: 10.1145/3205453

ACM Reference format:
Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, and Salvatore Trani. 2018. X-CLEAVER: Learning Ranking Ensembles by Growing and Pruning Trees. *ACM Trans. Intell. Syst. Technol.* 0, 0, Article 0 (2018), 26 pages.
DOI: 10.1145/3205453

1 INTRODUCTION

The problem of ranking items in response to a given query is of paramount importance for information retrieval systems. As exemplary case of how challenging that is, consider the processing of queries submitted to a Web Search Engine (WSE), where a small and relevant set of documents must be retrieved in a fraction of a second from a huge collection. The state of the art in ranking exploits supervised machine learning techniques based on Learning-to-Rank (LtR) algorithms [16]. Ranking models are in this case learnt from *ground truth* datasets composed of labeled training examples. The ranking function obtained assigns a relevance score to each candidate document with respect to the user query, where scores allow ranking and selecting the best documents that are eventually returned to the user.

While effectiveness of LtR methods has been always considered of primary importance, only recently efficiency of learnt models attracted the interest of the scientific community [1–3, 6, 19, 33]. Ranking models deployed in large-scale information retrieval systems must in fact feature very low latency as well as high throughput, due to the high rate of incoming user queries.

In this paper we tackle the problem of improving both efficiency and effectiveness of LtR models based on ensembles of additive regression trees, such as the ones learnt by gradient boosting algorithms, e.g., the state-of-the-art λ-MART [34]. We move from the simple observation that the cost of applying such ensemble models is linear in the number of their trees, and that ensembles composed of thousands of trees, despite being accurate, are very expensive when exploited for ranking large sets of candidate documents [20]. We thus propose a meta-algorithm, named X-CLEAVER (eXtended-CLEAVER) , which interleaves two novel steps of tree pruning and re-weighting within the usual iterative ensemble learning process: the pruning step aims at reducing the number of trees in the ensemble to improve its efficiency at scoring time, while tree re-weighting is an optimization process that aims to maximize the ranking quality of the pruned ensemble by tuning the weights associated with each tree. Any gradient boosting algorithm that produces a weighted ensemble of predictors can be used in conjunction with the X-CLEAVER meta-algorithm to grow the tree forest.

X-CLEAVER stems from and extends the CLEAVER algorithm [17], which applies similar optimizations *after* the completion of the learning phase to reduce the size of a given ensemble without affecting its quality. While CLEAVER is a single-pass algorithm applied once to the learnt ensemble, X-CLEAVER employs an iterative strategy where pruning and re-weighting optimizations are repeatedly applied during the learning process. Indeed, X-CLEAVER improves some of the pruning strategies proposed in [17] and, more importantly, it shows that embedding such steps within the boosting LtR algorithm is a profitable strategy to achieve more compact and effective ranking models. It is worth remarking that other approaches were proposed to produce simpler and faster tree-based ensembles [1, 33]. However, these proposals aimed at finding a trade-off between efficiency and effectiveness during the learning phase, while X-CLEAVER aims at improving ranking quality and decreasing scoring cost at the same time. This twofold opportunity is justified by two observations that we illustrate below.

The first observation is that regression/classification ensembles are known to encompass a high level of redundancy, and that some of their trees can be removed without harming significantly the

effectiveness of the resulting ensemble [15, 21]. We aim at designing efficient pruning strategies able to deal with large LtR ensembles and to remove a portion of their trees without affecting ranking accuracy. Pruning the ensemble obviously impacts the scores predicted by the ranking model. We thus propose to optimize the accuracy of the trees survived to the pruning step by fine-tuning their weights with an optimization heuristic driven by a ranking quality metrics.

A second observation makes this weight optimization step particularly interesting. It is related to the measure we use to evaluate the quality of a ranker, namely Normalized Discounted Cumulative Gain (*NDCG*) [13], and in particular *NDCG*@10. This quality measure is suitable in presence of documents associated with multiple levels of relevance, like the datasets commonly used in LtR experiments, where the judgments, labeling each candidate document for a given query, range from 0 (irrelevant) to 4 (perfectly relevant). Roughly speaking, *NDCG*@10 considers the ranks of the top-10 results, and penalizes highly relevant documents appearing lower in the result list, as the graded relevance value is reduced logarithmically proportional to the position of the result. Note that LtR gradient boosting algorithms either optimize the root mean squared error (RMSE) with respect to document relevance labels (e.g., the point-wise GBRT [10]), or optimize a proxy function of *NDCG*@10 (e.g., the list-wise λ-MART [34]). The λ-MART algorithm resort to this proxy since *NDCG*@10 is a non-differentiable function, and thus gradient descent cannot be applied. Specifically, λ-MART casts the ranking problem into a regression problem solved through gradient boosting that use the λ ranks as proxies of the gradient of *NDCG*@10 [7].

The proposed X-CLEAVER meta-algorithm is able to improve the effectiveness of the base LtR solutions adopted thanks to the optimization directly performed on the target objective function, e.g., *NDCG*@10, achieved by re-weighting the given ensemble predictors. This local optimization is especially profitable after the pruning step, which may potentially perturb the ensemble and drift the learning process away from the optimal solution.

By pushing the pruning and tree re-weighting phases within the learning process of the gradient boosting algorithm exploited, X-CLEAVER builds smaller models that outperforms the ones obtained by the original LtR algorithms in terms of both efficiency and effectiveness.

In summary, the main contributions of this work are the following:

- we propose X-CLEAVER, a novel LtR meta-algorithm aimed at optimizing state-of-the-art LtR algorithms generating forests of weighted regression trees by interleaving tree pruning and re-weighting within the iterative learning process;
- we propose several tree pruning strategies aimed at obtaining a faster ranking model (in terms of scoring time) and a heuristic optimization, driven by *NDCG*@10 (or any other ranking quality metric), for tuning tree weights. Experiments show that the proposed strategies allow us to prune up to 80% of trees in a λ-MART ensemble, without impacting its effectiveness;
- we conduct a comprehensive evaluation of X-CLEAVER on two publicly available datasets using different base gradient boosting algorithms. Experiments show that the models generated by X-CLEAVER remarkably outperform, in both ranking quality and scoring efficiency, the ones learnt by the respective base algorithms.

The paper is organized as follows. In Section 2 we discuss the research works that are related to LtR models optimization. In Section 3 we detail the proposed LtR algorithm showing the proposed pruning strategies and the greedy optimization process used to re-weight the trees in the ensemble. In Section 4 we first separately investigate the impact of the pruning and optimization processes, then we assess the performance of the novel X-CLEAVER algorithm as a whole. Finally, in Section 5 we draw the final conclusions and show some ideas we plan to investigate in the future.

2 RELATED WORK

Ranging from large-scale to company-wide search systems, LtR-based techniques are currently exploityed to improve the quality of the results delivered to search users. Herbrich *et al.* [12] presented the seminal work from which the area of LtR stemmed. Authors presented a new problem formulation, aiming at learning a ranking model by minimizing a loss function acting on pairs of elements, rather than on single labeled training instances as it is common in regression and classification tasks. Since then, a huge amount of works were published with the goal of improving the quality of the generated models. An excellent survey on models and techniques specifically tailored to Information Retrieval (and Search in particular) is that of Tie-Yan Liu [16].

Efficiency in LtR-based ranking models was investigated by following two different research directions. The first research line considered low-level optimizations to the inference phase by speeding-up the traversal of a given tree ensemble. Asadi *et al.* [2] proposed to re-arrange the code to traverse each tree of an ensemble, by transforming *control hazards* into less expensive *data hazards*, i.e., data dependencies introduced when one instruction requires the result of another. Moreover, to reduce *data hazards* the same work proposed to *vectorize* the scoring algorithm by interweaving the evaluation of a small set of candidate documents. Lucchese *et al.* [9, 18, 19] proposed QuickScorer, a scoring algorithm that adopts a new representation of the tree ensemble based on bit-vectors. The tree traversal, aimed to detect the leaves contributing to the final scoring of a document, is performed feature by feature, over the whole tree ensemble, through efficient logical bitwise operations.

The second research line concerning efficiency in LtR includes solutions that trade efficiency off for effectiveness as a post-learning phase. This direction was addressed mostly with model pruning and feature selection. Research in tree pruning dates back to 80's [26], and most of the recent contributions focus on pruning trees in an ensemble. One of the most influential work in this area is the one by Mehta *et al.* [22], where the minimum description length (MDL) principle was used to devise a novel tree pruning strategy. This strategy, instead of minimizing the length of the class sequence in the training sample together with the length of the decision tree, introduces a new length criterion to capture the intuitive idea of reducing the rate of misclassification.

Friedman and Popescu [11] proposed the Importance Sampled Learning Ensemble (ISLE) methodology. Given a set of weak learners, ISLE fits their weights by means of Lasso [32] regularized linear regression. Lasso is known to lead to sparse solutions, i.e., with few non-zero weights, thus allowing ISLE to prune trees. According to the authors, ISLE performs well with bagging models, but is not able to optimize boosting models (such as GBRT) which are at the base of state-of-the-art LtR solutions.

Two other relevant approaches were proposed by Margineantu *et al.* [21] and Li *et al.* [15]. In [15], weak learners are sorted on the basis of their classification confidence and then, after re-weighting, a subset is selected on the basis of the resulting classification accuracy. In [21], the best performing method, named *reduced-error pruning with backfitting*, grows a pruned set of weak classifiers by *i)* greedily adding a new element at each iteration, and, within each iteration, *ii)* replacing already selected elements with other elements until convergence. As both methods deal with classification ensembles, we cannot directly exploit them in an tr scenario. Moreover, since LtR training datasets employ hundreds of features and millions of documents, the exploitation of step *ii)* within an LtR scenario is not computationally feasible. In this work, we propose and evaluate a pruning strategy, named Quality-Loss, inspired by these two works, but targeting accuracy measured by a specific ranking metric. Instead of exploiting classification accuracy, ranking quality is used to sort the weak learners and to choose the less important to be pruned. Recursively, the accuracy of the remaining weak learners is re-evaluated and a new one is pruned.

Ren *et al.* [27] recently presented a work on pruning a random forest of decision trees. Authors proposed two techniques that exploit the global knowledge derived by having the whole model already trained and available. Global refinement jointly re-learns the leaf nodes of all trees under a global objective function, so that the complementary information between multiple trees is well exploited. Global pruning has the double goal of reducing the model size and, at the same time, of reducing the overfitting risk. Experiments conducted on real-world data showed that the model obtained by the pruning method has a smaller footprint and higher effectiveness. The main difference between this work and ours is that their main tasks are regression and classification, rather than ranking. In particular, our main goal is to improve the *NDCG* metrics, which is considerably different from minimizing other metrics such as those used in regression and classification.

