Post-Learning Optimization of Tree Ensembles for Efficient Ranking

Claudio Lucchese
ISTI–CNR, Italy and Istella Srl
c.lucchese@isti.cnr.it

Franco Maria Nardini
ISTI–CNR, Italy and Istella Srl
f.nardini@isti.cnr.it

Salvatore Orlando
Università Ca' Foscari
Venezia, Italy
orlando@unive.it

Raffaele Perego
ISTI–CNR, Italy and Istella Srl
r.perego@isti.cnr.it

Fabrizio Silvestri
Yahoo Labs, London
silvestr@yahoo-inc.com

Salvatore Trani
ISTI–CNR, Italy
s.trani@isti.cnr.it

ABSTRACT

Learning to Rank (LtR) is the machine learning method of choice for producing high quality document ranking functions from a ground-truth of training examples. In practice, efficiency and effectiveness are intertwined concepts and trading off effectiveness for meeting efficiency constraints typically existing in large-scale systems is one of the most urgent issues. In this paper we propose a new framework, named CLEaVER, for optimizing machine-learned ranking models based on ensembles of regression trees. The goal is to improve efficiency at document scoring time without affecting quality. Since the cost of an ensemble is linear in its size, CLEaVER first removes a subset of the trees in the ensemble, and then fine-tunes the weights of the remaining trees according to any given quality measure. Experiments conducted on two publicly available LtR datasets show that CLEaVER is able to prune up to 80% of the trees and provides an efficiency speed-up up to 2.6x without affecting the effectiveness of the model.

Keywords

Learning to Rank; Efficiency; Pruning

1. INTRODUCTION

The problem of ranking objects in response to a user query is of general interest as it is of paramount importance for all kinds of information systems. In Web Search, for example, the problem of ranking documents is particularly challenging due to the large amount of data to manage. Modern Web search engines are expected to return highly relevant results in a fractions of seconds to satisfy hard back-end latency requirements. Nowadays, Learning-to-Rank (LtR) [6] methodologies are pervasively used as effective solutions to the most difficult ranking problems. LtR methods, which *score* a set of candidate documents according to their relevance to a given user query, use machine learning models trained on *ground truth* datasets providing an ideal rankings compiled either using human-based or automatic methods. Typically these approaches do not directly take into account efficiency concerns resulting in models that cannot be used in production environments because they do not meet their strict efficiency requirements.

In this paper, we tackle the problem of improving efficiency in (LtR). We present a framework, named CLEaVER, for the optimization of tree ensemble ranking models *after* the learning phase has completed. The cost of predicting the relevance for a document by using a tree ensemble model is linear in the number of trees. Larger ensembles, despite being more accurate, come with larger costs thus being very expensive. We study the following novel problem: given a tree ensemble model is it possible to improve its efficiency without affecting its quality? We show that the answer is positive, and we solve the problem by pruning some of the trees in the ensemble and re-weighting the remaining ones.

CLEaVER integrates in a novel way a number of tree pruning strategies with a local optimization method aimed at re-weighing the trees in the pruned ensemble. We analyze several pruning strategies for the identification of the less relevant trees in a given ensemble, and we conduct a comprehensive evaluation of the various strategies on two publicly available datasets. The CLEaVER framework is able to improve the efficiency of a given ranking ensemble up to a 2.6x speed-up without affecting the effectiveness of original the model.

Related Work. Efficiency in Learning to Rank has been investigated in the past from two different research directions. The first includes proposals that trade efficiency off for effectiveness in the learning algorithm. Asadi and Lin observe that compact, shallow, and balanced trees yield faster predictions [1]. They incorporate a notion of execution cost during training to encourage trees with these topological characteristics. Authors propose two strategies for accomplishing this: i) by directly modifying the node splitting criterion during tree induction, and ii) by stage-wise tree pruning. Experiments on a standard learning-to-rank datasets show that the pruning approach is the best. Wang *et al.* present a unified framework for jointly optimizing effectiveness and efficiency [10]. Authors propose new metrics that capture the trade-off between these two competing forces and devise a

SIGIR '16, July 17 - 21, 2016, Pisa, Italy

© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-4069-4/16/07...$15.00

DOI: http://dx.doi.org/10.1145/2911451.2914763

strategy for automatically learning models that directly optimize the trade-off metrics. Experiments show that models learned in this way provide a good balance between retrieval effectiveness and efficiency. Authors also show that their approach naturally leads to a reduction in the variance of query execution times, which is important for query load balancing and user satisfaction.

