Filtering out Outliers in Learning to Rank

Federico Marcuzzi
federico.marcuzzi@unive.it
Università Ca' Foscari Venezia
Venice, Italy

Claudio Lucchese
claudio.lucchese@unive.it
Università Ca' Foscari Venezia
Venice, Italy

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

ABSTRACT

Outlier data points are known to affect negatively the learning process of regression or classification models, yet their impact in the learning-to-rank scenario has not been thoroughly investigated so far. In this work we propose SOUR, a learning-to-rank method that detects and removes outliers before building an effective ranking model. We limit our analysis to gradient boosting decision trees, where SOUR searches for outlier instances that are incorrectly ranked in several iterations of the learning process. Extensive experiments show that removing a limited number of outlier data instances before re-training a new model provides statistically significant improvements, and that SOUR outperforms state-of-the-art de-noising and outlier detection methods.

CCS CONCEPTS

• **Information systems → Learning to rank**; **Retrieval effectiveness**.

KEYWORDS

information retrieval, learning to rank, machine learning

ACM Reference Format:
Federico Marcuzzi, Claudio Lucchese, and Salvatore Orlando. 2022. Filtering out Outliers in Learning to Rank. In *Proceedings of the 2022 ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR '22), July 11–12, 2022, Madrid, Spain.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3539813.3545127

1 INTRODUCTION

Most Learning-to-Rank (LtR) learning algorithms assume that training sets do not contain noise and outliers [7]. However, this belief is not always true, because we are in a context where human-labeled datasets are used. Moreover, the features of query-document vectors may not be sufficient to discriminate relevant documents from non-relevant ones [3]. For example, in a pair-wise context, which still covers a large part of the state of the art LtR, the presence of outliers or erroneously labeled documents leads to a quadratic growth in the number of incorrect pairs. Consequently, it becomes necessary to consider the possible presence of outlier instances at training time. Unlike classification or regression, little has been done for LtR in this regard. Even less attention has been given to GBDT-based learning algorithms, which are nowadays the state of the art in LtR in terms of both effectiveness and efficacy.

In this work we propose a novel method for handling outliers in a LtR scenario. Our contributions are as follows. First we propose a definition of outlier instance in the LtR task that exploits the iterative nature of gradient boosted decision trees. The algorithm pays particular attention to outliers present in the training set that the model finds particularly difficult to rank correctly during subsequent gradient boosting rounds. Then, we show how the removal of these outliers leads to a statistically significant performance increase in terms of NDCG [12] on three publicly available datasets. In addition, we provide an extensive experimental analysis, we investigate the behaviour of the proposed algorithm on different kinds of queries and the impact of the outlier removal on the raw predictions of the forest. Furthermore, we support the goodness of our algorithm by comparing against several variants also inspired by related but less closer works.

The rest of the paper is structured as follow: Section 2 discusses the related work. In Section 3 we present *Surrender on Outliers and Rank* (SOUR), the main contribution of this work. Section 4 provides an extensive evaluation of SOUR against state-of-the-art competitors. In Section 5 we analysis in details the effects of SOUR on the learned models and in different scenarios. Finally, in Section 6 we conclude the work.

2 RELATED WORK

One of the most common strategies for managing outliers is *sample selection* [15, 18]. Outliers are found and treated differently than legitimate samples (removed [13], learned separately [11], reweighed [7], etc.). In [15] the authors propose a sample selection algorithm called Isolation Forest. As the name suggests, the algorithm trains the decision tree forest with a subset of the training set, and then uses the resulting model to predict whether the remaining instances are outlier or not. Instances that have, on average, a short path from root to leaves within the forest are considered outliers. An interesting early work is Robust C4.5 [13], a variant of C4.5 algorithm [20], robust to outliers. The algorithm iterates, by training at each phase a decision tree, and then using the resulting model to remove from the training set the instances that are misclassified. In the next phase, the pruned dataset is used to train the next decision tree, etc. The algorithm iterates until no classification errors occur in the training set. This technique, even if applied to a classification task, has some similarity to our method, but is much more aggressive and applies to classification only.

In [23] the authors argue that strategies based on selective sampling are sensitive to changes in noise distribution. Furthermore, the difficulty in identifying outlier instances may lead to the elimination of a number of legitimate instances. Another way to deal with noisy labels inside the dataset is robust learning. Most of these

---

ICTIR '22, July 11–12, 2022, Madrid, Spain

ACM ISBN 978-1-4503-9412-3/22/07…$15.00
https://doi.org/10.1145/3539813.3545127

Algorithm 1 SOUR Algorithm.

```
1:  function SOUR($\mathcal{D}, A, s, e, t, \mathcal{M}$)
2:      $F \leftarrow A(\mathcal{D}, \mathcal{M}, e)$                    ▷ Train a decision forest

3:      $k \leftarrow \mathcal{M}$.CUTOFF                          ▷ Get metric cutoff
4:      for $i \in \{s, s+1, s+2, \dots, e\}$ do    ▷ Find outliers after $i$ trees
5:          $F_i$ = GetTreesUpTo($F, i$)
6:          $O_i^- \leftarrow$ {not relevant documents with rank (by $F_i$) $\leq k$
7:                  above a relevant document with rank $> k$}
8:          $O_i^+ \leftarrow$ {relevant documents with rank (by $F_i$) $> k$
9:                  below a not relevant document with rank $\leq k$}

10:     $O^+ \leftarrow \bigcap_{i \in [s,\dots,e]} O_i^+$              ▷ Compute consistent outliers
11:     $O^- \leftarrow \bigcap_{i \in [s,\dots,e]} O_i^-$
12:     $O^* \leftarrow O^+ \cup O^-$
13:     $O \leftarrow$ Pick($O^+, O^-, O^*, t$)                 ▷ Select outliers

14:     $F \leftarrow A(\mathcal{D} \setminus O, \mathcal{M})$                ▷ Train on the clean dataset

15:     return $F$
```

