ILMART: Interpretable Ranking with Constrained LambdaMART

Claudio Lucchese
claudio.lucchese@unive.it
Ca' Foscari University of Venice
Italy

Franco Maria Nardini
francomaria.nardini@isti.cnr.it
ISTI-CNR
Italy

Salvatore Orlando
orlando@unive.it
Ca' Foscari University of Venice
Italy

Raffaele Perego
raffaele.perego@isti.cnr.it
ISTI-CNR
Italy

Alberto Veneri
alberto.veneri@unive.it
Ca' Foscari University of Venice
ISTI-CNR
Italy

ABSTRACT

Interpretable Learning to Rank (LtR) is an emerging field within the research area of explainable AI, aiming at developing intelligible and accurate predictive models. While most of the previous research efforts focus on creating post-hoc explanations, in this paper we investigate how to train effective and *intrinsically-interpretable* ranking models. Developing these models is particularly challenging and it also requires finding a trade-off between ranking quality and model complexity. State-of-the-art rankers, made of either large ensembles of trees or several neural layers, exploit in fact an unlimited number of feature interactions making them black boxes. Previous approaches on intrinsically-interpretable ranking models address this issue by avoiding interactions between features thus paying a significant performance drop with respect to full-complexity models. Conversely, ILMART, our novel and interpretable LtR solution based on LambdaMART, is able to train effective and intelligible models by exploiting a limited and controlled number of pairwise feature interactions. Exhaustive and reproducible experiments conducted on three publicly-available LtR datasets show that ILMART outperforms the current state-of-the-art solution for interpretable ranking of a large margin with a gain of nDCG of up to 8%.

CCS CONCEPTS

• **Information systems** → **Learning to rank**; • **Computing methodologies** → **Boosting**.

KEYWORDS

Interpretable Ranking, Interpretable Boosting, LambdaMART

ACM Reference Format:
Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, and Alberto Veneri. 2022. ILMART: Interpretable Ranking with Constrained LambdaMART. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22), July 11–15, 2022, Madrid, Spain*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3477495.3531840

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGIR '22, July 11–15, 2022, Madrid, Spain
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8732-3/22/07…$15.00
https://doi.org/10.1145/3477495.3531840

1 INTRODUCTION

Learning to Rank (LtR) is a vast research area focusing on learning effective models for solving ranking tasks. Although being effective, state-of-the-art LtR techniques do not deal with the *interpretability* of the learned model, i.e., the possibility given to humans to easily understand the reasoning behind predictions made by the model. The need for learning trustworthy models in LtR is becoming a topic of increasing interest, which tries to address important concerns about privacy, transparency, and fairness of ranking processes behind many everyday human activities.

For what regards the fairness of ranking algorithms, several solutions have been proposed to create fairer models, where the term *fair* takes different meanings according to the metric employed [2]. However, focusing only on devising a fairer model does not imply to be compliant with new regulations that are emerging. An example of that is *The Right of Explanation*, which allows each European citizen to claim a justification for every algorithm decision made with their data that affects their life [13]. Therefore, for highly-sensitive scenarios new LtR approaches have to be investigated to train ranking models that do not jeopardize interpretability in the name of effectiveness.

In literature, two main lines of work dealing with the interpretation of ranking models are investigated. The first concerns the creation of post-hoc explanations for black-box models, while the second focuses on the design of intrinsically-interpretable models. In the first line, the goal is the definition of techniques that allow, given a ranking model, to generate post-hoc explanations of the decisions made. Here, some of the explanation techniques used in regression or classification tasks have been adapted to the ranking scenario. For example, two recent contributions [17, 19] present two different approaches to adapt LIME [15] to the ranking task. The work by Singhs and Anand focuses only on pointwise LtR techniques and treats the ranking problem as a classification problem, where the probability of a document to be relevant is computed with respect to its score, and a simple linear SVM is used as a local explanation model [17]. The work by Verma and Ganguly evaluates the consistency and correctness of the explanations generated by LIME by varying the sampling strategies used to create the local neighborhood of a document [19]. In the same line, Fernando *et al.* propose an approach to reuse SHAP [12] for the ranking task [5].

Even though post-hoc explanations can be seen as a viable way to inspect and audit accurate ranking models, they can produce explanations that are not faithful to internal computations of the model, and thus they can be misleading [16]. For this reason, Zhuang *et al.* pioneer the problem of developing intrinsically-interpretable ranking models [20]. In detail, they propose *Neural RankGAM*, an adaptation of GAM [6] to the ranking task. Although Neural RankGAM nicely exploits the interpretability of GAMs, the authors show that the ranking performance achieved is significantly worse than those provided by state-of-the-art black-box models. However, it is worth noticing that in other contexts, such as in regression and classification tasks, other intrinsically interpretable models appear to be as accurate as other black-box models. A notable example of an interpretable and effective solution is Explainable Boosting Machine (EBM), a C++ implementation of the ^{2}" algorithm [9], where each component of the Generalized Additive Models (GAM) are learned through a mixed bagging-boosting procedure [8].