Another recent development in ensembles pruning is the work of Qian *et al.* [25] where authors, unlike previous ones, address the trade-off between effectiveness (maximizing the generalization performance) and efficiency (minimizing the number of weak learners), by using a bi-objective problem formulation whose solution is found through an evolutionary Pareto optimization method combined with a local search step. Empirical results showed that this method is effective in reducing the size of the ensemble, and in increasing the effectiveness of the method. The goal of our method is different, as we aim at reducing the size of the ensemble up to an apriori fixed number of weak learners, while keeping the same quality level, thus resulting in an improved efficiency. In addition, our method is specifically tailored to LtR methods. From the same group of researchers, a Pareto optimization method was also used but with a different goal, i.e., to reduce the number of features used while keeping the quality as high as possible [24]. Again, as in the case of ensemble of trees, the goal was reached through optimizing two objectives at the same time.

Finally, Nan *et al.* [23] proposed to prune a random forest as a 0-1 integer program with linear constraints, which encourages feature re-use. They aim at reducing the size of a random forest, while also trying to reduce the number of variables used in the ensemble. It can be considered as a sort of pruning and feature selection mechanism. The method was empirically tested, and provided good performance improvements with respect to the state of the art. The main differences between this work and ours is that the method is not explicitly targeting ranking problems, and it is not clear if and how it could be extended to other ensemble methods, such as λ-MART or GBRT.

All these works, as our preliminary proposal in [17], share the common characteristic of being designed as post-processing solutions aimed at optimizing a fully learnt ensemble. Differently, X-CLEAVER learns, re-weights, and prunes trees in an interleaved manner.

A similar solution that adopts pruning during the model building phase is the one proposed by Asadi and Lin [1], where authors observed that compact, shallow, and balanced trees yield faster predictions. They thus suggested to incorporate a notion of execution cost during the training, to encourage trees with these topological characteristics. Authors proposed two strategies for accomplishing this: i) by directly modifying the node splitting criterion during tree induction, and ii) by stage-wise tree pruning. This approach is actually orthogonal to the one we are proposing in this paper, as it prunes each tree immediately after it is built and independently of the others, while we propose to prune subsets of trees of the ensemble. We use this strategy as a baseline to evaluate the X-CLEAVER ability in generating efficient models.

3 GROWING AND PRUNING TREE ENSEMBLES

Gradient boosting is an iterative technique used for regression or classification problems that learns from a training set an additive ensemble $\mathcal{E}$ of weighted *weak learners* minimizing a given loss function. In this work we focus on ensemble-based ranking models where the weak learners are decision trees. Large ensembles encompassing thousands of trees are usually required to achieve

the high ranking quality required by WSEs [6, 28]. Since the cost of scoring a document with an ensemble $\mathcal{E}$ is linear in its size $|\mathcal{E}|$, the ability of learning compact and effective ensembles is a very desirable property. Reducing document scoring time allows in fact to better cope with time budget constraints and improves the overall throughput of large search infrastructures [3, 8].

In the LtR scenario, the model $\mathcal{E}$ is learnt from a *ground truth* dataset containing query-document pairs (q, d) labeled with multi-graded relevance levels. The goal of an LtR algorithm is to learn a model able to produce a document ranking that agrees with the relevance ordering in the ground truth. Let $\mathcal{E} = \{(t_1, w_1), \ldots, (t_{|\mathcal{E}|}, w_{|\mathcal{E}|})\}$ be an additive ensemble of decision trees learnt by a gradient boosting LtR algorithm, where each tree t_i is weighted by w_i, $w_i \in \mathbb{R}$. Given a query q and a document d, we denote $s_i(q, d)$ the score contribution predicted by tree t_i. The global ensemble prediction $S(q, d)$ is computed as:

$$S(q, d) = \sum_{i=1}^{|\mathcal{E}|} w_i \cdot s_i(q, d) \tag{1}$$

Gradient boosting algorithms learn trees and weights iteratively. A new pair (t_i, w_i) is generated at iteration i to improve the predictions of the ensemble built so far. Indeed, each new tree t_i approximates the gradient of a loss function with respect to the model parameters. The exploitation of gradient boosting in the context of LtR is not trivial since the rank-based metrics adopted are either flat or discontinuous everywhere.

The proposed X-CLEAVER *meta-algorithm* can run on top of any *base* gradient boosting LtR algorithm $\mathcal{A}$ by providing a framework for the direct optimization of a given rank-based metric M. Specifically, X-CLEAVER interleaves three phases during the learning:

i) a *growing phase*, during which the base algorithm $\mathcal{A}$ is used to learn a set of additional trees, named *delta model*;
ii) a *pruning phase*, which removes the less relevant trees from the delta model, thus improving the *model efficiency*;
iii) a *re-weighting phase*, during which the weights associated with the remaining trees of the delta model are fine-tuned, by directly optimizing rank-based metric M thus improving the *model effectiveness*.

These three phases are iterated until the desired final ensemble size is obtained.

Without loss of generality, we use *NDCG* as the reference quality measure to be maximized. Specifically, we use *NDCG*@10 as commonly done for a Web search scenario. Given a ranked document list, *NDCG*@10 is a normalized measure that only weights the top-10 ranked documents according to their predicted relevance $S(q, d)$ and discounts their contribution according to their rank position. A basic approach to maximize *NDCG* is that of GBRT, where gradient boosting is driven by the Root Mean Squared Error (RMSE) loss, i.e., the root mean squared loss between the ensemble predictions and the relevance levels. Unfortunately, unless we have perfect predictions, minimizing RMSE does not provide any guarantee of optimizing *NDCG*. The λ-MART algorithm, which is considered the state-of-the-art in LtR, exploits approximations of the gradients of the target rank-based metric, named λ-ranks. At each iteration i, λ-MART trains a new tree t_i so as to best approximate such λ-ranks in place of the non-differentiable rank-based metric. Thus, λ-MART employs a proxy loss function, which may generate sub-optimal ranking choices.

In this work, we show how gradient boosting LtR algorithms, e.g., GBRT and λ-MART, can be used as base algorithms of X-CLEAVER, to build ranking models achieving a significant advantage in effectiveness and efficiency.

Algorithm 1 X-CLEAVER.

1: **function** X-CLEAVER($N, n, p, \mathcal{A}, M$)
  input:
2: N : *ensemble size*
3: n : *number of regression trees trained per iteration*
4: p : *ratio of pruned trees per iteration*
5: $\mathcal{A}$: *base learning algorithm*
6: M : *rank-based metric*
  output:
7: $\mathcal{E}$: *trained ensemble*
8: $\mathcal{E} \leftarrow \emptyset$ ▷ the current ensemble model
9: **while** $|\mathcal{E}| < N$ **do**
10: $\widehat{\mathcal{E}} \leftarrow \mathcal{A}$.GROWMODEL($\mathcal{E}, n$) ▷ build delta model
11: $\widehat{\mathcal{E}}_P \leftarrow$ PRUNE($\widehat{\mathcal{E}}, M, p$)
12: $\widehat{\mathcal{E}}_W \leftarrow$ REWEIGHT($\widehat{\mathcal{E}}_P, M$)
13: **if** $M(\mathcal{E} \cup \widehat{\mathcal{E}}_W) \leq M(\mathcal{E})$ **then** ▷ no gain condition
14: **break**
15: $\mathcal{E} \leftarrow \mathcal{E} \cup \widehat{\mathcal{E}}_W$
16: **return** $\mathcal{E}$

3.1 The X-CLEAVER Algorithm

The pseudocode of X-CLEAVER is illustrated in Algorithm 1. During its first phase, X-CLEAVER exploits the base algorithm $\mathcal{A}$ to *grow* $\mathcal{E}$, the (initially empty) current model (line 10). At each iteration the base algorithm $\mathcal{A}$ is used to train, on the basis of the current model $\mathcal{E}$, a new set of n weak learners, denoted with $\widehat{\mathcal{E}}$.

Before adding $\widehat{\mathcal{E}}$ to the current model $\mathcal{E}$, X-CLEAVER applies the *pruning* and *re-weighting* phases. During the pruning phase (line 11), a fraction p of trees in $\widehat{\mathcal{E}}$ is removed. Pruning aims to reduce the scoring time, by shrinking the ensemble size. It exploits the fact that some weak learners are highly correlated, and therefore they can be removed from the model without impacting significantly its accuracy. In Section 3.2 we explore different pruning strategies aimed at identifying the set of trees that less impact the overall model prediction power. We denote by $\widehat{\mathcal{E}}_P$ the set of weak learners in $\widehat{\mathcal{E}}$ that survive after the application of the pruning strategy.

As the pruning phase impacts predictions, X-CLEAVER *re-weights* the trees survived in $\widehat{\mathcal{E}}_P$ (line 12). To this end, it exploits a local optimization strategy based on line search. As discussed in Section 3.3, the weights associated with the weak learners of $\widehat{\mathcal{E}}_P$ are fine-tuned by directly optimizing the given rank-based metric M. The final pruned and re-weighted *delta model* is denoted by $\widehat{\mathcal{E}}_W$. If $\widehat{\mathcal{E}}_W$ is not able to improve the quality of the current model $\mathcal{E}$ as measured by M, then X-CLEAVER terminates (line 13). Otherwise, $\widehat{\mathcal{E}}_W$ is added to $\mathcal{E}$ and the above three phases iterated until the desired ensemble size N is achieved.

3.2 Pruning Phase

The pruning phase of X-CLEAVER identifies a fraction p of trees to be removed from delta model $\widehat{\mathcal{E}}$. Note that similar pruning strategies were preliminarly investigated in CLEAVER [17]. We improve some of them as discussed below.

- LAST: this trivial strategy prunes the last trees added to $\widehat{\mathcal{E}}$. The motivation is that trees are progressively built and added to the ensemble to refine the quality of the overall extracted model. The last trees are thus expected to provide a smaller contribution.
- RANDOM: this technique prunes uniformly at random a fraction p of trees from $\widehat{\mathcal{E}}$. The best set out of r different pruning rounds is selected, where quality is measured according to metric M. Note that CLEAVER adopted a single pruning round, while X-CLEAVER exploits M in choosing the best random-generated subset, thus providing more stable and reliable results. In our tests, we set $r = 100$.
- SKIP: this strategy removes trees that are uniformly scattered along the sequence of trees in $\widehat{\mathcal{E}}$. One tree every $1/p$ is removed, i.e., trees at position $\lceil i/p \rceil$ with $i = 1, \ldots, \lceil np \rceil$. The rationale is that trees learnt in close iterations are similar, and potentially redundant.
- LOW-WEIGHTS: this strategy removes from $\widehat{\mathcal{E}}$ the fraction p of trees with the lowest weights w_i. The assumption here is that, due to the low weights, they are less relevant for the final document scoring. For those learning algorithms $\mathcal{A}$ that associate uniform weights with all the trees, a preliminary *re-weigthing phase* is applied to $\widehat{\mathcal{E}}$ (see Sec. 3.3) in order to obtain the weights to which this pruning strategy is applied.
- QUALITY-LOSS: this strategy considers the actual impact of a tree in $\widehat{\mathcal{E}}$ to the optimization of rank-based metric M. The impact of a tree is measured as the loss variation of the metric M caused by its removal. The tree with the smallest impact is removed, and the impact of the remaining trees in the pruned ensemble is recomputed before pruning the next tree. The procedure is repeated until a fraction p of trees is removed from $\widehat{\mathcal{E}}$. Although the greedy choice of which tree to prune is locally optimal, there is no guarantee about the global optimality of the pruned ensemble $\widehat{\mathcal{E}}_P$. A simplified version of the same strategy, presented in CLEAVER [17], measures the trees' impact only once, and then selects the trees to discard according to this impact-based ranking. Despite being less demanding in terms of computational complexity, it does not take into account trees' dependencies, i.e., two highly redundant trees risk to be kept both. This enhanced pruning strategy achieves a significant improvement over the basic variant in [17].
- SCORE-LOSS: this strategy considers the normalized contribution provided by each tree to the final score $S(q, d)$. The contribution is measured as

$$\frac{w_i \cdot s_i(q, d)}{S(q, d)}$$

  and it is averaged over all query-document pairs of the training dataset. Then, the fraction p of trees in $\widehat{\mathcal{E}}$ with the lowest average contribution is pruned.