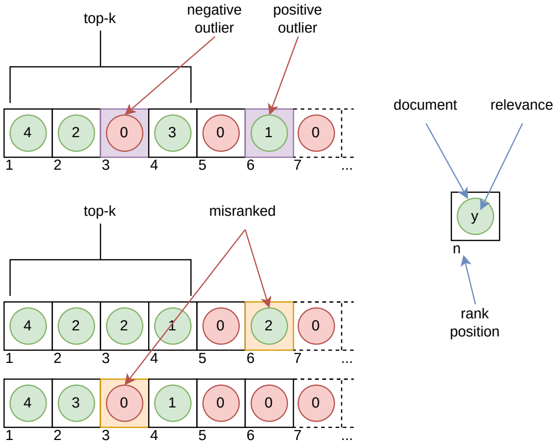

Figure 1: Outlier vs. misranked documents. Top: the document at rank 3 occupies a position in the top-k ($k = 4$) that might be filled by the relevant document at rank 6 and vice versa. Bottom: even if the ranking is not correct, the top-k are either full with relevant documents or contain all the possible relevant documents.

strategies attempt to define a robust loss function in the presence of noise in the training set [8, 10, 21, 23, 24]. An attempt in the LtR context was made in [3], a pair-wise meta-learning approach that takes a linear model as input and generates an outlier-robust model. The input model hypothesis is refined through a non-linear sigmoid function. The sigmoid meta ranker performs a non-convex optimization that converges to a local optimum close to the original hypothesis. The algorithm exploits the non-linearity of the sigmoid function in order to suppresses the effect of outliers. In other words it softens the large negative loss generated by incorrectly ordered pairs. A more recent attempt of noise-robust training in LtR is PeerRank [23], which uses the PeerLoss [16] loss function. The idea is to add synthetic noise instances by drawing at random data points from the training set and pairing them with random labels. PeerLoss [16] maximizes the given loss function on the original training instances and maximizes it on the synthetic noisy ones, so as to punish models that rank correctly such noise instances. PeerRank focuses on neural networks, and we provide a more effective implementation based on LambdaMART [22].

3 SOUR LEARNING-TO-RANK ALGORITHM

In this section we present our LtR algorithm which exploits outliers removal at training time. The rationale is that, rather than training on the full dataset, it is more beneficial to remove outlier instances before the training process. Hence the name *Surrender on Outliers and Rank* (SOUR). The proposed algorithm, illustrated in Alg. 1, targets gradient boosted decision trees and encompasses three steps: *i)* base model training, *ii)* outlier detection and removal, and *iii)* final model training.

During the first step (line 2), a base decision forest F is trained up to e trees on the full dataset $\mathcal{D}$ by using any given LtR algorithm A and a quality metric $\mathcal{M}$. Ideally, F is the best model we can learn by using A. Note, SOUR focus on quality metrics that use a cutoff, such as NDCG, ERR [6], Precision, etc., but it can easily handle other metric trough hyper-parameter tuning.

The second and core step of the algorithm (lines 4–13) consists in using the learnt model F to identify ourliers in the training dataset. To discuss our technique, we need to distinguish between *outlier* and *misranked* instances. For the sake of simplicity and efficiency, we identify outlier instances solely on the basis of the documents' rank, derived from the scores that F assigns them. We recall that the most commonly used quality metrics, including NDCG, use a cutoff threshold k, thus limiting their evaluation to the contribution of the top-k ranked documents.

Intuitively, we name *positive outliers* those relevant documents that are "pushed out" of the top-k scored documents and *negative outliers* those not relevant documents that are "pushed in" the top-k ranks. We give below a formal definition which is exemplified in Fig. 1.

Definition 3.1 (Positive Outlier). Given a ranked list of labeled documents D and a threshold k, a document $d \in D$ is said to be a *Positive Outlier* if *a)* it is relevant, i.e., it has a relevance label greater than 0, *b)* it is ranked below rank k and *c)* there is at least one document with label equal to 0 within rank k.

Definition 3.2 (Negative Outlier). Given a ranked list of labeled documents D and a threshold k, a document $d \in D$ is said to be a *Negative Outlier* if *a)* it is not relevant, i.e., with label equal to 0, *b)* it is ranked within rank k and *c)* there is at least one document below rank k with label greater than 0.

We call *misranked documents* all other ranking errors do not comply with neither of the above two definitions. Fig. 1 shows examples of outlier documents (top) versus misranked ones (bottom). Note that no (negative) outliers exist when the top-k positions only include relevant documents, and no (positive) outliers exist when all the relevant documents fall in the top-k positions.

As we are focusing on a forest model F, we also consider the intermediate rankings generated after i stages/trees of F. We think that an outlier document may affect all the steps of the training process and, therefore, looking only at the final model F might be a sub-optimal strategy. Let F_i be the sub-forest including only the first i trees of F, we look all the positive/negative outliers detected at multiple stages of the model F and we search for documents that are consistently considered outliers across those intermediate stages, as defined below.

Definition 3.3 (Consistent Positive/Negative Outliers). Given a ranked list of labeled documents D, a ranking forest F, two integer constants s and e, and let F_i be the sub-forest including only the first i trees of F. We define as *Consistent Positive/Negative Outliers* those documents that result to be positive/negative outliers in ranking generated by the sub-forest F_i, for every $i \in [s, e]$.

We argue that the model finds more difficult to properly rank these documents and that this difficulty is due to the outlier nature of those instances, rather than to the limitations of the learning algorithm.

Equipped with the above definitions, we can discuss the core step of SOUR (lines 4–13). SOUR has three tuning hyper-parameters: the first and last tree of interest s and e, and the outlier type to be exploited $t \in \{\textsc{pos}, \textsc{neg}, \textsc{all}\}$ which can be positive, negative or their union. For each forest cut $i \in [s, e]$, we first extract the sub-forest F_i that includes the first i trees of F. The sub-forest F_i is used to score and rank every document in the training dataset $\mathcal{D}$, and for each ranked list in $\mathcal{D}$ we compute the positive and negative outliers according to Definitions 3.1, 3.2, respectively denoted with O_i^+ and O_i^-. Then, we compute the set of consistent outliers by intersecting all the positive or negative outliers found so far, i.e., $O^+ = \bigcap_i O_i^+$ and $O^- = \bigcap_i O_i^-$. We also consider the set $O^* = O^+ \cup O^-$. Depending on $t \in \{\textsc{neg}, \textsc{pos}, \textsc{all}\}$, one of O^+, O^-, O^* is chosen to be the outlier set identified by SOUR.