In this paper, we contribute to the development of intrinsically-interpretable ranking models with ILMART, a novel LtR solution based on LambdaMART. ILMART creates effective and intelligible models by encompassing a limited and controlled number of pairwise feature interactions. We evaluate ILMART on three publicly-available LtR datasets. The results of our reproducible experiments show that ILMART outperforms the current state-of-the-art solution for interpretable ranking of a large margin, with a gain in terms of nDCG of up to 8% with respect to Neural RankGAM.

2 INTERPRETABLE LAMBDAMART

ILMART produces ensemble-based ranking models having the structure of a GAM, which is considered an interpretable model [8]. Before introducing ILMART, we review the main aspects of GAMs to highlight similarities and differences with respect to the newly proposed technique.

Generalized Additive Models. Let $\boldsymbol{x} \in \mathbb{R}^3$ be an instance in a 3-dimensional vector space, with G_9 being the value of its 9-th feature. An instance $\boldsymbol{x}$ is associated with a target label $\sim$ generated by some unknown target function $6(\cdot)$, i.e., $6(\boldsymbol{x}) = \sim$. In its basic form, a GAM models the target variable $\sim$ as a combination of $?$ main effects and non-zero interaction effects, as defined in [6]:

$$;(\grave{}(\boldsymbol{x})) = U + \sum_{9=1}^{?} B_9(G_9) + \sum_{9=1•:=1}^{?=?} B_{9:}(G_9•G_:)$$

Where $\grave{}(\boldsymbol{x}) = [\sim|\boldsymbol{x}]$, with $\sim$ following a distribution of the exponential family, and $;(\cdot)$ being the so-called *link function* that is able to describe the relationship between the $?$ main effects and the interactions effects with the expected value of $\sim$. In addition, the number of non-zero interaction effects is equal to , i.e. $|\{B_{9:}(\cdot)|B_{9:}(\cdot) < 0\}| =$. Usually, the $?$ main effects and the and the interactions are modeled through simple third order splines. From the explainability point of view, the analyst is promptly provided with a plot of each basis function B_9 that clearly describes the relation between each feature and the target variable $\sim$, while $B_{9:}$ values represent the contribution of the interactions between feature pairs. For classification and regression tasks, (shallow) decision trees instead of splines were investigated in [9].

Interpretable LambdaMART. We now introduce ILMART, our novel method based on LambdaMART to learn ranking models that are both accurate and interpretable. ILMART presents similarities with GAM, and builds the prediction $\hat{\sim}$ of a unknown target variable $\sim$ as follows:

$$\hat{\sim} = \underbrace{\sum_{9\in\mathcal{J}} g_9(G_9)}_{|\mathcal{J}|=? \text{ main effects}} + \underbrace{\sum_{(8•9)\in\mathcal{K}} g_{89}(G_8•G_9)}_{|\mathcal{K}|= \text{ interaction effects}} \qquad (1)$$

$\underbrace{\phantom{\sum_{9\in\mathcal{J}} g_9(G_9)}}_{' \text{ trees}}$ $\underbrace{\phantom{\sum_{(8•9)\in\mathcal{K}} g_{89}(G_8•G_9)}}_{) \text{ trees}}$

where $g_9(G_9)$ and $g_{89}(G_8•G_9)$ are ensembles of trees. Each tree $\mathbb{C} \in g_9(G_9)$ models a *main effect* in which the only feature allowed for each split is G_9.

On the other hand, each tree $\mathbb{C} \in g_{89}(G_8•G_9)$ models the *interaction effects*, where the only features allowed in a split of tree $\mathbb{C}$ are either G_8 or G_9. Similarly to the notation used to describe a GAM, $?$ denotes the number of distinct main effects G_8, whereas denotes the number of distinct interaction effects $(G_8•G_9)$.

To learn the ensemble of trees modeled by Eq. (1), we adopt a modified version of the LambdaMART boosting algorithm. Unlike GAM, we do not make any assumption on the distribution of $\sim$ to guide the learning process. The learning strategy to generate the main effects $\mathbb{C} \in g_9(G_9)$ and the interaction effects $\mathbb{C} \in g_{89}(G_8•G_9)$ in Equation 1 is made of three steps depicted in Fig. 1: *i)* Main Effects Learning, which learns a set of trees, each one working on a main effect (single feature); *ii)* Interaction Effects Selection, which selects the top- most important interactions effects (feature pairs); *iii)* Interaction Effects Learning, which learns a set of trees, each one exploiting one of the interaction effects selected by the previous step.