The second research line considers low-level optimizations to the inference phase by speeding-up the traversal of a given tree ensemble. Asadi *et al.* [2] propose to rearrange the code visiting the ensemble by transforming *control hazards* into less expensive *data hazards*, i.e., data dependencies introduced when one instruction requires the result of another. Lucchese *et al.* [7] propose QuickScorer a scoring algorithm, which 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.

To the best of our knowledge this is the first work focusing on post-learning optimization of tree-based LtR models. The two research directions above are orthogonal to ours as we aim at producing a faster model that does not lose its effectiveness and that can by integrated in any scoring algorithm. Moreover, our methodology is totally agnostic w.r.t. both the LtR algorithm. It can be applied to all LtR algorithms producing ensemble models.

2. OPTIMIZATION OF TREE ENSEMBLES

Let $\mathcal{F} = \{t_1, \ldots, t_n\}$ be an additive ensemble of regression trees, composed of n decision trees. Let $s_i(q,d)$ be the score returned by decision tree t_i to document d in response to query q. Moreover, we denote with $S(q,d)$ the final score predicted by forest $\mathcal{F}$ for query-document pair (q,d), obtained as a linear combination of tree predictions:

$$S(q,d) = \sum_{i=1}^{n} \lambda_i \cdot s_i(q,d)$$

where λ_i is the weight associated with tree t_i.

We make no assumption on the learning algorithm used to train $\mathcal{F}$ and on the loss function optimized during the training. Independently of the algorithm or the loss function adopted, we observe that the cost for computing score $S(q,d)$ is linear in the size n of the forest. As it is desirable to keep this cost as low as possible, either to comply with time budget constraints or to improve the overall effectiveness of query processing by ranking larger amount of candidate documents returned for a given query [4], we aim at reducing the complexity of a tree-based model by pruning trees. Specifically, given an input forest $\mathcal{F}$ providing the desired quality, CLEaVER produces a smaller forest $\mathcal{F}_p$ with at least the same effectiveness as $\mathcal{F}$ but with higher efficiency. Our solution consists in a combination of *pruning strategies* and *tree weighting* optimization. The CLEaVER framework encompasses the two following steps:

1. prune $\mathcal{F}$ to select a subset $\mathcal{F}_p$ of $p, p < n$ trees. We propose and assess different strategies to choose the trees to be removed from $\mathcal{F}$. These strategies estimate differently the impact of each tree in order to devise the less relevant ones;
2. assign new weights to the trees in $\mathcal{F}_p$ so as to optimize a given loss function. We use a greedy line search algorithm to optimize the weights λ_i by minimizing a given loss function.

The resulting forest $\mathcal{F}_p$ therefore contains a subset of the trees in $\mathcal{F}$ but with a different weighting scheme. Since our goal is to produce a more efficient model without affecting its effectiveness, CLEaVER evaluates different pruned model sizes p, and then returns the smallest forest $\mathcal{F}_p$ providing at least the same quality as $\mathcal{F}$ according to the user-defined loss function.

In the following, we first discuss the pruning strategies integrated in CLEaVER, and then discuss how line search is used to improve weights λ_i.

Ensemble Pruning Strategies. Let $p < n$ be the number of trees in the pruned forest $\mathcal{F}_p$. The goal of CLEaVER is to remove $n-p$ trees without affecting the predictive power of the initial forest $\mathcal{F}$. To this end, we propose the following strategies:

- Last: the last $n-p$ trees in the forest are pruned. The rationale is that forests are usually build sequentially by progressively adding trees optimizing a given loss function. The last trees learned generally provide limited changes in the predicted scores and ranking. The resulting model is equivalent to a model trained with p trees.
- Random: selects uniformly at random $n-p$ trees from the forest and removes them.
- Skip: keeps p equidistant trees from the forest. The rationale is that close trees are quite similar and potentially redundant. The distance between the selected trees is computed as $\lceil \frac{n}{p} \rceil$.
- Low-Weights: removes the $n-p$ trees with the lowest weights λ_i. The assumption is that they are less relevant for the final document scoring. When the learning algorithm used associates uniform weights to all the trees, a preliminary line search process is applied to $\mathcal{F}$ in order to obtain the initial vector Λ to which this strategy is applied.
- Score-Loss: considers the contribution given by each tree to the final score of a document. The normalized contribution for each tree is measured as $s_i(q,d)/S(q,d)$ and used to select the $n-p$ trees that contribute less.
- Quality-Loss: is based on the impact of each tree to the optimization of a given loss function L. For a given tree t_i, the average quality loss over the training dataset is computed by scoring the document with $S(q,d) - s_i(q,d)$. The $n-p$ trees that generate the smallest quality drop can be considered less important and thus removed from the forest.