During the last step (line 14), a new model is trained after removing the outliers O from the input dataset $\mathcal{D}$, and the resulting forest is eventually returned.

The SOUR algorithm incurs the additional cost of running A a second time, which is typically not an issue given the efficiency of tree-learning algorithms. We observe that the cost of outlier detection during the second step is negligible w.r.t. the cost of running A. In the experimental section we also show that the size of O is rather small and, therefore, running A on the cleaned dataset is still as expensive as with the original dataset. Yet, the resulting model brings interesting performance improvements.

4 SOUR EXPERIMENTAL EVALUATION

4.1 Datasets

We conduct experiments on three publicly available datasets: Yahoo! Learning to Rank Challenge Set 1[4], MSLR Web30K Fold 1[19] and Istella-X[17], summarized in Tab. 1. We highlight that they have very different average query lengths. The limited number of documents clearly makes Yahoo! unsuitable for our analysis, but we include it anyway for the sake of completeness. All datasets include graded relevance labels with a different fraction of non relevant documents: Yahoo! having 26% out of 709,877, MSLR-30K with 51% out of 3,771,125 and Istella-X with 99.8% out of 26,791,447. We retain the provided train, validation, test splits.

Table 1: Datasets properties.

	Istella-X	MSLR-30K	Yahoo!
#features	220	136	519
#queries	10,000	31,531	29,921
avg. query length	2,679.14	119.60	23.72
#documents	26,791,447	3,771,125	709,877
relevance label distribution			
0	26,475,076	1,940,952	185,192
1	26,604	1,225,770	254,110
2	5,108	504,958	202,700
3	9,619	69,010	54,473
4	5,040	90,435	13,402
%non relevant documents	99.8%	51%	26%

Table 2: SOUR hyper-parameter tuning on Istella-X. NDCG is computed on the validation set.

		t = neg		t = pos		t = all	
s	e	#trees	NDCG@10	#trees	NDCG@10	#trees	NDCG@10
0	1000	940	0.7702	523	0.7686	724	**0.7696**
100	1000	693	0.7669	893	0.7698	766	0.7677
200	1000	870	**0.7717**	451	0.7659	587	0.7679
300	1000	566	0.7688	997	0.7684	687	0.7684
400	1000	537	0.7685	579	0.7679	824	0.7684
500	1000	971	0.7699	626	0.7689	718	0.7670
600	1000	635	0.7694	566	0.7667	949	0.7684
700	1000	568	0.7684	690	0.7672	671	0.7682
800	1000	624	0.7680	939	***0.7728***	679	0.7684
900	1000	982	0.7717	783	0.7683	960	0.7685
0	100	762	0.7679	998	0.7653	936	0.7651
0	200	837	0.7715	692	0.7676	775	0.7641
0	300	724	0.7692	998	0.7685	797	0.7683
0	400	912	0.7686	709	0.7643	692	0.7688
0	500	833	0.7701	437	0.7614	690	0.7668
0	600	438	0.7674	689	0.7696	517	0.7628
0	700	627	0.7693	824	0.7686	507	0.7660
0	800	607	0.7670	563	0.7694	884	0.7660
0	900	996	0.7707	1000	0.7701	973	0.7698

4.2 Models

As baselines, we use λ-LGBM and λ-MART to refer to the λ-MART [22] variants implemented by the LightGBM [14] software respectively with and without gradient normalization. SOUR is implemented[1] on top of λ-LGBM. Hyper-parameters tuning follows previous works [2, 17].

4.3 SOUR hyper-parameter tuning

In this subsection, we discuss in detail results only for the Istella-X dataset. The same behaviour was observed for the other datasets.

---

[1] Source code available at https://anonymous.4open.science/r/Filtering-out-Outliers-in-Learning-to-Rank-E89A/.

Tab. 2 reports NDCG@10 scores of SOUR on the Istella-X validation set when varying s (start) and e (end) parameters and removing one of the three different sets of outliers, namely O^-, O^+, or O^*, according to parameter $t \in \{\text{NEG}, \text{POS}, \text{ALL}\}$. For our analysis, where we vary the SOUR's parameters s, e and t, we exploit a forest F of 1000 trees. Once selected the document instances to drop, we train the final forest with early stopping on the validation set for the number of trees. The size of the final model is reported in the table.

We report results on the two most interesting scenarios we found. In the first case, we search for documents that were considered outliers by all the last trees of the forest ($e = 1000$), while in the second case we consider outlier documents in the initial trees ($s = 0$). We include the case $s = 0$, $e = 1000$ that removes documents that resulted to be outlier in every stage of the forest.

Overall, results are quite close, but we can observe that the ALL strategy performs consistently worse than the others and that on average the best results are achieved when $e = 1000$. The latter result is inline with expectations. The initial trees are still on an unstable path along the gradient descent and, therefore, the outlier documents might be very different from those of the final model. Conversely, when setting $e = 1000$ and varying s we may ignore errors in the initial trees and focus on outlier documents in the final trees of the ranking algorithm.

In Fig. 2 we show the number of documents actually removed on varying the hyper-parameters of SOUR. The number of outliers is generally very small and varies from ~2,000 to a maximum of ~12,000. Keeping $e = 1000$ and varying s allows to remove a larger number of documents. As discussed above, this is due to the great instability of the initial trees which is likely to reduce the intersection computed to determine the consistent outliers. Therefore, not only the quantity, but also the quality of the consistent outliers computed when s is not small is expected to be larger. According to Tab. 2, the best results were achieved with $t =$ POS, $s = 800$ and $e = 1000$, which corresponds to the removal of about 2,400 documents in a dataset with more than 26 million instances. We show that the removal of this limited number of documents is sufficient to achieve statistically significant improvements.

Hereinafter, we limit the analysis to $e = 1000$ and choose the other parameters on the basis of the validation set performance.

To conclude the analysis of the t hyper-parameter, in Fig. 3 we show the behaviour on the Istella-X validation set of the best settings of SOUR which are highlighted in boldface in Tab. 2. In this comparison we include the baseline λ-LGBM model. SOUR performs significantly better than the baseline with any configuration of t. In fact, all the three variants have a steeper NDCG curve achieving earlier an interesting accuracy. The plot, which includes the 100 early stopping iterations, shows how the proposed SOUR is able to continue increasing its accuracy when additional trees are used w.r.t. λ-LGBM. Later we report final performance on the test set.