Eventually, the *pruning phase* produces the pruned ensemble $\widehat{\mathcal{E}}_P \subseteq \widehat{\mathcal{E}}$ (line 11).

3.3 Re-weighting phase

The re-weighting phase of X-CLEAVER fine-tunes the pruned model $\widehat{\mathcal{E}}_P$ in order to create a new ensemble $\widehat{\mathcal{E}}_W$ having the same trees of $\widehat{\mathcal{E}}_P$ and a new set of weights such that $M(\mathcal{E} \cup \widehat{\mathcal{E}}_W)$ is maximized.

Let $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_{|\widehat{\mathcal{E}}_W|}\}$ be the weights vector of $\widehat{\mathcal{E}}_W$. Our goal is to solve the following optimization problem:

$$\overline{\Gamma} = \arg\max_{\Gamma \in \mathbb{R}^{|\widehat{\mathcal{E}}_W|}} M(\mathcal{E} \cup \widehat{\mathcal{E}}_W).$$

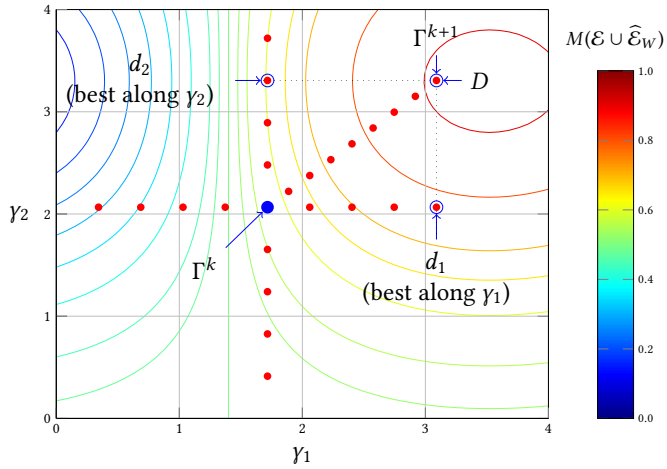

Fig. 1. Example of the Line Search strategy on a 2D search space.

Finding the optimal $\overline{\Gamma}$ is not feasible because i) M is not differentiable, and ii) Γ can have a high number of dimensions. However, local-search heuristics have proven to effectively address such optimization problem [30, 31]. Instead of relying on an approximation of the gradient of a target non-differentiable ranking metric, they evaluate the actual ranking measure after small modifications of current weight values. In [30], each weight is optimized independently of the others by computing the variation in the ranking measure occurring at the *jumping points*, that are the weight's values affecting the documents ranking. This process is applied to each of the weights and iterated until convergence. Analogously to [31], we adopt a less computationally expensive approach that does not require to compute all the possible *jumping points*. In particular, X-CLEAVER exploits an iterative two-step procedure based on *line search* as follows.

Let Γ^k be the vector of weights found at iteration k of our line search procedure. During the first step, a descent direction originating in Γ^k is identified. Then, in the second step, a new weight vector Γ^{k+1} that improves the loss function L is searched along such direction. Starting from Γ^0 being the weights of $\widehat{\mathcal{E}}_P$, the two steps detailed below are iterated until L, measured on a separate validation set, does not change for a fixed number of iterations:

(1) Given the solution Γ^k at iteration k, a line search along each axis of the weight vector is performed independently. For each dimension, the weight γ_i is replaced with a candidate weight computed by testing σ equi-spaced samples in the interval $[\gamma_i - \omega, \gamma_i + \omega]$, while keeping all other γ_j fixed. The parameters σ and ω affect the granularity of the local search. The value of ω is also reduced by a shrinking factor η at each iteration to favor a fine-grained optimization when the algorithm approaches a local maximum. The best among the σ samples, denoted by d_i, is selected by evaluating $M(\mathcal{E} \cup \widehat{\mathcal{E}}_W)$ with the modified set of weights. Eventually, the independent line searches lead to a new point $D = \{d_1, d_2, \ldots, d_{|\widehat{\mathcal{E}}_W|}\}$. Fig. 1 exemplifies the line search algorithm on a two-dimensional search space: along each axis, $\sigma = 9$ samples (horizontal and vertical red dots) are evaluated independently in order to choose the best weight updates (circled in blue) along the directions γ_0 and γ_1.

(2) An additional line search is conducted along the promising descent direction identified at the previous step, i.e., along the segment connecting Γ^k to D. Again σ equi-spaced samples are evaluated and the best is chosen so as to maximize $M(\mathcal{E} \cup \widehat{\mathcal{E}}_W)$. The best weight vector

Table 1. Properties of the MSLR-WEB30K-F1 and Istella-S datasets.

Dataset	MSLR-WEB30K-F1	Istella-S
# queries	31,351	33,018
# query-doc pairs	3,771,125	3,408,630
# features	136	220
avg. # docs/query	120	103
# pos. query-doc pairs	1,830,173 (48.53%)	388,224 (11.39%)
# neg. query-doc pairs	1,940,952 (51.47%)	3,020,406 (88.61%)

found, denoted by Γ^{k+1}, is used as the starting point for the next iteration. In Fig. 1, the best among the σ samples is exactly point D.

To reduce the overall computational cost of the above search process, we exploit the following optimizations. First, thread-level parallelism is used in order to explore the different search directions during the first step, i.e., to find the various d_i, and to evaluate the σ samples during the second step. Second, we avoid visiting the whole tree ensemble for scoring documents after each weight update. The ensemble of trees is in fact visited only once and tree predictions $s_i(q, d)$ for all the documents of the training dataset are stored in memory, thus allowing a fast access to these predictions when the single weight γ_i is updated.

4 EXPERIMENTAL EVALUATION

We conduct experiments by employing two public LtR datasets: i) the MSLR-WEB30K-F1[1] (Fold 1) dataset and ii) the Istella-S[2] dataset [17]. The MSLR-WEB30K-F1 dataset contains 3,771,125 query-document pairs referred to 31,351 queries, while the Istella-S dataset includes 33,018 queries and 3,408,630 query-document pairs. Query-document pairs are represented in the two datasets with 136 and 220 features, respectively. The characteristics of the datasets are listed in Table 1.

The query-document pairs in both datasets are labeled by relevance judgments that are natural numbers ranging from 0 (irrelevant) to 4 (perfectly relevant). While the size of the two datasets is comparable, they show a different proportion between positive (label > 0) and negative (label $= 0$) examples. Indeed, the Istella-S dataset contains a lower number of positive query-doc pairs than MSLR-WEB30K-F1 (11.39% versus 48.53% of the examples). Moreover, as reported in Table 2, the distributions of positive labels in the two datasets are different with MSLR-WEB30K-F1 showing a skewed distribution that is not observable in Istella-S. MSLR-WEB30K-F1 comes with a number of low relevance labels of 66.98% of the total set of positive examples. In the Istella-S dataset the low relevance class accounts instead for only 21.42%, while the medium and the high relevance classes account for 35.03% and 24.20%, respectively. MSLR-WEB30K-F1 shows a small number of perfectly relevant query-document pairs (i.e., 1.66% of the total), while in the Istella-S dataset the same class accounts for 19.35%. This differences are probably due to the diverse annotation processes followed in the respective companies.

We split each dataset in training, validation and test sets according to a 60%-20%-20% schema. Then, we use training and validation datasets to train models according to several state-of-the-art LtR learning algorithms, namely Gradient Boosting Regression Trees (GBRT) [10], Lambda-MART (λ-MART) [34] and Oblivious Lambda-Mart (Ω_λ-MART) [28]. GBRT is a *point-wise* algorithm that uses Mean Squared Error (MSE) as loss function to learn a predictor of the relevance labels.

[1] http://research.microsoft.com/en-us/projects/mslr/
[2] http://blog.istella.it/istella-learning-to-rank-dataset/

Table 2. Distribution of positive labels in the MSLR-WEB30K-F1 and Istella-S datasets.

# pos. query-doc pairs per label	MSLR-WEB30K-F1	Istella-S
# 1s (low relevance)	1,225,770 (66.98%)	83,167 (21.42%)
# 2s (medium relevance)	504,958 (27.59%)	135,989 (35.03%)
# 3s (high relevance)	69,010 (3.77%)	93,957 (24.20%)
# 4s (perfect relevance)	30,435 (1.66%)	75,111 (19.35%)
Total # pos. examples	1,830,173	388,224

λ-MART is a *list-wise* LtR algorithm using *NDCG* in its loss function and trying to learn the ranking function induced by the examples in the training set. Ω_λ-MART is a variation of λ-MART where oblivious regression trees [14] are learnt in place of standard, possibly unbalanced, regression trees. In oblivious regression trees the same splitting criterion, i.e., the same test "feature f_i is less than threshold t_j", is used for all the nodes lying on the same level of a tree. As a consequence, the resulting trees are balanced. The goal of reducing the degrees of freedom of the algorithm when growing trees at training time is twofold: first, to reduce the risk of overfitting; second, to allow for an improved efficiency at scoring time, since less tests have to be made when traversing each tree and most importantly they can also be easily evaluated in parallel, disregarding the path of each instance. λ-MART and Ω_λ-MART resulted to be the most effective LtR algorithms among those participating to the Yahoo! Learning to Rank Challenge [4]. We exploit the open-source implementation of these algorithms provided by QuickRank [5]. We fine-tune the hyper parameters of these algorithms by sweeping them to maximize *NDCG*@10 (tests conducted using different cut-off values did not show appreciable differences).

We vary the maximum number of leaves in each tree for GBRT and λ-MART in the set $\{5, 10, 25, 50\}$, while for Ω_λ-MART we vary the maximum depth of each tree in the set $\{4, 5, 6\}$. Moreover, we vary the learning rate in $\{0.05, 0.1, 0.5, 1.0\}$. To avoid overfitting, we allow each algorithm to train up to 1,500 trees unless there is no improvement in *NDCG*@10 on the validation set during the last 100 iterations. For GBRT and λ-MART we obtained the best results in terms of *NDCG*@10 for both datasets using trees with a maximum number of 50 leaves, and a learning rate equal to 0.05 (the same settings were reported as optimal in [6]), while for Ω_λ-MART we obtained the best results by using trees with a maximum depth of 5 and a learning rate of 0, 1.