Main Effects Learning. In the initial step, illustrated on top of Fig. 1, we learn an ensemble of $'$ trees modeling the main effects. To this end, we constrain the LambdaMART boosting procedure to use a single feature per tree in the first $'$ boosting rounds. In other words, at a given boosting round of LambdaMART, when feature G_9 is chosen for the root's split of a tree $\mathbb{C}$, we force the algorithm to use G_9 for all the further splits in $\mathbb{C}$, until a stopping criterion for tree growth is reached. LambdaMART stops its boosting rounds when no improvements are observed on the validation set. Eventually, we obtain an ensemble of $'$ trees only using for their splits a set $\mathcal{J}$ of $?$ features, $? \leq 3$ and $? \leq '$, modeling the main effects of Equation 1. Thus, $?$ is not an input parameter of ILMART, but it is the number of distinct features used by the ensemble of $'$ trees modeling the main effects only. We name this initial model ILMART. It is worth noting that, as a side effect of this first step, we achieve the additional effect of potentially reducing the cardinality of the features, from 3 to $?$.

Interaction Effects Selection. In the second step, we still exploit the LambdaMART boosting procedure to *select* the top- interaction effects, as illustrated in the middle of Fig. 1. Specifically, we continue the learning of the ILMART model obtained so far, by enabling the interactions between all possible pairs of $?$ features identified by the previous step, according to the interaction heredity principle [4]. This is accomplished by constraining LambdaMART to fit trees with only 3 leaves (and 2 splits), where the 2 splits use a pair of distinct

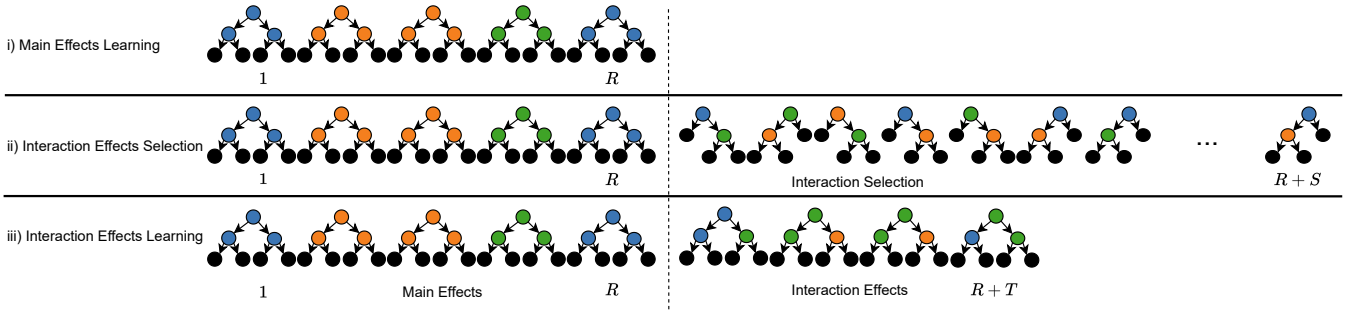

Figure 1: ILMART learning process. Each color corresponds to a different feature used in the splits, and nodes colored in black are leaves. The trees for the main and interaction effects are represented with the same depth for illustration purposes only.

features. The boosting round is stopped when the size of the set $\mathcal{K}$, composed of the distinct feature pairs used by the ensemble of (trees, is exactly , where $\leq \binom{?}{2}$. The (trees are generated solely for the purpose of identifying the interaction effects and they are discarded afterwards.

Note that for selecting the main interaction effects, we rely on the capability of LambdaMART in identifying, at each boosting round, the features and the splits with the highest contributions in minimizing the loss function.

Thus, the interaction effects selected by the boosting procedure are likely be generated in order of importance.

Interaction effects learning. Finally, the last step sketched on the bottom of Fig. 1 consists in adding to the initial ILMART model) new trees learned by constraining LambdaMART to use only the top- feature pairs identified in the previous step. The number) of new learned trees is not an input of algorithm, but it is chosen on the basis of the validation set. The final model, obtained by adding to ILMART the) trees working on the interaction effects is named ILMART$_{\beta}$.