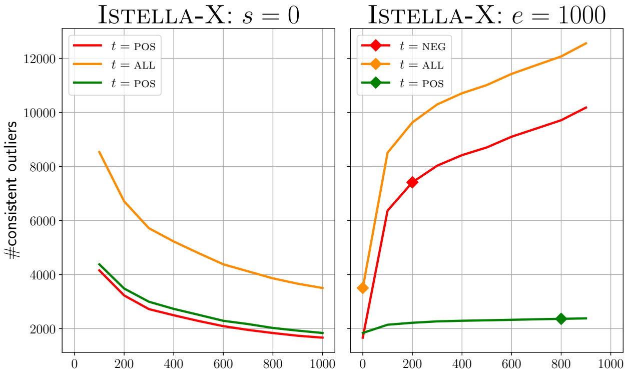

Figure 2: Number of consistent outlier documents removed by SOUR when varying s, e and t. Diamonds are the best s values in correspondence to the highest NDCGs in Tab. 5 for each t.

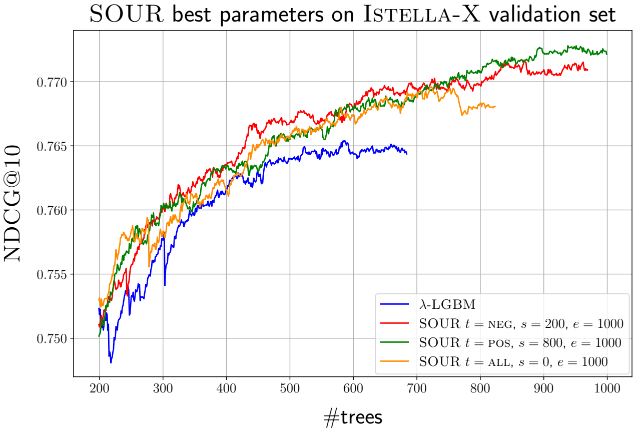

Figure 3: Performance of SOUR when varying t on Istella-X validation set.

4.4 Does model-based outlier detection perform better than data-based outlier detection?

There are two common methods for detecting outliers, which we refer to as data-based and model-based. The former consists in finding instances that do not fit with the data distribution at hand, whereas the latter strategy is to spot instances that are hard to be correctly classified by a given model. SOUR falls into the second category. In Fig. 4 we evaluate Isolation Forest [15] as a method to detect outliers. We implemented a variant named Iso-λ-LGBM where an isolation forest is used to remove outliers from the dataset before training the final λ-LGBM model. As for SOUR, on the basis of parameter t, we filter out only positive outlier, only negative ones, or both kinds of documents. For the isolation forest, we adopted the parameters suggested by the authors [15].

Fig. 4 shows the performance of Iso-λ-LGBM on varying t against the proposed SOUR and the λ-LGBM baseline. Performance is measure in terms of NDCG@10 on the validation set of Istella-X. First we note that $t =$ ALL is the best setting for Iso-λ-LGBM, with interesting improvements over λ-LGBM. However, the proposed SOUR performs better than Iso-λ-LGBM, thus allowing to conclude that the proposed model-based approach is more effective than a data-based strategy based on isolation forests.

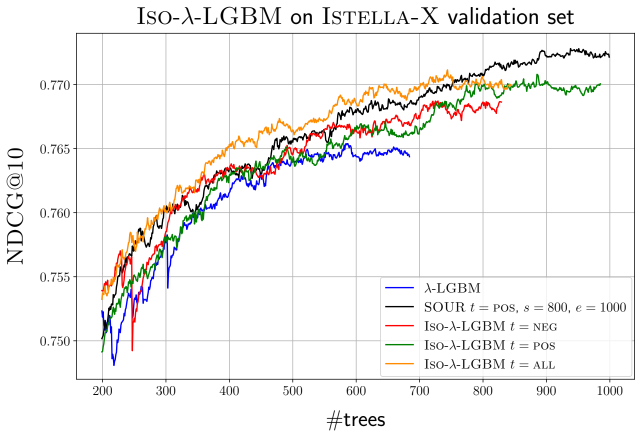

Figure 4: Performance of Iso-λ-LGBM when varying t on Istella-X validation set.

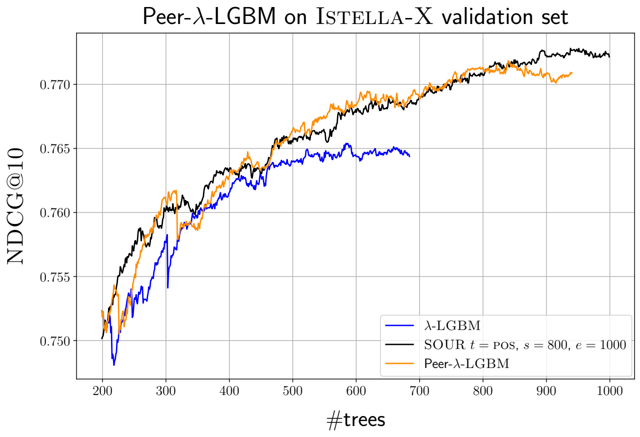

Figure 5: Performance of Peer-λ-LGBM on Istella-X validation set.

4.5 Does outlier removal perform better than data augmentation?

PeerRank [23] is a robust training algorithm that uses a data augmentation mechanism. The rationale is to inject the training dataset with synthetic outlier instances so that the trained algorithm can learn to rank them poorly. We thus provide Peer-λ-LGBM, our implementation of PeerRank on top of λ-LGBM.

Fig. 5 shows that the Peer-λ-LGBM outperforms λ-LGBM and its behaviour is very similar to that of SOUR. Indeed, SOUR is still able to provide some additional improvement, and in the next section we provide comprehensive results on the three datasets adopted.