Different settings for the greedy line search step were tested. The parameter σ entails a cost-quality trade-off: larger values provide higher quality at a larger cost. The parameter ω drives the extension of the search process. Experimentally, we found a good trade-off by using the following settings: $\sigma = 20, \omega = 2, \eta = 0.95$, independently of the base algorithms adopted in X-CLEAVER.

Finally, λ-MART resulted to outperform GBRT and Ω_λ-MART in all the tests conducted. Therefore, in the following we employ λ-MART as reference base algorithm for the evaluation of X-CLEAVER, while we use GBRT and Ω_λ-MART to provide experimental evidence of the generality of our framework, i.e., the capability of X-CLEAVER to improve the performance of different LtR ensemble-based algorithms.

The tests were performed on a machine equipped with a dual CPU AMD Opteron 6276, a 16 cores NUMA processor clocked at 2.30GHz, with 16 MB of cache L3 and 32GB of DDR3 RAM. To facilitate the reproducibility of the results, we release our implementation of X-CLEAVER as part of the QuickRank C++ library for Learning to Rank [5][3].

[3]The source code of QuickRank is available at: http://quickrank.isti.cnr.it.

4.1 Effectiveness of pruning strategies

We first investigate the effectiveness of X-CLEAVER pruning strategies when applied to ranking models of varying sizes previously trained with the reference λ-MART algorithm. This corresponds to run a single iteration of our X-CLEAVER meta-algorithm. A similar analysis is reported in [17]. However, the experiments discussed here are novel as two pruning strategies (RANDOM and QUALITY-LOSS) have been improved, while the QuickRank library was further optimized for both effectiveness and efficiency.

Tables 3, 4, and 5 report the values of *NDCG*@10 measured on the test sets for the original λ-MART models and the ones obtained by applying a single iteration of X-CLEAVER. Specifically, the three tables report the results obtained on both datasets starting from models consisting of 100, 500, and 1,000 trees, respectively. The cell reporting the *NDCG*@10 achieved by the reference λ-MART model is highlighted in each table using a dark-gray background. In the same row of each table we report the performance of the intermediate λ-MART models (trained by the same learning process) having a number of trees ranging from 90% to 10% of the reference one.

We are interested here in comparing the performance of the reference λ-MART models with those obtained with X-CLEAVER for different pruning levels. In particular we want to assess if our pruning and optimization strategies together are able to produce ranking models that are smaller and at least as effective as the reference model. As an example, the λ-MART reference model of 100 trees reaches on MSLR-WEB30K-F1 a *NDCG*@10 of 0.4540. The intermediate λ-MART model with 50 trees scores only 0.4333. The X-CLEAVER model of 50 trees, obtained from the MSLR-WEB30K-F1 100-tree model using the QUALITY-LOSS strategy, achieves instead a *NDCG*@10 of 0.4643. Note that LAST achieves better results than λ-MART thanks to the re-weighting procedure.

For each pruning strategy assessed in a different row of the tables, light-gray cells highlight where we achieved performances greater than or equals to those obtained using the reference λ-MART model. The values in bold highlight instead the most aggressive pruning rate preserving the ranking quality of the reference model. Moreover, values labeled with * identify the performance of the smallest models which proved to be statistically equivalent to the reference λ-MART model. Randomization test with 10,000 permutations and p-value ≤ 0.05 is performed to assess if the *NDCG*@10 differences between the reference and the pruned models are statistically significant or not [29].

The reported results confirm the validity of the proposed strategies. Table 3, referred to ensembles generated from a reference model of 100 trees, shows that X-CLEAVER is able to improve the reference model by using any pruning strategies. For the least aggressive pruning rates, we can observe also remarkable gains in terms of *NDCG*@10. On the MSLR-WEB30K-F1 dataset, RANDOM and QUALITY-LOSS perform the best in granting equivalent ranking quality with very aggressive pruning rate. In these cases X-CLEAVER is able to prune up to 80% of the original ensemble without loosing quality. On the Istella-S dataset, RANDOM, SKIP, QUALITY-LOSS, and SCORE LOSS strategies can prune up to 80% of the trees and obtain *NDCG*@10 figures greater or equal to that of the reference 100-tree model.

When the size of the reference λ-MART model is increased to 500 trees (Table 4), the best performing pruning strategy results to be QUALITY-LOSS. This strategy is able to prune up to 70% of the trees and provide models with accuracy equivalent or higher than the reference ensemble. The superiority of QUALITY-LOSS in pruning the ensemble is even more evident when models of 1,000 trees are considered (Table 5). Here, QUALITY-LOSS is the only pruning strategy allowing to improve the performance of the reference λ-MART ensemble with pruning rates of up to 60% on MSLR-WEB30K-F1 and to 50% on Istella-S.

Table 3. Values of *NDCG*@10 measured for λ-MART reference models of 100 trees and X-CLEAVER ones obtained at various pruning levels with the different pruning strategies. The * symbol labels the most aggressive pruning rate for each strategy resulting in a model statistically equivalent to the reference one.

	MSLR-WEB30K-F1									
	Pruned Model Size									
Strategy	100	90	80	70	60	50	40	30	20	10
λ-MART	.4540	.4507	.4477	.4432	.4386	.4333	.4260	.4179	.4085	.3983
LAST	.4648	.4620	.4605	**.4573***	.4516	.4482	.4407	.4300	.4107	.4003
RANDOM	-	.4644	.4635	.4629	.4624	.4606	.4602	.4599	**.4553***	.4455
SKIP	-	.4645	.4646	.4630	.4632	.4611	.4597	**.4596**	.4523*	.4465
LOW-WEIGHTS	-	.4646	.4647	.4653	.4633	.4622	**.4544***	.4411	.4188	.3888
QUALITY-LOSS	-	.4646	.4651	.4648	.4644	.4643	.4641	.4630	**.4580***	.4486
SCORE-LOSS	-	.4645	.4630	.4627	.4616	.4609	.4586	**.4554***	.4472	.4330

	Istella-S									
	Pruned Model Size									
Strategy	100	90	80	70	60	50	40	30	20	10
λ-MART	.6988	.6946	.6897	.6835	.6766	.6671	.6571	.6441	.6266	.6137
LAST	.7121	.7094	.7052	**.7001***	.6906	.6833	.6727	.6629	.6454	.6201
RANDOM	-	.7131	.7129	.7118	.7086	.7122	.7089	.7053	**.7015***	.6893
SKIP	-	.7124	.7117	.7111	.7104	.7086	.7076	.7041	**.7012***	.6906
LOW-WEIGHTS	-	.7122	.7120	.7115	.7104	**.7054***	.6919	.6674	.6404	.6204
QUALITY-LOSS	-	.7128	.7127	.7126	.7127	.7130	.7132	.7116	**.7098**	.6979*
SCORE-LOSS	-	.7104	.7091	.7108	.7110	.7068	.7052	.6937	**.7003***	.6852

Table 4. Values of *NDCG*@10 measured for λ-MART reference models of 500 trees and X-CLEAVER ones obtained at various pruning levels with the different pruning strategies. The * symbol labels the most aggressive pruning rate for each strategy resulting in a model statistically equivalent to the reference one.

	MSLR-WEB30K-F1									
	Pruned Model Size									
Strategy	500	450	400	250	300	250	200	150	100	50
λ-MART	.4766	.4758	.4752	.4752	.4745	.4722	.4694	.4644	.4540	.4333
LAST	.4783	.4780	**.4776**	.4765	.4765*	.4751	.4730	.4708	.4648	.4482
RANDOM	-	**.4772**	.4760*	.4744	.4748	.4743	.4741	.4698	.4679	.4615
SKIP	-	**.4773**	.4759	.4769	.4758*	.4746	.4741	.4724	.4698	.4622
LOW-WEIGHTS	-	.4759	.4754*	.4367	.4281	.4129	.4065	.3981	.3923	.3554
QUALITY-LOSS	-	.4781	.4781	.4777	.4780	.4787	.4760	**.4774***	.4740	.4686
SCORE-LOSS	-	.4753	.4758	.4753	.4750	.4753*	.4734	.4736	.4696	.4546

	Istella-S									
	Pruned Model Size									
Strategy	500	450	400	250	300	250	200	150	100	50
λ-MART	.7433	.7419	.7397	.7379	.7343	.7302	.7239	.7146	.6988	.6671
LAST	.7473	.7460	.7454	**.7439***	.7417	.7379	.7342	.7247	.7121	.6833
RANDOM	-	.7469	.7468	.7459	.7459	**.7450**	.7420*	.7374	.7348	.7249
SKIP	-	.7463	.7473	.7459	.7452	**.7448**	.7430*	.7396	.7367	.7255
LOW-WEIGHTS	-	.7470	**.7438***	.7390	.7147	.7022	.5963	.5789	.5551	.4969
QUALITY-LOSS	-	.7475	.7470	.7464	.7457	.7449	.7447	**.7433***	.7390	.7309
SCORE-LOSS	-	.7461	.7458	.7455	**.7435**	.7423*	.7376	.7320	.7248	.6999

Table 5. Values of *NDCG*@10 measured for λ-MART reference models of 1000 trees and X-CLEAVER ones obtained at various pruning levels with the different pruning strategies. The * symbol labels the most aggressive pruning rate for each strategy resulting in a model statistically equivalent to the reference one.

	MSLR-WEB30K-F1									
	Pruned Model Size									
Strategy	1000	900	800	700	600	500	400	300	200	100
λ-MART	.4789	.4785	.4784	.4779	.4774	.4766	.4752	.4745	.4694	.4540
Last	.4789	.4785	.4776	.4790	.4789	.4783*	.4776	.4765	.4730	.4648
Random	-	.4788	.4764	.4783*	.4754	.4768	.4720	.4721	.4670	.4580
Skip	-	.4784	.4778	.4782*	.4771	.4771	.4754	.4734	.4700	.4620
Low-Weights	-	.4788	.4780	.4788	.4788	.4783*	.4738	.4605	.4337	.4304
Quality-Loss	-	.4793	.4797	.4798	.4797	.4795	**.4804**	.4783*	.4758	.4722
Score-Loss	-	.4768	.4761	.4751	.4709	.4760	.4765	.4753	.4732	.4688

	Istella-S									
	Pruned Model Size									
Strategy	1000	900	800	700	600	500	400	300	200	100
λ-MART	.7495	.7491	.7478	.7467	.7451	.7433	.7397	.7343	.7239	.6988
Last	.7509	.7510	.7499	**.7498**	.7485*	.7473	.7454	.7417	.7342	.7121
Random	-	.7505	.7502	**.7506**	.7490*	.7464	.7446	.7430	.7387	.7297
Skip	-	.7506	.7495	**.7500**	.7485	.7493*	.7459	.7422	.7384	.7259
Low-Weights	-	.7489*	.7359	.6507	.6362	.6222	.5995	.5831	.5452	.5031
Quality-Loss	-	.7514	.7509	.7514	.7519	**.7507**	.7488*	.7462	.7431	.7371
Score-Loss	-	.7509	.7493	**.7498**	.7486*	.7479	.7460	.7453	.7365	.7196

From the three tables we can observe that the number of light-gray cells decreases as the size of the reference model increases. The possible explanation of this phenomenon is twofold. First, large models are more effective than small ones and the space left for improvements is thus very little. Second, our line search optimization process is not particularly effective on ensembles having a large number of trees since it is more likely to converge to a local optimum. We lean towards this second hypothesis that will be supported by some of the experiments discussed in the following.