3 EXPERIMENTAL EVALUATION

We experimentally assess the performance of ILMART and ILMART$_{\beta}$ on three public datasets for LtR, namely Istella-S, Web30K, and Yahoo. The Istella-S dataset [11] includes 33,018 queries with an average of 103 documents per query. Each document-query pair is represented by 220 features. The Web30K dataset [14] is composed of more than 30,000 queries (in 5 folds), with an average of 120 documents per query and 136 features per query-document pair. The Yahoo dataset [3] is composed of two sets, including document-query pairs with 699 features. The feature vectors of the three datasets are labeled with relevance judgments ranging from 0 (irrelevant) to 4 (perfectly relevant). In our experiments, we use train/validation and test splits from fold 1 of the Web30K dataset and data from "set 1" of the Yahoo dataset.

Competitors. We compare the performance of ILMART and ILMART$_{\beta}$ with the following competitors:

- Neural RankGAM (NRGAM), a neural-based approach to learn GAMs for ranking. The approach exploits standalone neural networks to instantiate sub-models for each individual feature. It is the current state-of-the-art technique for learning interpretable models for ranking [20].
- EBM, an efficient implementation of ^{2} " using a mixed bagging-boosting procedure [9]. In this case the learned model is used as a pointwise ranker, since the implementations available are only made to solve regression and classification tasks. We denote with EBM$_{\beta}$ the EBM model using interaction effects.

Metrics. We measure the performance in terms of normalized Discounted Cumulative Gain (nDCG) at three different cutoffs, i.e., {1• 5• 10}. We compute the nDCG metric by employing exponential weighing of the relevance [1]. By default, queries with missing relevant documents have been assigned a nDCG = 1"0. The statistical significance is computed with a two-sided Fisher's randomization test [18] with significance level ? < 0"05. The test is computed using the RankEval library [10].

Implementation and training settings. We implement ILMART as an extension of the LightGBM library[1] [7]. The extension works by adding the possibility to constraint the boosting procedure to use only a limited number of features per tree. The source code used in our experiments along with the models trained on the three public datasets is publicly available online[2].

The training of ILMART and ILMART$_{\beta}$ optimizes nDCG@10. We perform hyper-parameter tuning by varying the number of leaves in the trees in {32• 64• 128} and the learning rate in {0"001• 0"01• 0"1}. Early stopping, i.e., stopping boosting if no improvement on the validation set is observed for 100 consecutive rounds, is used for training in step one (main effects learning) and three (interaction effects learning). The optimal ILMART models learned with the settings above are made of 914, 895 and 1,969 trees for the Web30K, Yahoo, and the Istella-S datasets, respectively. On the same datasets, the three optimal ILMART$_{\beta}$ models are made of 1,365, 1,215 and 3,024 trees, respectively. Neural RankGAM is also trained by optimizing nDCG@10. We use the public implementation of NRGAM available in Keras[3] and made available by the authors of the original paper [20]. We train NRGAM by using the same settings reported by the authors. In addition, we employ a batch size of 128, and we train the networks with 3,000 epochs for Web30K, 700 for Yahoo and 1,000 for Istella-S, with early stopping set to 100 epochs. We also learn EBM by optimizing nDCG@10. The performance of EBM is fine-tuned by varying the number of outer-bags in {5, 10, 15} for

[1] https://github.com/microsoft/LightGBM
[2] https://github.com/veneres/ilmart
[3] https://github.com/tensorflow/ranking/issues/202

Table 1: Performance comparison in terms of nDCG. Statistically significant improvements w.r.t. NRGAM are marked with an asterisk (*). Best results are reported in bold.

Dataset	Method	nDCG @1	nDCG @5	nDCG @10	?	
Web30K	NRGAM	44.80	43.72	45.73	136	0
	EBM	41.70	43.14	45.66	136	0
	EBM$_8$	46.51*	45.96*	48.01*	136	50
	ILMART	45.17	45.00*	47.05*	79	0
	ILMART$_8$	**48.94***	**47.76***	**49.55***	79	46
Yahoo	NRGAM	67.92	70.13	75.07	699	0
	EBM	70.33*	72.77*	77.29*	699	0
	EBM$_8$	70.22*	72.85*	77.36*	699	50
	ILMART	70.57*	72.64*	77.21*	153	0
	ILMART$_8$	**70.80***	**73.20***	**77.57***	153	48
Istella-S	NRGAM	63.37	63.04	69.18	220	0
	EBM	57.75	59.67	66.90	220	0
	EBM$_8$	62.29	63.05	69.71	220	50
	ILMART	66.99*	66.29*	72.36*	106	0
	ILMART$_8$	**69.06***	**68.22***	**74.04***	106	50

the two versions with and without interactions. In our experiments, we employ a publicly-available implementation of EBM[4]. For what regards the two strategies encompassing interaction effects, i.e., ILMART$_8$ and EBM$_8$, we limit the number of maximum interactions allowed to 50.