4.6 SOUR performance evaluation

Tab. 3 reports the performance in terms of NDCG@5 and NDCG@10 of SOUR and its competitors. Performance is measured on the datasets' test sets. First, we highlight that the proposed SOUR outperforms other algorithms on every dataset and for both metrics. The the smallest improvement is for the Yahoo! dataset, while interesting gains are instead visible for both Istella-X and MSLR-30K. The improvement, even if not large, is statistically significant in most cases. It is well known how apparently small improvements in NDCG are indeed very relevant [5]. This confirms that the proposed SOUR algorithm is effective, and that further investigation is worthwhile.

There is not a best choice between negative and positive outliers to remove and for the starting tree s and, therefore, we tune both t and s on a validation set. On average, looking at the last few hundred trees is the best option for SOUR. We highlight the small number of documents removed by SOUR, e.g., only 1% of the non relevant documents is dropped from the MSLR-30K dataset. In fact, the number of removed documents is much smaller than Iso-λ-LGBM. Even in this case, Yahoo! is an exception. In this regard, we highlight the peculiar nature of the Yahoo! dataset, which has more positive than negative documents, and it has a very limited number of documents per query and, therefore, removing a fee of such small set of documents may not provide large benefits.

Finally, we include in our analysis a variant, namely *last*-SOUR, where the ourliers are selected by only looking at the ranking computed by the whole λ-LGBM forest ($s = e =$ *last tree*). Its performance confirms that SOUR performs well when it looks at *consistent* outliers across different trees of the forest. In this regard, the proposed outlier detection mechanism is novel as it exploits the iterative nature of gradient boosting decision trees.

Regarding competitors, we highlight that Peer-λ-LGBM is always more accurate than Iso-λ-LGBM. We leave as future work the investigation of the possibility of embedding the PeerRank cost function into SOUR. As a side note, we highlight the impact of gradient normalization in λ-LGBM that is often neglected.

5 SOUR IN-DEPTH ANALYSIS

In this section we analyze with further details the behaviour of SOUR. We first try to understand what provides the performance improvement of our approach, then we justify experimentally our choices by contrasting against alternative variants. The analysis adopts the experimental setting described above but it is limited on the mid-sized MSLR-30K dataset.

5.1 On which queries SOUR performs best?

In Tab. 4 we provide a breakdown of the effectiveness of SOUR and its competitors over two different set of queries. The set Q_o includes those queries with at least one consistent outlier document, while Q_c contains the remaining "clean" queries not having any outlier. For all algorithms we use the best configuration reported in Tab. 3, except for Iso-λ-LGBM for which we use the best model trained with $t =$ neg that has 725 trees. Note that $t =$ neg for all methods. We compute the consistent outliers on the basis of the λ-LGBM model built during the first step of SOUR and we vary s and keep fixed $e = 1000$. The goal is to measure how the different algorithms improve or worsen the score of λ-LGBM on the two different kinds of queries. Intuitively we expect an improvement on the queries with outliers Q_o and negligible changes on the remaining queries Q_c.

Table 3: Comparison of the proposed SOUR against the state of the art. Statistically significant differences w.r.t. SOUR according to Fisher's randomization test [9] (with a one-sided p-value) are marked with * ($p = 0.05$) and ** ($p = 0.01$).

model	#trees	NDCG@5	#trees	NDCG@10	model	#trees	s	t	$\lvert O \rvert$	NDCG@5	#trees	s	t	$\lvert O \rvert$	NDCG@10
dataset: ISTELLA-X															
λ-MART	206	0.6711^{**}	223	0.7272^{**}	Iso-λ-LGBM	533		ALL	0.78%	0.7311^{*}	745		ALL	0.02%	0.7739^{**}
λ-LGBM	535	0.7282^{**}	585	0.7723^{**}	*last*-SOUR	670	535	NEG	0.03%	0.7338	811	585	NEG	0.08%	0.7769
Peer-λ-LGBM	883	0.7332	841	0.7749^{**}	**SOUR**	617	0	POS	0.01%	**0.7362**	939	800	POS	0.01%	**0.7804**
dataset: MSLR-30K															
λ-MART	998	0.4997^{**}	979	0.5216^{**}	Iso-λ-LGBM	771		ALL	6.81%	0.5045^{**}	725		ALL	5.92%	0.5244^{**}
λ-LGBM	802	0.5048^{**}	775	0.5246^{**}	*last*-SOUR	822	802	POS	42.87%	0.5042^{*}	859	775	NEG	1.55%	0.5284^{**}
Peer-λ-LGBM	1000	0.5053^{*}	991	0.5248^{**}	**SOUR**	876	700	NEG	1.01%	**0.5081**	928	700	NEG	1.01%	**0.5304**
dataset: YAHOO!															
λ-MART	975	0.7495^{**}	829	0.7904^{**}	Iso-λ-LGBM	989		NEG	6.08%	0.7488^{**}	648		ALL	24.95%	0.7903^{**}
λ-LGBM	840	0.7524^{*}	701	0.7946^{*}	*last*-SOUR	997	840	NEG	0.76%	0.7528	862	701	NEG	2.73%	0.7935^{**}
Peer-λ-LGBM	987	0.7536	662	0.7950	**SOUR**	999	400	POS	23.26%	**0.7544**	941	300	POS	18.20%	**0.7956**

The first row of Tab. 4 reports the results on the full test set $Q_o \cup Q_c$, which correspond with the best results in Tab. 3. When we vary s to consider different sets of outliers, we observe an interesting behaviour. All algorithms perform better than λ-LGBM on the set Q_o. Intuitively, each model deals correctly with outliers documents and queries containing them are ranked with larger accuracy. On the other hand, the accuracy over Q_c slightly drops for Iso-λ-LGBM and Peer-λ-LGBM. This means that the strategies of these algorithm actually hinder the performance in case of absence of outliers. Indeed, SOUR is the only algorithm that always improves over both Q_o and Q_c.

The last couple of rows include the spacial case where outliers are computed by considering λ-LGBM at the best validation iteration, i.e., $s = e = 775$. Even in this simplified setting, SOUR is the only able to improve on both kind of queries.

We may conclude that SOUR provides a larger improvement on queries with outliers, and it is the only that doesn't lose the ability to rank queries that do not contains outliers, but rather ranks them better than the other models.