We conclude that Quality-Loss is the best performing pruning strategy among the proposed ones, as it provides the smallest models in all experiments conducted. Indeed, it is the only strategy that exploits the ranking metric being optimized. The overall benefit of our proposal is remarkable since on both the datasets X-CLEAVER can exploit Quality-Loss to remove from 50% to 90% of the trees in the reference ensemble without hindering ranking quality.

4.2 Qualitative analysis of pruning strategies

We analyze here the behavior of the pruning strategies embedded in X-CLEAVER by evaluating the distribution of the trees removed from the ensemble. The analysis is conducted by applying a pruning rate $p = 50\%$ to a λ-MART model with $1,000$ trees trained on the Istella-S dataset.

The histograms reported in Figure 2 show the percentage of pruned trees across buckets of 100 consecutive trees of the input ensemble. As an example, the Last strategy (Fig. 2 upper-center) straightforwardly remove all the trees in the last five buckets, for a total of 500 trees out of $1,000$. Not surprisingly, the Skip strategy performs a uniform pruning as it removes equidistant trees from the input forest, and Random behaves similarly, despite in this case the pruning is not perfectly uniform due to the smart selection of the best performing pruned ensemble subset.

Interestingly, the Low-Weights strategy removes most of the trees from the first buckets of the ensemble. A possible explanation is that initial trees are the most redundant, while the last ones try

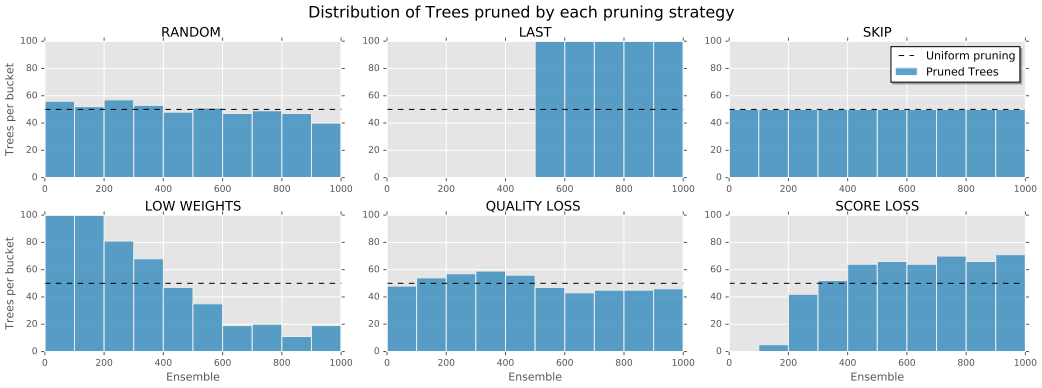

Fig. 2. Distributions of removed trees for each pruning strategy (with $p = 50\%$) applied to a λ-MART model with 1,000 trees trained on Istella-S. Values are averaged over buckets of 100 consecutive trees.

to fine-tune document scores. The Score-Loss strategy, which considers the relative contribution of each tree to document score, behaves symmetrically to Low-Weights and tends to prune trees from the last buckets of the ensemble. Boosting learning algorithms in fact assigns larger scores to the leaves of the initial trees which accounts for the largest part of the final score, and smaller scores to the last trees which are responsible for fine-tuning.

Quality-Loss, the best performing strategy, shows a smoother pruning behavior. Pruned trees are distributed almost uniformly, except for those in the range 100 to 500 where the pruning is more aggressive. This suggests that evaluating the contribution of each tree to the given metric function (as opposed to the final document score) leads to a more balanced pruning, which in turn results in a more accurate model.

Finally, we analyze how tree weights are modified by our ReWeight phase aimed at optimizing the *NDCG* score. To this end we measure the variations to per-bucket average weights after the Quality-Loss pruning strategy halved the size of a λ-MART model of 1,000 trees trained on Istella-S. Figure 3 shows the results of this analysis where the dashed horizontal line represents the uniform weights of the reference model, rescaled to the unit. Interestingly, the average weights in the first two buckets result to be lower than in the original model, while the weights in the last buckets are boosted. Moreover, the weights increase monotonically, suggesting that the last trees of the ensemble are very important, and that the original learning algorithm tends to underestimate their importance. The analysis reveals that, while the pruning step removes redundancy, the line search algorithm fine-tunes the weights by giving more importance to the last part of the forest.

4.3 X-CLEAVER analysis

In this section we discuss the experiments aimed at assessing if X-CLEAVER can train smaller and more accurate models than the reference LtR algorithm. We use as a reference the models trained with λ-MART by using the best training parameters discussed before (50 leaves and a learning rate equal to 0.05). The resulting ensembles have 1,199 and 1,497 trees for MSLR-WEB30K-F1 and Istella-S, respectively. Given the previous experimental results and to make the discussion clearer, we limit the analysis to the best performing pruning strategy, i.e., Quality-Loss.

As shown in Table 6, the reference λ-MART models achieve 0.4791 and 0.7530 in terms of *NDCG*@10 on MSLR-WEB30K-F1 and Istella-S, respectively. To provide a consistent baseline for each pruned X-CLEAVER model, we also report *NDCG*@10 values of the incremental λ-MART

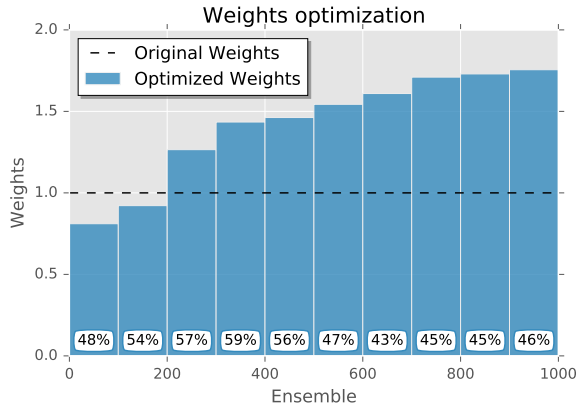

Fig. 3. Average per-bucket weights of the optimized model. The dashed line corresponds to the uniform weight used in the reference model. The value reported inside each bar measures the percentage of pruned trees in the bucket.

models generated every 100 iterations. As in the previous table, light-gray cells highlight performance greater or equals than the ones of the reference λ-MART model, while bold results evidence the best performing model for each ensemble size. Finally, values labeled with symbol * identify the performance of the smallest models which proved to be statistically equivalent to the reference one.

To first investigate the contribution of line search, we apply the re-weighting strategy to the incremental λ-MART models produced during the learning of the reference model (see line λ-MART + LS). On both datasets, the larger benefit is achieved on small models. On the MSLR-WEB30K-F1 dataset line search boosts the 600-trees model to the same performance of the 1,000-trees λ-MART. Similarly, on Istella-S, the 700-trees models after line search provides a better *NDCG* than the original 1,000-trees λ-MART. With larger models, line search results to be less beneficial. For example on the tests conducted with the MSLR-WEB30K-F1 dataset it does not provide any improvement from models with 800 and more trees.

To show that, even when applied only once, our optimization strategies still works well, we run a single iteration of the X-CLEAVER algorithm and report the results of the tests conducted in the row labeled X-CLEAVER$_{i=1}$. In this case we set X-CLEAVER to apply a $p = 50\%$ pruning rate to a λ-MART model being double in size than the final one. We first observe that X-CLEAVER$_{i=1}$ always achieves better results than line search only, for every model size tested. Second, we highlight that X-CLEAVER$_{i=1}$ is able to generate models statistically equivalent to the reference λ-MART one, having a much smaller number of trees: 300 trees (-75%) for MSLR-WEB30K-F1 and 800 trees (-47%) for Istella-S. This means that learning a larger number of trees and then pruning them with our QUALITY-LOSS strategy provide the subsequent line search procedure with a set of effective trees to be optimized. Pruning and re-weighting are thus able to boost each other in building compact and effective ranking models.

Finally, we address the main research question of this work, i.e., whether it is effective to embed pruning and re-weighting strategies within an ensemble learning algorithm. To answer this question we measure the performance of different X-CLEAVER models obtained by varying its hyper-parameters, i.e., the step size n, ranging in $\{100, 200, 400\}$, and the pruning rate, ranging in

Table 6. Comparison in terms of *NDCG*@10 among λ-MART and the different X-CLEAVER models, by varying p, n and N. Values in bold highlight the best performing model for each ensemble size, while light-gray cells highlight models performing equivalently or better than the reference λ-MART model. The * symbol labels the smallest model for each X-CLEAVER setting resulting to be statistically equivalent to the reference one.

MSLR-WEB30K-F1

Strategy	n	p	Model Size (N) 100	200	300	400	500	600	700	800	900	1000	1199	Time
λ-MART	-	-	.4540	.4694	.4745	.4752	.4766	.4774	.4779	.4784	.4785	.4789	.4791	18h 12m
λ-MART + LS	-	-	.4648	.4730	.4765	.4776	.4783*	.4789	.4790	.4776	.4785	.4789	-	
X-CLEAVER$_{i=1}$	$2N$	50%	.4733	.4767	.4792*	.4786	.4795	.4798	-	-	-	-	-	
X-CLEAVER	100	0%	.4643	.4734	.4750	.4770	.4775	.4777*	.4782	.4784	.4785	.4785	-	17h 42m
	200	50%	.4741	.4783*	.4797	.4801	.4807	.4803	.4807	.4810	.4809	.4810	-	34h 45m
	200	75%	**.4762**	.4781*	.4799	.4808	.4815	**.4825**	**.4829**	**.4828**	.4830	.4830	-	66h 57m
	400	50%	-	.4770	-	.4809	-	.4818	-	.4822	-	.4828	-	31h 01m
	400	75%	.4745	**.4790***	**.4809**	**.4810**	**.4822**	.4823	.4828	.4831	**.4834**	**.4841**	-	62h 23m
X-CLEAVER$_G$	400	75%	.4745	.4773	.4798*	.4799	.4804	.4801	.4811	.4818	.4829	.4816	-	69h 55m

Istella-S

Strategy	n	p	Model Size (N) 100	200	300	400	500	600	700	800	900	1000	1497	Time
λ-MART	-	-	.6988	.7239	.7343	.7397	.7433	.7451	.7467	.7478	.7491	.7495	.7530	8h 21m
λ-MART + LS	-	-	.7121	.7342	.7417	.7454	.7473	.7485	.7498	.7499	.7510	.7509	-	
X-CLEAVER$_{i=1}$	$2N$	50%	.7332	.7439	.7469	.7480	.7507	.7514	.7515	.7533*	-	-	-	
X-CLEAVER	100	0%	.7123	.7371	.7403	.7464	.7463	.7474	.7479	.7491	.7491	.7493	-	8h 47m
	200	50%	.7340	.7418	.7493	.7512	.7523*	.7545	.7545	.7551	.7550	.7551	-	16h 06m
	200	75%	.7375	.7464	**.7513**	**.7542***	**.7553**	**.7549**	.7544	.7550	.7552	.7551	-	37h 10m
	400	50%	-	.7438	-	.7504	-	.7545	-	.7556	-	.7575	-	13h 20m
	400	75%	**.7399**	**.7469**	.7497	.7514	.7530*	.7541	**.7546**	**.7557**	**.7566**	**.7577**	-	26h 11m
X-CLEAVER$_G$	400	75%	**.7399**	.7455	.7480	.7504	.7527*	.7545	.7548	.7563	-	-	-	31h 26m

{50%, 75%}. To provide some additional insights, we also evaluate X-CLEAVER with $n = 100$ and no pruning.