All the pruning strategies are followed by the line search optimization aimed at tuning the weights of the trees in the pruned forest $\mathcal{F}_p$.

Tree Weighting with Line Search. Let $L \colon \mathbb{R}^p \to \mathbb{R}$ be a given loss function, and let $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_p\}$ be the current weighting schema for forest $\mathcal{F}_p$. Optimizing the tree weighting means finding:

$$\overline{\Lambda} = \operatorname*{argmin}_{\Lambda \in \mathbb{R}^p} L(\Lambda).$$

Due to the large dimensionality involved and the non differentiability of most loss functions, finding $\overline{\Lambda}$ is not possible. We thus adopt a heuristic approach to improve the weighting schema Λ obtained after pruning: *i)* find a descent direction $D \in \mathbb{R}^p$ along which L decreases, and *ii)* compute a step length α so as to minimize $L(\Lambda + \alpha \cdot D)$. This procedure is iterated as long as the solution improves beyond a given tolerance. The descent direction D can be chosen with various methods, *e.g.*, by computing gradient, and the step length α can be computed either exactly or approximately. The strategy we use to find α is referred to as *line search*. It is worth noting that the above process performs a local exploration of the solution space, without making any assumption on the loss function employed.

In our LtR scenario, we are interested in ranking loss functions such as *NDCG*, for which it is not possible to directly compute the gradient. Therefore, we adopt a greedy variant of the above procedure as in [9]. Given the current solution Λ^k at iteration k, we perform a line search along each axis of the weight vector independently, *i.e.*, find the value $d_i = \delta_i + \lambda_i^k$ that optimizes the loss function while keeping fixed all the other directions. The value of δ_i is found by choosing it among σ equi-spaced samples in an interval $[-\omega, +\omega]$ around λ_i^k, excluding negative values. The resulting vector D defines a promising descent direction along which to perform line search. The new weight vector Λ^{k+1} is chosen by optimizing $L(\Lambda^k + \alpha \cdot D)$. In this case, the best value for α is chosen among σ equi-spaced samples in the same interval $[0, 1]$. The above procedure is iterated until no improvement is measured on a separate validation set.

In order to make faster the above search process, at each iteration the search interval $[-\omega, +\omega]$ is reduced by a shrinking factor η. In addition, thread-level parallelism is used in order to evaluate different directions d_i and different σ samples in parallel. Finally, we avoid to visit the whole forest at each step. 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 a memory data structure.

3. EXPERIMENTS

Experiments were conducted with two publicly available datasets: the MSN-1[1] (Fold 1) dataset and a new dataset provided by Istella[2] (Small) we make publicly available with this work. The MSN-1 dataset consists of 31,351 queries and 136 features extracted from 3,771,125 query-document pairs, while the Istella dataset is composed of 33,018 queries and 220 features extracted from 3,408,630 query-document pairs. The query-document pairs in both the datasets are labeled with relevance judgments ranging from 0 (irrelevant) to 4 (perfectly relevant). Each dataset is split in train, validation and test sets according to a 60%-20%-20% scheme.

Training and validation data were used to learn a *full reference* model, to which we applied CLEaVER[3] by testing different pruning levels in the range 0%-90% with steps of 10%. Eventually, we selected the smallest (and thus most efficient) model generated by CLEaVER still providing greater or equal performance on the validation set w.r.t. the *reference un-pruned* model, measured in terms of *NDCG*@10. The parameters adopted for the greedy line search are $\sigma = 20$ equi-spaced samples taken from an interval of radius $\omega = 2$ and adopting a shrinking factor $\eta = 0.95$.