Table 4: Performance breakdown over Q_o and Q_c in terms of NDCG@10 on the MSLR-30K test set.

s	query type	$\lvert Q_* \rvert$	λ-LGBM	SOUR	Iso-λ-LGBM	Peer-λ-LGBM
	$Q_o \cup Q_c$	6306	0.5246	0.5304	0.5244	0.5248
0	Q_o	3265	0.4555	0.4623	0.4558	0.4574
	Q_c	3041	0.5988	0.6036	0.5982	0.5970
100	Q_o	4306	0.4647	0.4717	0.4658	0.4669
	Q_c	2000	0.6537	0.6569	0.6508	0.6493
200	Q_o	4504	0.4683	0.4752	0.4697	0.4706
	Q_c	1802	0.6655	0.6685	0.6614	0.6601
300	Q_o	4618	0.4710	0.4779	0.4726	0.4734
	Q_c	1688	0.6712	0.6740	0.6664	0.6651
400	Q_o	4690	0.4725	0.4794	0.4741	0.4749
	Q_c	1616	0.6758	0.6786	0.6706	0.6694
500	Q_o	4742	0.4740	0.4809	0.4755	0.4764
	Q_c	1564	0.6780	0.6806	0.6728	0.6712
600	Q_o	4799	0.4756	0.4822	0.4768	0.4779
	Q_c	1507	0.6809	0.6841	0.6761	0.6740
700	Q_o	4851	0.4764	0.4830	0.4777	0.4787
	Q_c	1455	0.6855	0.6885	0.6805	0.6783
800	Q_o	4906	0.4780	0.4846	0.4792	0.4803
	Q_c	1400	0.6882	0.6912	0.6832	0.6805
900	Q_o	4957	0.4794	0.4859	0.4806	0.4816
	Q_c	1349	0.6908	0.6940	0.6858	0.6831
best	Q_o	5051	0.4809	0.4877	0.4822	0.4833
	Q_c	1255	0.7008	0.7026	0.6945	0.6917

5.2 What is the impact of removing outliers on the trained forest?

To understand the source of the performance improvement provided by SOUR, we further investigate the score assigned to outlier documents and contrast it with positive and negative ones both in λ-LGBM and in SOUR. We use the best parameters of the two algorithms. Tab. 5 reports the average score of the 30 leaves of the forest that are reached by the largest number of negative outliers O^{-} (recall that t = NEG is the best strategy for MSLR-30K). Similarly for the positive documents in the top-k (P_k) and the negative documents (N). We can observe that in λ-LGBM the negative outlier documents receive a score that is closer to that of the positive documents rather than the negative ones (see the last table's row: *avg.* $1 - 50$). In SOUR, the score of negative outlier documents slightly increases, but the gap between positive documents in P_k and negative documents in N is now much larger. This means that SOUR does not improve its ability to detect outliers (which is expected, as SOUR did not see them at training time), but SOUR has a better capability of distinguishing between positive and negative documents.

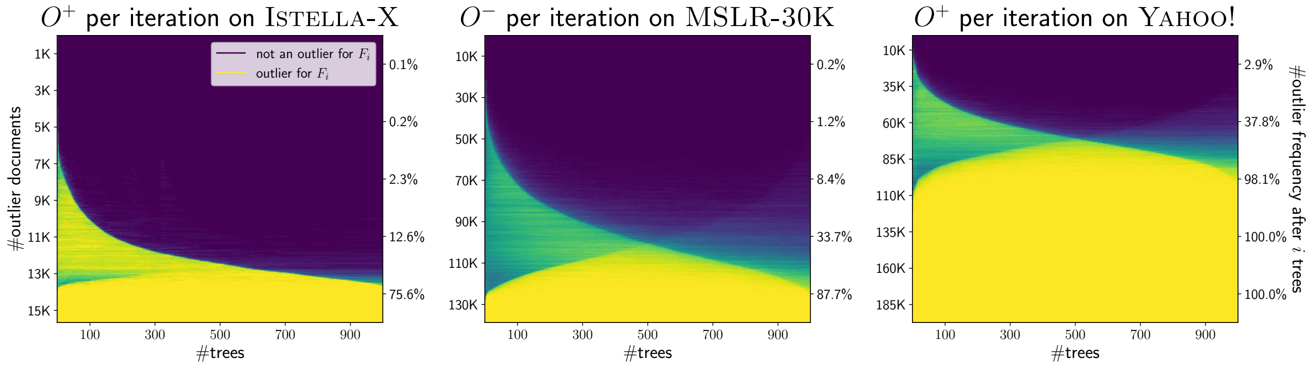

Figure 6: Outliers per iterations found by λ-LGBM on each training set with NDCG@10. Limited to documents that result to be outliers at least once, documents are sorted by their outlier classification frequency.

Table 5: Exit leaves average score on MSLR-30K **test set at different trees of the λ-LGBM and SOUR forests.**

	λ-LGBM			SOUR		
tree	P_k	O^-	N	P_k	O^-	N
1	0.038	0.031	-0.022	0.044	0.036	-0.022
10	0.170	0.111	-0.174	0.177	0.144	-0.183
20	0.269	0.176	-0.293	0.283	0.208	-0.366
30	0.284	0.182	-0.413	0.371	0.277	-0.470
40	0.353	0.274	-0.532	0.392	0.315	-0.622
50	0.425	0.197	-0.645	0.437	0.260	-0.663
avg. 1–50	0.257	0.161	-0.358	0.280	0.207	-0.395

5.3 Do Consistent Outliers perform better than frequent outliers?

In this section we show the results obtained by a p-SOUR, a variant of SOUR. The original SOUR algorithm removes from the dataset all documents that are consistently outliers in an interval $[s, e]$ of the training iterations. Instead, p-SOUR removes documents that are outliers with frequency larger than p%, i.e., at more than p% trees during training. Therefore p-SOUR is a less restrictive variant of SOUR as it does not requires documents to be outliers in all the sub-forests F_i between iteration s and iteration e.

By comparing p-SOUR and SOUR we want to answer the question of whether it is effective to consider documents that are outliers in all trees from the s-th to the e-th, or it is sufficient to detect frequent outliers.