When no pruning is performed and line search applied after the generation of every bunch of 100 trees, no significant improvement is observed. In particular, for larger models, it is better to learn a λ-MART model and eventually apply line search just once. This again confirms the benefit of blending pruning and re-weighting altogether. When pruning is applied with a larger step size, results improve dramatically. X-CLEAVER always outperforms λ-MART models of the same size, even after line search, and it is able to provide better models than the reference λ-MART model for several pruning rates highlighted in the table by light-gray cells. In particular, X-CLEAVER achieves a *NDCG*@10 of 0.4790 on MSLR-WEB30K-F1 with only 200 trees, 83% less than the full reference model, and of 0.7542 on Istella-S with 400 trees, 73% less then the reference model. The best performing models are usually obtained with a pruning rate $p = 75\%$. This is in accordance with the experiments shown in Section 4.1, where QUALITY-LOSS is able to prune up to 70% of the original ensemble made up of 500 trees without loosing effectiveness. Using this pruning rate, the X-CLEAVER best models outperform the λ-MART ones with a *NDCG*@10 of 0.4841 (compared to 0.4791) on MSLR-WEB30K-F1 and 0.07577 (compared to 0.7530) on Istella-S.

Finally, we also test a variant of X-CLEAVER, employing the weight optimization step in a global fashion. We call it X-CLEAVER$_G$. Unlike the original X-CLEAVER, which applies line search

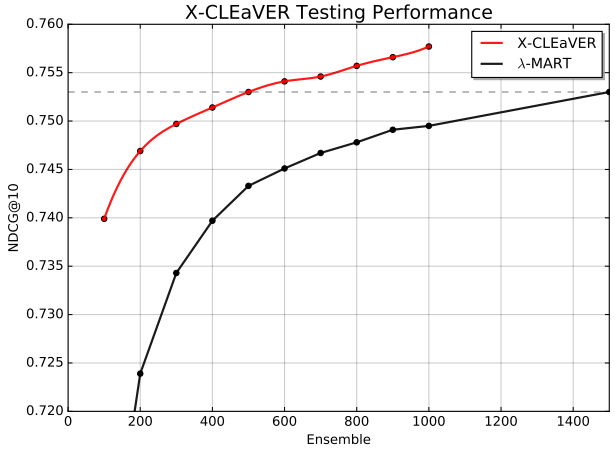

Fig. 4. Comparison of X-CLEAVER and λ-MART effectiveness in terms of *NDCG*@10 on the Istella-S dataset. The hyper-parameters adopted are $p = 75\%$ and $n = 400$.

and pruning only to the new trees trained at each iteration, X-CLEAVER$_G$ works by pruning and re-weighting the entire model at each iteration. Results show that this strategy achieve worse results than X-CLEAVER on both MSLR-WEB30K-F1 and Istella-S, proving the benefits of the local optimization strategy over a global one.

In summary, Figure 4 reports the *NDCG*@10 performance of the best performing X-CLEAVER model on the Istella-S dataset (i.e., $N = 1000$, $p = 75\%$ and $n = 400$), compared to that of the reference λ-MART model. The *NDCG*@10 values achieved by the two models is measured at every 100 trees. X-CLEAVER outperforms λ-MART for every ensemble size and the results indicate that the gap between the two models is considerable. Despite providing an increased effectiveness with respect to the reference model, the efficiency is as well increased, since X-CLEAVER is able to achieve the same performance of λ-MART with a reduced number of trees.

To conclude, Table 7 compares the actual per-document scoring time of different X-CLEAVER and λ-MART models. The scoring time is measured by exploiting QUICKSCORER which is the state-of-the-art algorithm for the evaluation of regression tree forests [9, 19]. As expected, the speed-up provided by X-CLEAVER compact models is proportional to the reduction in their size. On the MSLR-WEB30K-F1 dataset, the reference λ-MART models has 1,199 trees and requires about 22.33 μs to score a document, while X-CLEAVER can produce a model of only 200 trees with the same effectiveness but being four times faster at scoring time. Similar results are achieved on the Istella-S dataset with a 3.1 speed-up. When considering the X-CLEAVER model of 1,000 trees, despite showing a higher effectiveness in terms of *NDCG* with respect to the reference model, it proves to be also more efficient by a speed-up factor of 1.1 and 1.5 on MSLR-WEB30K-F1 and Istella-S, respectively.

We also compare X-CLEAVER with another solution described in literature, aiming at finding an efficiency/effectiveness trade-off. In particular, we consider the stage-wise tree pruning technique, proposed by Asady and Lin [1], which was proved to perform best in their experiments. As discussed in the related work section, this approach prunes the leaves of each tree immediately after it is built and independently of the others, with the goal of learning more compact, shallow, and balanced trees, thus yielding faster predictions. We train an λ-MART model adopting this pruning strategy,

Table 7. Per document scoring time of λ-MART and X-CLEAVER (n = 400,p = 75%) models. The * symbol labels model resulting to be statistically better to the λ-MART one.

MSLR-WEB30K-F1

Algorithm	# Trees	*NDCG*@10	Time (μs.)	Speed-up
λ-MART	1,199	0.4791	22.33	–
X-CLEAVER	1,000	0.4841*	20.35	1.1x
	200	0.4790	4.92	4.1x
Asady and Lin	1,172	0.4737	26.86	–

Istella-S

Algorithm	# Trees	*NDCG*@10	Time (μs.)	Speed-up
λ-MART	1,497	0.7530	48.33	–
X-CLEAVER	1,000	0.7577*	31.87	1.5x
	500	0.7530	15.64	3.1x
Asady and Lin	1,956	0.7430	11.14	–

by learning up to 2,000 trees per dataset. To avoid overfitting, however, the algorithm stops adding additional trees when there is no improvement in *NDCG*@10 on the validation set during the last 100 iterations. The algorithm has also a hyper-parameter α that defines a maximum unbalance tolerance threshold, and it thus drives the amount of pruning applied. We set this value to 0.2, according to the experimental results observed by the authors in their paper. By using these settings, the resulting forests are composed of 1,172 trees on MSLR-WEB30K-F1 and 1,956 trees on Istella-S. Results on MSLR-WEB30K-F1 show that the Asady and Lin algorithm is not able to provide any improvement, neither in terms of effectiveness nor in terms of efficiency. The behavior is different on the Istella-S dataset, where it obtains a significant speed-up with respect to λ-MART, at the cost of a significantly reduced accuracy. However, X-CLEAVER has a lightly slower scoring time ($\approx$ 15 vs $\approx$ 11 μs), without any loss in terms of *NDCG*@10.

4.4 Training behavior

We further analyze the training behavior of X-CLEAVER across multiple iterations thus investigating the impact of the pruning and re-weighting strategies. In Figure 5 we report the *NDCG*@10 achieved by X-CLEAVER on the training set as a function of the number of trees generated. In detail, the solid lines show the performance of λ-MART in performing the GrowModel phase devoted at learning a delta model $\widehat{\mathcal{E}}$, composed of *n weak rankers*. This phase ends at the blue circles, when the learning of a given delta model $\widehat{\mathcal{E}}$ ends, annotated by blue numbers that identify the X-CLEAVER iteration number. Then, X-CLEAVER executes the Prune and ReWeight phases, which lead to a model $\widehat{\mathcal{E}}_W$, corresponding to square red squares labeled with the same iteration number. For instance, during the first iteration, X-CLEAVER uses λ-MART to train 400 trees from scratch, and achieves a *NDCG*@10 of 0.7724. This point is identified by the blue number 1. Then, the pruning strategy discards 300 out of the 400 trees, and line search re-weights the remaining ones, leading to a *NDCG*@10 of 0.7821 identified by the red number 1. The gaps between the corresponding blue and red numbers (0.01 in this example) highlights the ability of X-CLEAVER to remove the less relevant trees and optimize the metric measure *NDCG*@10. Red numbers identify the performance of the X-CLEAVER model at the end of every iteration, thus describing the performance of the

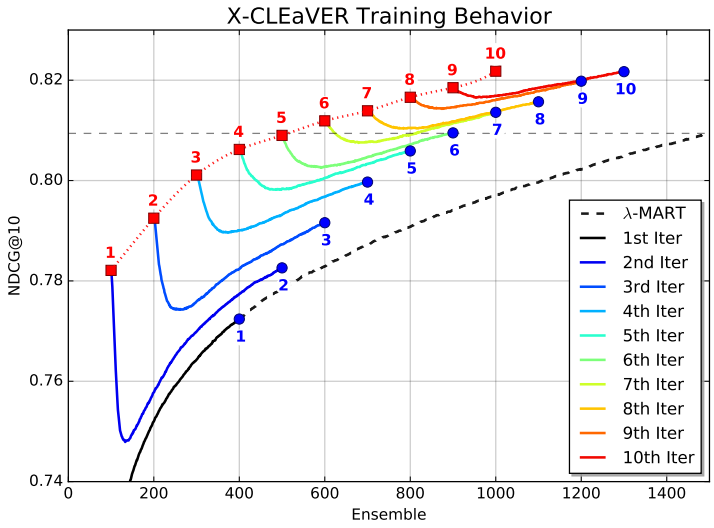

Fig. 5. Values of *NDCG*@10 measured on the training dataset (Istella-S) across multiple iterations of X-CLEaVER. The hyper-parameters adopted are $p = 75\%$ and $n = 400$.

intermediate models. Each subsequent iteration starts from the pruned and re-weighted model obtained at the end of the previous iteration, i.e., $\mathcal{E} = \mathcal{E} \cup \widehat{\mathcal{E}}_W$, shown as a new solid line of a different color. Lastly, the dashed black line identifies the performance achieved by the reference λ-MART algorithm. Since both X-CLEaVER and λ-MART start from scratch at the beginning, they initially train the same 400 trees. The horizontal dashed black line, instead, represents the performance of the reference λ-MART model on the training set.