Experimental results. Table 1 reports the results of the evaluation of ILMART, ILMART$_8$ and their state-of-the-art competitors. For all the datasets considered, results show that ILMART achieves a statistically significant improvement in terms of nDCG for almost all the cutoffs except the case of nDCG@1 on Web30K. More importantly, ILMART$_8$ always achieves the best performance in terms of nDCG that always result in a statistically significant improvement over NRGAM. The best improvements of ILMART$_8$ over *all* the competitors is obtained on Istella-S, where nDCG@10 improves of up to 6% w.r.t. EBM$_8$. On Web30K, we observe that ILMART$_8$ improves nDCG@10 of up to 8% w.r.t NRGAM, although the best competitor is, in this case, EBM$_8$, whose performance still lags behind ILMART$_8$. On Web30K, EBM$_8$ also achieves statistically significant improvements in terms of nDCG with respect to NRGAM. Even though EBM$_8$ is used as a pointwise ranker, these differences confirm the efficacy of tree-based approaches in LtR and the need to include pairwise interactions to learn more effective ranking functions.

Furthermore, it is worth highlighting that our proposed techniques perform a feature selection during the *main effects learning* phase, while the competitors leave the feature selection to the final user. This is an important difference between ILMART and its competitors as a high number of features used do not allow to easily understand the global behavior of the model. For example, the Yahoo dataset comes with a large number of features, i.e., 699. ILMART and ILMART$_8$ effectively reduce the initial set of 699 features to 153 (column ? in Table 1) with, at the same time, a significant increase of the effectiveness of the learned model. In addition, we point out that eventually ILMART$_8$ may have a number of interaction effects smaller than 50 due to the training of the new) trees which may not include all the pairs selected.

We now investigate how the number of interactions added to ILMART$_8$ can influence its prediction accuracy. Fig. 2 compares the results of ILMART$_8$ with the ones of NRGAM by reporting their results in terms of nDCG. We observe that the ranking accuracy of ILMART$_8$ increases steeply when the first interaction effects are added; then, the gain obtained by adding new interaction trees saturates. This means that ILMART$_8$ is able to significantly improve the accuracy of an interpretable LtR model by adding a small number of interaction effects, thus without hindering the overall interpretability of the model. Even though the addition of interactions effects improves a lot the accuracy of the model, we acknowledge that the difference from the full-complexity model LambdaMART is still marked. For example, on Web30K an optimized black-box LambdaMART model achieves a nDCG@10 close to 52%.

Finally, to show how the main and interaction effects can be explored, Fig. 3 presents three main effects and one interaction effects learned with ILMART$_8$ trained over Web30K. The three main effects selected are the ones associated with the features with highest feature importance, i.e., PageRank (PR), Query-url Click Count (QUCC), and Language Model for Information Retrieval (LIMIR), while the interaction effect between URL Click Count (UCC) and QUCC is the one with the highest average contribution for each value of (G_8•G_9). The 2D plots and the heatmap are built by aggregating all the trees using the same features, i.e., by representing the contribution of each $g_9(G_9)$ and each $g_{89}(G_8$•$G_9)$. From the plots, the analyst can easily understand the exact behavior of the model at varying values of each feature. For example, taking into account the interaction effect, the higher the value of UCC and QUCC the larger is the positive contribution of $g_{89}(G_8$•$G_9)$.

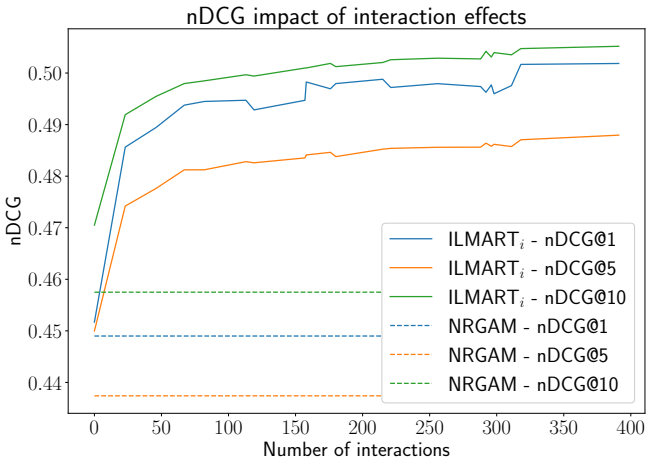

Figure 2: nDCG by varying the number of interaction effects in ILMART$_8$ on the Web30K dataset.

[4]https://interpret.ml/docs/ebm.html