Fig. 6 shows for each training iteration which document is considered an outlier on the basis of the λ-LGBM model. The hourglass-shaped plots show that the number of outlier documents is typically reduced iteration after iteration, except from a bunch of documents that are almost always and, consistently, considered outliers. Fixing the p parameter of p-SOUR corresponds to tuning how many of the documents in the bottom of the plots should be removed from the training.

In Tab. 6 we summarize the performance of p-SOUR in terms of NDCG@10 on the test set of MSLR-30K. When increasing p the performance of p-SOUR also increases up to $p = 90\%$. With small values of p, the strategy removes many negative documents, including those documents that are marked as outliers for a few iterations. The latter are not outliers but rather good negative examples to keep in the dataset. Removing them is deleterious for the model. Beyond $p = 90\%$, results are stable with the best performance when $p = 98\%$.

Table 6: Performance achieved by p-SOUR with t = NEG, on MSLR-30K **test set by varying p. The highest value in bold.**

#trees	NDCG@10	p	#trees	NDCG@10	p
676	0.5169	10%	799	0.5274	91%
869	0.5208	20%	836	0.5268	92%
994	0.5226	30%	817	0.5269	93%
865	0.5219	40%	926	0.5278	94%
754	0.5233	50%	839	0.5268	95%
896	0.5258	60%	859	0.5274	96%
964	0.5271	70%	885	0.5276	97%
750	0.5260	80%	976	**0.5289**	98%
924	0.5279	90%	932	0.5275	99%
		our model	**SOUR**	t = NEG, s = 700	
			928	**0.5304**	

Finally, we can see that the highest NDCG value obtained by p-SOUR in the test set (in bold) is lower than the best performance of SOUR. Thus, although p-SOUR is capable of removing instances that are not useful for training, the SOUR strategy is able to produce more effective models. This highlights that removing documents consistently found as outliers is better than removing frequent outlier documents.

5.4 Does removing outliers perform better than exploiting outliers?

So far we observed that consistent outliers are hard to be ranked correctly and therefore we decided to remove them from the training. The following question arises: could we resort to *curriculum learning* [1] to properly exploit those hard instances? According

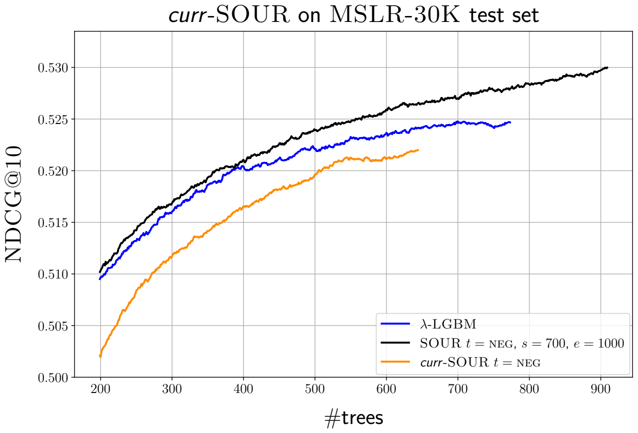

Figure 7: Performance of *curr*-SOUR on MSLR-30K **test set.**

to curriculum learning, a delayed injection, in the training set, of difficult documents allows a more accurate training of the model. In our case, we experiment a delayed injection of outliers.

To answer this question we subdivide the outliers found in different forest stages, i.e., from 0 to 1000, in 5 sets: outliers that appear only in the intervals $[0, 50]$, $[0, 200]$, $[0, 500]$, $[0, 800]$ and $[0, 900]$. We inject each set, in the training set, when the number of boosting iterations is greater than the right end of the interval. We call this strategy *curr*-SOUR.

However, in Fig. 7 the results show how this idea does not fit well in this scenario and that the model effectiveness is far from SOUR and close to λ-LGBM only on the last iterations. This seems that we can not treat these outliers as instances of different difficulty, but they must be removed entirely from the beginning of the training phase.

5.5 Is SOUR robust to the amount of outliers?

In this section we analyze the behavior of SOUR and λ-LGBM on datasets with different amount of outliers. The analysis focuses on testing SOUR capability to detect outliers during the training phase and how the removal strategy effects the prediction phase, in particular compared to a baseline λ-LGBM model trained on the whole synthetic dataset.

We use MSLR-30K training and validation set and with probability $n = \{0.05, 0.075, 0.1\}$ we flip a document with relevance label 0 to 4. Let $\mathcal{D}^n_{\text{TRAIN}}$ and $\mathcal{D}^n_{\text{VALID}}$ be respectively the synthetic training set and synthetic validation set, we perform training on $\mathcal{D}^n_{\text{TRAIN}}$ and early stopping on $\mathcal{D}^n_{\text{VALID}}$. Model evaluation is done on the original test set and the results are summarized in Fig. 8. Since we introduce outliers with label greater than 0, we train SOUR with t = POS. In order to study the behavior of SOUR on different search intervals $[s, e]$, we set $e = 1000$ and vary the value of s.

It can be clearly seen that as the noise rate increases, the gap between the models trained with SOUR and λ-LGBM increases. This phenomenon is present since the first boosting interactions. All models trained with SOUR have superior performance with respect to the baseline, in particular SOUR with t = POS, $s = 0$ and $e = 1000$. In this setting, SOUR removes documents that are always detected as outliers. This suggests that most synthetic outliers are always considered outliers from the very first iterations and that using smaller $[s, e]$ intervals leads to the removal of good positive examples that are being pushed down to the list, for a sufficient number of iterations, due to synthetic outliers. For the sake of truth, SOUR in not able to detect and remove all the synthetic outliers. In fact, as the noise rate increase, both SOUR and λ-LGBM lose part of their effectiveness.