An interesting effect deserving attention is the performance of the λ-MART algorithm when it restarts the training of a previously optimized model to produce a new delta model $\widehat{\mathcal{E}}$. Indeed, the first trees of the delta model, added to the optimized model $\mathcal{E}$ learnt so far, cause a significant drop of the performance on the training set. This is reflected by the descending direction of the training curve, at the beginning of each iteration (except the first one). The reason of this unexpected behavior can be explained by one of the motivation behind this work, i.e., λ-MART does not directly optimize the given ranking measure, e.g., *NDCG*, but can only indirectly optimize it by using the proxy λ-ranks. Apparently, the models produced at the end of each X-CLEaVER iteration contrast with the learning direction of λ-MART.

However, the performance drop never falls under the training performance of the previous X-CLEaVER iteration when considering same-sized models. In fact, each X-CLEaVER iteration brings significant improvements both before and after the local optimizations. This behavior deserves future investigations as it may shed light on novel optimization strategies for rank-based loss functions. As preliminary conclusion, we believe that the X-CLEaVER local optimizations have the effect of "teleporting" λ-MART on a different region of the gradient boosting search space that would not be explored otherwise. This allows to generate novel trees, some of them apparently introducing a performance drop and some of them providing a significant improvement with respect to the reference λ-MART. Eventually, the local optimizations of X-CLEaVER select the most relevant trees and re-weight them properly. We thus believe that X-CLEaVER benefits from a wider exploration of the search space.

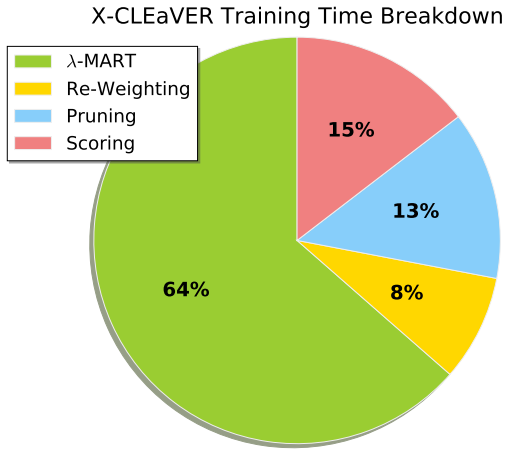

Fig. 6. X-CLEAVER computational cost breakdown. The hyper-parameters adopted are $p = 75\%$ and $n = 400$ on the Istella-S dataset.

4.5 Training cost analysis

The last column of Table 6 reports the training times of the various strategies, referred to the largest model trained. We recall that the training time is an off-line cost: it is worthwhile to pay an extra cost at training time if better or more efficient models can be exploited at prediction time. Overall, the best performing model of X-CLEAVER shows a training time up to 3.5 times slower than the reference λ-MART model, i.e., 62 hours against 18 for MSLR-WEB30K-F1, and 26 hours against 8 for Istella-S. The X-CLEAVER cost is mainly due to the larger number of trees trained: in order to build a model of $N = 1,000$ trees with a pruning rate of $p = 75\%$, the number of trees that we have to actually generate is 4,000. However, X-CLEAVER rewards this additional training cost with more accurate ranking predictions and more faster models.

We further investigate the training time of X-CLEAVER by considering the four main phases of the algorithm: i) training a new delta model $\widehat{\mathcal{E}}$ of n weak rankers using the λ-MART model; ii) computing score predictions $s_i(q, d)$ for every document in the training dataset and for every weak ranker in $\widehat{\mathcal{E}}$, thus allowing a more efficient implementation of the pruning and re-weighting processes; iii) pruning a fraction p of weak rankers according to the QUALITY-LOSS strategy; iv) re-weighting the remaining trees via line search. We measure the fraction time spent by X-CLEAVER across those phases in order to train the best performing model on the Istella-S dataset (i.e., $N = 1,000$, $p = 75\%$, and $n = 400$).

As shown in Figure 6, the λ-MART training accounts for about 2/3 of the total time. Learning the weak learners is the most demanding phase. Recall that in order to train 1,000 trees, a total of 4,000 trees are generated across the 10 iterations of the algorithm. The scoring phase has a non-trivial cost of about 15%. This is due to the large number of documents and trees generated during each iteration. Note that X-CLEAVER exploits λ-MART as a black box algorithm, and therefore every generated tree is re-evaluated. Otherwise, the process could be optimized by storing the tree predictions as they are learnt. Pruning has a similar cost of about 13%. Note that QUALITY-LOSS is the most expensive pruning strategy proposed. Nevertheless, the overall cost is relatively small and well balances the higher quality provided in comparison to the other pruning strategies. Finally,

the cost of line search optimization is limited to about 8%. Such limited cost is achieved thanks to several optimizations, including multi-threading exploitation.

The additional cost introduced by X-CLEAVER is not small. The local optimizations cover 1/3 of the total cost, and a much larger number of trees is generated to build the final model.

However, if we focus on models with similar ranking quality, we observe that the total training time of X-CLEAVER is comparable to that of λ-MART. When considering the X-CLEAVER algorithm with parameters $p = 75\%$ and $n = 400$, it is sufficient to train 200 trees on MSLR-WEB30K-F1 to achieve similar performance to the reference λ-MART model with 1,199 trees, and 500 trees are sufficient on Istella-S to equal the reference λ-MART model with 1,497 trees. In these cases, the training times of X-CLEAVER are of about 11 hours on both datasets, while λ-MART requires 18 hours on MSLR-WEB30K-F1 and 8 on Istella-S.

We thus conclude that X-CLEAVER has a training cost comparable with the state-of-the-art λ-MART algorithm. In addition, it is able to *i)* create much more compact models providing the same ranking quality, and *ii)* learn models with better ranking quality at the cost of some additional computation at training time.

4.6 Generalization to different LtR algorithms

We finally explore the use of X-CLEAVER with two additional LtR algorithms, GBRT and Ω_λ-MART, with the goal of understanding whether X-CLEAVER is able to improve efficiency and effectiveness of ranking ensembles independently of the base algorithm adopted. Recall that GBRT exploits MSE as a loss function, while Ω_λ-MART is a variant of λ-MART, but it generates balanced oblivious trees to reduce over-specialization and improve scoring efficiency.

Table 8 reports the values of *NDCG*@10 on the test sets of MSLR-WEB30K-F1 and Istella-S datasets for each of these LtR base algorithms, namely the algorithm $\mathcal{A}$ used by the GrowModel phase of X-CLEAVER. To give insights on the performance of the models with a growing number of trees, we also report *NDCG*@10 values every 100 learnt trees. As in the previous table, light-gray cells highlight models performing equivalently or better than the full base model (regarding the base algorithm adopted), while bold results evidence the best performing model for each ensemble size. Finally, values labeled with symbol * identify the performance of the smallest models which proved to be statistically equivalent to the full base one. The X-CLEAVER hyper-parameters adopted are those that lead λ-MART to the best performing model in previous analysis (i.e., $N = 1{,}000$, $p = 75\%$ and $n = 400$).

We first highlight that λ-MART is the best performing among the three LtR base algorithms on both the datasets, with a higher margin on Istella-S than on MSLR-WEB30K-F1 dataset. Indeed, on the Istella-S dataset λ-MART obtains the same effectiveness as a GBRT model made of 1,499 trees by using less than 300 trees, and less than 500 trees when compared to the full Ω_λ-MART model composed of 1,500 trees.

Regarding the adaptability of X-CLEAVER to different base learning algorithm $\mathcal{A}$, the figures reported in Table 8 show that X-CLEAVER boosts effectiveness independently of the base algorithm used, showing equivalently or better performance by using much less trees (highlighted in light-gray). In particular, X-CLEAVER$_{\mathcal{A}=\text{GBRT}}$ trains a model that is statistically equivalent to GBRT in terms of *NDCG*@10 by using only 200 trees on both the datasets, compared to the 1,041 and 1,499 trees of the full base model. Therefore, this saves up to 87% in the number of the trees composing the forests. X-CLEAVER$_{\mathcal{A}=\Omega_\lambda\text{-MART}}$, on the other hand, obtains the same effectiveness with only 300 trees on both the datasets, out of the 1,500 trees of the full base model, by thus saving 80% of the number of trees. Overall, given the specific differences among the base algorithms tested, X-CLEAVER achieves a significant boost independently o f the learning strategy and the

Table 8. *NDCG@10* obtained with GBRT, λ-MART and Ω_λ-MART state-of-the-art LtR algorithms and the X-CLEAVER models trained using these base algorithms for the GROWMODEL phase. Values in bold highlight the best performing model for each ensemble size, while light-gray cells highlight models performing equivalently or better than the full base model. The * symbol labels the smallest model for each X-CLEAVER setting resulting to be statistically equivalent to the full base one.

MSLR-WEB30K-F1

Strategy	Model Size (N)										
	100	200	300	400	500	600	700	800	900	1000	Full
λ-MART	.4540	.4694	.4745	.4752	.4766	.4774	.4779	.4784	.4785	.4789	.4791
GBRT	.4591	.4673	.4711	.4730	.4749	.4758	.4764	.4768	.4772	.4777	.4777
Ω_λ-MART	.4436	.4573	.4631	.4667	.4693	.4713	.4724	.4739	.4748	.4754	.4778
X-CLEAVER$_{\mathcal{A}=\lambda\text{-MART}}$	**.4745**	**.4790***	**.4809**	.4810	**.4822**	.4823	.4828	.4831	.4834	**.4841**	-
X-CLEAVER$_{\mathcal{A}=\text{GBRT}}$	.4743	.4786*	**.4809**	**.4817**	.4821	**.4829**	**.4838**	**.4836**	.4835	-	-
X-CLEAVER$_{\mathcal{A}=\Omega_\lambda\text{-MART}}$	.4722	.4757	.4790*	.4792	.4805	.4804	.4810	.4809	.4813	.4815	-

Istella-S

Strategy	Model Size (N)										
	100	200	300	400	500	600	700	800	900	1000	Full
λ-MART	.6988	.7239	.7343	.7397	.7433	.7451	.7467	.7478	.7491	.7495	.7530
GBRT	.7037	.7151	.7199	.7227	.7242	.7249	.7254	.7261	.7266	.7273	.7297
Ω_λ-MART	.6735	.7008	.7147	.7214	.7255	.7283	.7315	.7333	.7357	.7370	.7417
X-CLEAVER$_{\mathcal{A}=\lambda\text{-MART}}$	**.7399**	**.7469**	**.7497**	**.7514**	**.7530***	**.7541**	**.7546**	**.7557**	**.7566**	**.7577**	-
X-CLEAVER$_{\mathcal{A}=\text{GBRT}}$	.7262	.7316*	.7353	.7374	.7382	.7393	.7404	.7400	.7402	.7403	-
X-CLEAVER$_{\mathcal{A}=\Omega_\lambda\text{-MART}}$	.7335	.7388	.7405*	.7442	.7446	.7464	.7472	.7488	.7493	.7516	-

shape of the trees learnt by the LtR algorithm. We conclude this analysis by highlighting that, among the base algorithms tested, λ-MART is the one allowing X-CLEAVER to achieve the best performance, thus confirming the advantage in using a stronger LtR method as the base algorithm of our meta-algorithm.