For the sake of fairness, we aimed at building a *full reference* model that is the best performing state-of-the-art ranking ensemble. On both datasets, we trained two different algorithms: *i)* the Gradient Boosting Regression Tree (GBRT, a.k.a., MART), a *point-wise* algorithm that uses the root mean squared error as loss function, resulting in a predictor of relevance labels [5], and *ii)* λ-MART, a *list-wise* algorithm that is capable of using *NDCG* in its loss function, resulting in a predictor of the ranking [11]. We used the open-source implementation of these algorithms provided by [3]. Both were fine-tuned by sweeping their parameters aiming at maximizing *NDCG*@10. The maximum number of leaves was tested in the set $\{5, 10, 25, 50\}$, while the learning rate in $\{0.05, 0.1, 0.5, 1.0\}$. To avoid overfitting, we allowed the algorithm to train up to $1,500$ trees unless the relative improvement in *NDCG*@10 on the validation set during the last 50 iterations was smaller than 0.5‰. The resulting forests include 737 and 736 trees for MSN-1 and Istella, respectively, with unitary weighting schema (*i.e.*, $\lambda_i = 1, 1 \leq i \leq n$). To assess the validity of the pruning strategies also for smaller forests, we build two more *reference* models consisting of the first 100 and 500 trees of the *full reference*. *NDCG*@10 values for the RANDOM strategy were averaged over 10 runs. We also performed randomization test [8] to assess if the differences in effectiveness between the reference models and the pruned ones are statistically significant.

We found that the largest number of leaves and the smaller learning rate always provide the best results with both LtR algorithms. We highlight that it was not possible, by varying the training parameters, to build more efficient rankers (e.g., with smaller total number of nodes) providing the same level of quality of the *reference* models. MART resulted the best performing on MSN-1, and λ-MART on Istella.

Results. Table 1 reports the performance of CLEaVER on MSN-1 and Istella datasets. In bold the best performance in terms of number of trees composing the pruned forest. Interestingly, both RANDOM and SKIP perform quite well, meaning that trees are somehow redundant by construction. Strategy LAST is the worst performing, suggesting that the last trees built by the model still provide a relevant contribution. The LOW-WEIGHTS pruning performs worse than expected, especially with the largest models. We deduce the tree weight cannot be evaluated in isolation from the actual predictions at the tree leaves. The pruning based on SCORE-LOSS exhibits good performance on average, especially on the Istella dataset. We highlight that this strategy approximates well the final document score, but small score variations may generate large rank variations, thus resulting in a negative impact on the *NDCG* metric. Finally, QUALITY-LOSS is quite consistently the best pruning strategy on both the datasets and for all the model sizes, except for the full λ-MART model on Istella dataset where SKIP and SCORE-LOSS are able to prune 10% more trees. Indeed, QUALITY-LOSS is the only strategy that takes into consideration the target quality function *NDCG*@10.

Pruning the reference models allows to significantly reduce the number of trees from 50% to 70% thus obtaining a scoring speed-up from 1.6x to 2.6x while preserving the effectiveness of the original *reference* model.

We run statistical significance tests, with significance level $p = 0.05$, obtaining a two-fold result. On the one hand, they

[1] http://research.microsoft.com/en-us/projects/mslr/
[2] http://blog.istella.it/istella-learning-to-rank-dataset/
[3] CLEaVER is available here: http://quickrank.isti.cnr.it.

Table 1: Comparison of CLEaVER pruning strategies against the *reference* models. Pruned forest size, document scoring time (μs), and *NDCG*@10 are reported for each of the pruned models.

MART on MSN-1

	100 Trees				500 Trees				737 Trees			
Strategy	Trees	Time	Speed-up	*NDCG*@10	Trees	Time	Speed-up	*NDCG*@10	Trees	Time	Speed-up	*NDCG*@10
Reference	100	5.55	–	0.4590	500	19.36	–	0.4749	737	27.46	–	0.4766
Random	40	3.28	1.7x	0.4603	350	14.27	1.4x	0.4756	516	19.64	1.4x	0.4768
Last	60	3.93	1.4x	0.4601	400	16.17	1.2x	0.4755	590	22.76	1.2x	0.4771
Skip	**30**	**2.84**	**2.0x**	0.4593	300	12.98	1.5x	0.4749	442	17.25	1.6x	0.4766
Low-Weights	40	3.26	1.7x	0.4609	400	15.75	1.2x	0.4753	663	25.28	1.1x	0.4779
Quality-Loss	**30**	**2.89**	**1.9x**	0.4618	**150**	**7.35**	**2.6x**	0.4752	**369**	**14.42**	**1.9x**	0.4771
Score-Loss	50	3.50	1.6x	0.4591	250	11.16	1.7x	0.4751	442	17.47	1.6x	0.4766