4.7 Hyper-Parameters guidelines

We conclude the presentation of X-CLEAVER by detailing some "best practices" to achieve the best performance by properly setting the additional hyper-parameters introduced by the meta-algorithm. This discussion is based on the experience we made while performing the experiments presented in this section.

First, we discuss the pruning rate p and the step size n, assuming that the pruning strategy adopted is QUALITY-LOSS given its superior performance over the other proposed strategies. To this regard, we suggest to adopt a pruning rate p equals to 75%, because it leads to the best performance disregarding the step size n. Indeed, pruning rates close to 75% allow X-CLEAVER to achieve the best effectiveness on both its single-iteration (Tables 3, 4, and 5) and multiple-iterations (Table 6) variants. On the other hand, we observed that the step size n does not influence too much the effectiveness of the final models. For example, no significant effectiveness were observed by setting $n = 200$ or 400.

Finally, we highlight that p affects the training time of X-CLEAVER much more than n. First, a strong pruning rate p influences the number of meta iterations needed to reach the desired ensemble size, because less trees are kept in the additive model at the end of each meta iteration. Second, a smaller step size n also results in an increased number of meta iterations, despite each of

them costs less in terms of training time, given the reduced number of trees to be learnt per step, and the reduced number of weights to optimize.

To summarize, the two hyper-parameters impact both training time and effectiveness of the learnt model. If the former aspect is more important than the latter, we suggest to decrease at first the pruning rate, and then to increase the step size.

Similar considerations can be made for the hyper-parameters driving the greedy line search used in the re-weighting phase. The parameters σ and ω control the granularity of the local search. The extension ω of the local search is also reduced by a shrinking factor η at each iteration, to favor a fine-grained optimization when the algorithm approaches a local maximum. Larger values of σ entail higher quality but at a larger cost. The parameter ω instead, together with η, mainly affects the number of iterations needed until convergence. We experimentally found a good trade-off by using the following settings: $\sigma = 20, \omega = 2, \eta = 0.95$. However, to reduce the time spent in the re-weighting phase, σ can be lowered to half this value without hindering so much the effectiveness of the final model, and η can be increased so as to reach convergence with less iterations.

Finally, since hyper-parameters are usually dataset-dependent, for their setting we suggest to test their possible configurations in a single iteration of X-CLEAVER, thus reducing the tuning time. This hint comes from the experimental concordance of the results observed in both single-iteration and multiple-iterations variants of X-CLEAVER.

5 CONCLUSION

We focused on the problem of improving efficiency and effectiveness of ranking models based on additive ensembles of weighted regression trees.By analysing these models, we observed that the trees in the ensemble have some degree of similarity and that the list-wise metrics used to measure ranking quality, e.g., *NDCG*, differ from the loss functions minimized during the learning process. We thus proposed X-CLEAVER, a novel meta-learning algorithm that embeds local optimizations, i.e., iterated pruning and tree re-weighting steps, within the base LtR algorithm used to grow the tree forest. While pruning aims at reducing the redundancy present in the model, tree re-weighting counterbalances the negative effects of pruning on the model quality, by actually optimizing a rank-based quality measure like *NDCG*.

We analyzed several pruning strategies having the goal of identifying the redundant trees to be removed from the portion of ensemble generated at each iteration of X-CLEAVER, and we formalized a greedy optimization process to optimize the weights associated with the trees survived to the pruning phase. The experiments conducted on two public LtR datasets showed that the *pruning* and *re-weighting* phases successfully achieve their goals: *pruning* is able to remove up to 80% of the given ensemble, and *re-weighting* provides a boosts in performance well beyond the effectiveness of the base LtR algorithm. As a result, the models generated with X-CLEAVER remarkably outperforms those generated with the reference LtR algorithm. For example we showed that the *NDCG*@10 achieved on MSLR-WEB30K-F1 by a X-CLEAVER model of 200 trees is statistically equivalent to that measured with a λ-MART model of 1,199 trees. Similar figures were achieved on the Istella-S dataset, where a X-CLEAVER model of 400 trees resulted to provide the same ranking quality of a λ-MART model using 1,497 trees. From an opposite perspective, when the number of trees to include in the final model is fixed, the models trained with X-CLEAVER resulted to consistently outperform those generated with λ-MART in terms of *NDCG*@10. Analogous improvements were measured for X-CLEAVER models trained by using GBRT and Ω_λ-MART as base LtR algorithms, thus showing the generality of X-CLEAVER and its independence from the specific LtR solution adopted. As future work it would be interesting to investigate the application of the proposed

methodology to ensembles of weighted regression trees trained to address other regression and classification problems.

REFERENCES

[1] Nima Asadi and Jimmy Lin. 2013. Training Efficient Tree-Based Models for Document Ranking. In *Proc. ECIR 2013, Moscow, Russia, March 24-27, 2013. Proceedings*. 146–157.

[2] Nima Asadi, Jimmy Lin, and Arjen P. de Vries. 2014. Runtime Optimizations for Tree-Based Machine Learning Models. *IEEE Transactions on Knowledge and Data Engineering* 26, 9 (2014), 2281–2292.

[3] Berkant Barla Cambazoglu and Ricardo A. Baeza-Yates. 2015. *Scalability Challenges in Web Search Engines*. Morgan & Claypool Publishers.

[4] B Barla Cambazoglu, Hugo Zaragoza, Olivier Chapelle, Jiang Chen, Ciya Liao, Zhaohui Zheng, and Jon Degenhardt. 2010. Early exit optimizations for additive machine learned ranking systems. In *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 411–420.

[5] Gabriele Capannini, Domenico Dato, Claudio Lucchese, Monica Mori, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, and Nicola Tonellotto. 2015. QuickRank: a C++ Suite of Learning to Rank Algorithms (http://quickrank.isti.cnr.it/). *IIR '15: 6th Italian Information Retrieval Workshop* (2015).

[6] Gabriele Capannini, Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, and Nicola Tonellotto. 2016. Quality versus efficiency in document scoring with learning-to-rank models. *Information Processing & Management* 52, 6 (2016).

[7] Wei Chen, Tie-Yan Liu, Yanyan Lan, Zhiming Ma, and Hang Li. 2009. Ranking Measures and Loss Functions in Learning to Rank. In *Proceedings of the 22Nd International Conference on Neural Information Processing Systems (NIPS'09)*. 315–323.

[8] Van Dang, Michael Bendersky, and W Bruce Croft. 2013. Two-Stage learning to rank for information retrieval. In *Advances in Information Retrieval*. Springer, 423–434.

[9] Domenico Dato, Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Nicola Tonellotto, and Rossano Venturini. 2016. Fast Ranking with Additive Ensembles of Oblivious and Non-Oblivious Regression Trees. *ACM Transactions on Information Systems* 35 (2016). To appear.

[10] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of Statistics* (2001), 1189–1232.

[11] Jerome H Friedman, Bogdan E Popescu, and others. 2003. Importance sampled learning ensembles. *Journal of Machine Learning Research* 94305 (2003).

[12] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. 1999. Large margin rank boundaries for ordinal regression. *Advances in neural information processing systems* (1999), 115–132.

[13] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.* 20, 4 (2002), 422–446.

[14] Ron Kohavi. 1994. Bottom-up induction of oblivious read-once decision graphs. In *European Conference on Machine Learning*. Springer, 154–169.

[15] Leijun Li, Qinghua Hu, Xiangqian Wu, and Daren Yu. 2014. Exploration of classification confidence in ensemble learning. *Pattern recognition* 47, 9 (2014), 3120–3131.

[16] Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval* 3, 3 (2009), 225–331.

[17] Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, and Salvatore Trani. Post-Learning Optimization of Tree Ensembles for Efficient Ranking. In *Proc. SIGIR 2016*. ACM, 949–952.

[18] Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Nicola Tonellotto, and Rossano Venturini. Exploiting CPU SIMD Extensions to Speed-up Document Scoring with Tree Ensembles. In *In Proc. ACM SIGIR 2016*. ACM, New York, NY, USA.

[19] Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Nicola Tonellotto, and Rossano Venturini. 2015. QuickScorer: A Fast Algorithm to Rank Documents with Additive Ensembles of Regression Trees. In *Proc. SIGIR 2015*. ACM, 73–82.

[20] Craig Macdonald, Rodrygo LT Santos, and Iadh Ounis. 2013. The whens and hows of learning to rank for web search. *Information Retrieval* 16, 5 (2013), 584–628.

[21] Dragos D Margineantu and Thomas G Dietterich. 1997. Pruning adaptive boosting. In *ICML*, Vol. 97. 211–218.

[22] Manish Mehta, Jorma Rissanen, and Rakesh Agrawal. 1995. MDL-based Decision Tree Pruning. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*. AAAI Press, 216–221.

[23] Feng Nan, Joseph Wang, and Venkatesh Saligrama. 2016. Pruning Random Forests for Prediction on a Budget. *Advances in neural information processing systems* (2016).

[24] Chao Qian, Jing-Cheng Shi, Yang Yu, Ke Tang, and Zhi-Hua Zhou. 2016. Parallel Pareto Optimization for Subset Selection. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, USA*. 1939–1945.
[25] Chao Qian, Yang Yu, and Zhi-Hua Zhou. 2015. Pareto Ensemble Pruning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI'15)*. AAAI Press, 2935–2941.
[26] J. R. Quinlan. 1986. Induction of Decision Trees. *Mach. Learn.* 1, 1 (March 1986), 81–106.
[27] Shaoqing Ren, Xudong Cao, Yichen Wei, and Jian Sun. 2015. Global Refinement of Random Forest. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
[28] Ilya Segalovich. 2010. Machine learning in search quality at Yandex. Presentation at the industry track of the 33rd Annual ACM SIGIR Conference. https://goo.gl/xUAq3r. (2010).
[29] Mark D. Smucker, James Allan, and Ben Carterette. 2007. A Comparison of Statistical Significance Tests for Information Retrieval Evaluation. In *Proc. CIKM '07*. ACM.
[30] Ming Tan, Tian Xia, Lily Guo, and Shaojun Wang. 2013. Direct optimization of ranking measures for learning to rank models. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 856–864.
[31] Michael Taylor, Hugo Zaragoza, Nick Craswell, Stephen Robertson, and Chris Burges. 2006. Optimisation methods for ranking functions with multiple parameters. In *Proc. CIKM 2006*. ACM, 585–593.
[32] Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* (1996), 267–288.
[33] Lidan Wang, Jimmy Lin, and Donald Metzler. 2010. Learning to Efficiently Rank. In *Proc. SIGIR 2010*. ACM, 138–145.
[34] Q. Wu, C.J.C. Burges, K.M. Svore, and J. Gao. 2010. Adapting boosting for information retrieval measures. *Information Retrieval* (2010).

Received October 2017; revised March 2018; accepted April 2018.