λ-MART on Istella

	100 Trees				500 Trees				736 Trees			
Strategy	Trees	Time	Speed-up	*NDCG*@10	Trees	Time	Speed-up	*NDCG*@10	Trees	Time	Speed-up	*NDCG*@10
Reference	100	5.37	–	0.6923	500	15.74	–	0.7397	736	20.40	–	0.7432
Random	30	2.65	2.0x	0.7003	250	9.10	1.7x	0.7424	515	14.81	1.4x	0.7449
Last	70	4.22	1.3x	0.6969	400	13.55	1.2x	0.7418	442	14.09	1.4x	0.7437
Skip	**20**	**2.36**	**2.3x**	0.6976	250	9.09	1.7x	0.7416	**368**	**11.42**	**1.8x**	0.7438
Low-Weights	30	2.85	1.9x	0.6986	350	11.07	1.4x	0.7418	589	15.55	1.3x	0.7437
Quality-Loss	**20**	**2.29**	**2.3x**	0.6989	**200**	**7.83**	**2.0x**	0.7412	442	13.22	1.5x	0.7438
Score-Loss	**20**	**2.13**	**2.5x**	0.6976	300	10.68	1.5x	0.7407	**368**	**12.39**	**1.6x**	0.7433

show a statistical equivalence when dealing with forests of 500 or more trees. On the other hand, they prove that the improvement of the pruned model w.r.t. the reference one when dealing with 100-tree models is statistically significant. Smaller models, whose quality is still far from optimal, can greatly benefit from the line search optimization, which can also better deal with a low-dimensional search space.

To confirm the effectiveness of CLEaVER we also performed the following experiment. We allowed pruning as long as no statistically significance difference w.r.t. the *reference* model is observed. In this setting, CLEaVER provides an even further improvement on the largest models: 70% pruning (221 trees) with Quality-Loss on MSN-1 and 60% (294 trees) with Skip on Istella, corresponding to a scoring speed-up of 2.9x and 2.1x respectively.

An important outcome of this work is that, in order to build a model with a given number of trees, we found to be best performing to use CLEaVER to prune a large model than to exploit line search optimization only. As an example, a λ-MART model with 100 trees on Istella dataset provides an *NDCG*@10 of 0.6923 and of 0.7085 after line search reweighting. Instead, by pruning a larger model of 500 trees with the Quality-Loss strategy, CLEaVER produces a model providing an outstanding *NDCG*@10 of 0.7329.

4. CONCLUSION AND FUTURE WORK

We proposed a novel algorithm, named CLEaVER, which is able to improve the efficiency of a given ensemble-based model. CLEaVER, first removes the less relevant elements in the ensemble according to their expected impact, then fine-tunes the weights of the remaining ones through line search. Eventually, CLEaVER provides a smaller and more efficient model with at least the same quality as the original one. In our experiments, CLEaVER allowed to built models being more than twice as fast. As future work we intend to integrate CLEaVER in several ensemble learning algorithms to systematically prune and improve the models while being generated. We also believe CLEaVER can be successfully extended to non tree-based ensembles and applied to other tasks than document ranking.

Acknowledgments. This work was partially supported by the EC H2020 Program INFRAIA-1-2014-2015 SoBigData: Social Mining & Big Data Ecosystem (654024).

5. REFERENCES

[1] N. Asadi and J. Lin. Training efficient tree-based models for document ranking. In *Advances in Information Retrieval*, pages 146–157. Springer, 2013.

[2] N. Asadi, J. Lin, and A. P. de Vries. Runtime optimizations for tree-based machine learning models. *IEEE TKDE*, 2014.

[3] G. Capannini, C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, and N. Tonellotto. Quality versus Efficiency in Document Scoring with Learning-to-Rank Models. *Information Processing and Management*, 2016.

[4] V. Dang, M. Bendersky, and W. B. Croft. Two-stage learning to rank for information retrieval. In *Advances in Information Retrieval*, pages 423–434. 2013.

[5] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 2001.

[6] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in IR*, 3(3):225–331, 2009.

[7] 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 *ACM SIGIR*, pages 73–82. 2015.

[8] M. D. Smucker, J. Allan, and B. Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *ACM CIKM*, 2007.

[9] M. Taylor, H. Zaragoza, N. Craswell, S. Robertson, C. Burges. Optimisation methods for ranking functions with multiple parameters. In *CIKM*, 2006.

[10] L. Wang, J. Lin, and D. Metzler. Learning to efficiently rank. In *ACM SIGIR*, 2010.

[11] Q. Wu, C. Burges, K. Svore, and J. Gao. Adapting boosting for information retrieval measures. *Information Retrieval*, 2010.