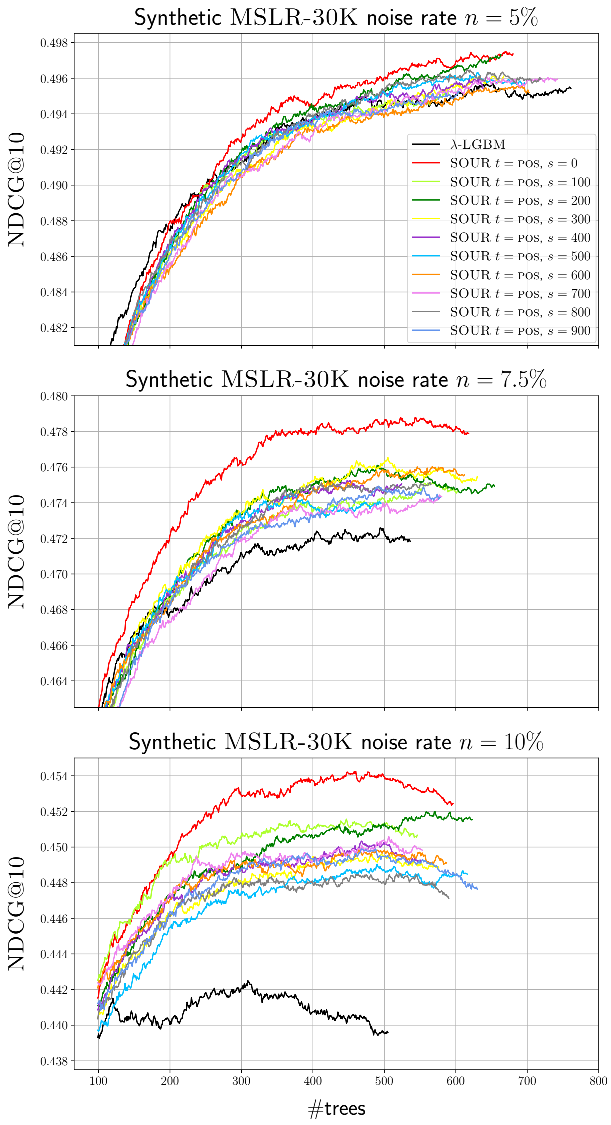

Figure 8: SOUR and λ-LGBM effectiveness on MSLR-30K **test set, by varying noise rate** $n = \{0.05, 0.075, 0.1\}$**.**

6 CONCLUSION

We developed a new sample selection strategy for outliers detection based on persistent errors in multiple iterations of the learning algorithm. We have shown how the removal of these outliers can generate better performing models in terms of NDCG. Through extensive experiments we have shown that the proposed strategy has a statistically significant increase in performance compared to the state of the art.

A number of extension of SOUR are clearly possible. One could use the model scores, in addition to the ranks, and the instances' features to detect outliers. Additionally, it would be possible to make the outlier filtering a dynamic process conducted while building, tree after tree, a ranking model. We leave these analysis to future work. Finally, this result opens the way toward employing this new method to other models such as Artificial Neural Networks.

REFERENCES

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009 (ACM International Conference Proceeding Series, Vol. 382)*, Andrea Pohoreckyj Danyluk, Léon Bottou, and Michael L. Littman (Eds.). ACM, 41–48. https://doi.org/10.1145/1553374.1553380

[2] Sebastian Bruch. 2021. An Alternative Cross Entropy Loss for Learning-to-Rank. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia (Eds.). ACM / IW3C2, 118–126. https://doi.org/10.1145/3442381.3449794

[3] Vitor R Carvalho, Jonathan L Elsas, William W Cohen, and Jaime G Carbonell. 2008. A meta-learning approach for robust rank learning. In *SIGIR 2008 workshop on learning to rank for information retrieval*, Vol. 1.

[4] Olivier Chapelle and Yi Chang. 2011. Yahoo! Learning to Rank Challenge Overview. In *Proceedings of the Yahoo! Learning to Rank Challenge, held at ICML 2010, Haifa, Israel, June 25, 2010 (JMLR Proceedings, Vol. 14)*, Olivier Chapelle, Yi Chang, and Tie-Yan Liu (Eds.). JMLR.org, 1–24. http://proceedings.mlr.press/v14/chapelle11a.html

[5] Olivier Chapelle, Thorsten Joachims, Filip Radlinski, and Yisong Yue. 2012. Large-scale validation and analysis of interleaved search evaluation. *ACM Transactions on Information Systems (TOIS)* 30, 1 (2012), 6.

[6] Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. 2009. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*, David Wai-Lok Cheung, Il-Yeol Song, Wesley W. Chu, Xiaohua Hu, and Jimmy Lin (Eds.). ACM, 621–630. https://doi.org/10.1145/1645953.1646033

[7] Wenkui Ding, Xiubo Geng, and Xudong Zhang. 2015. Learning to Rank from Noisy Data. *ACM Trans. Intell. Syst. Technol.* 7, 1 (2015), 1:1–1:21. https://doi.org/10.1145/2576230

[8] Lei Feng, Senlin Shu, Zhuoyi Lin, Fengmao Lv, Li Li, and Bo An. 2020. Can Cross Entropy Loss Be Robust to Label Noise?. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, Christian Bessiere (Ed.). ijcai.org, 2206–2212. https://doi.org/10.24963/ijcai.2020/305

[9] R.A. Fisher. 1935. *The design of experiments. 1935.* Oliver and Boyd, Edinburgh.

[10] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6572

[11] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor W. Tsang, and Masashi Sugiyama. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 8536–8546. https://proceedings.neurips.cc/paper/2018/hash/a19744e268754fb0148b017647355b7b-Abstract.html

[12] Kalervo Järvelin and Jaana Kekäläinen. 2000. IR evaluation methods for retrieving highly relevant documents. In *SIGIR 2000: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 24-28, 2000, Athens, Greece*, Emmanuel J. Yannakoudakis, Nicholas J. Belkin, Peter Ingwersen, and Mun-Kew Leong (Eds.). ACM, 41–48. https://doi.org/10.1145/345508.345545

[13] George H. John. 1995. Robust Decision Trees: Removing Outliers from Databases. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95), Montreal, Canada, August 20-21, 1995*, Usama M. Fayyad and Ramasamy Uthurusamy (Eds.). AAAI Press, 174–179. http://www.aaai.org/Library/KDD/1995/kdd95-044.php

[14] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 3146–3154. https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html

[15] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation Forest. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy.* IEEE Computer Society, 413–422. https://doi.org/10.1109/ICDM.2008.17

[16] Yang Liu and Hongyi Guo. 2020. Peer Loss Functions: Learning from Noisy Labels without Knowing Noise Rates. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event (Proceedings of Machine Learning Research, Vol. 119).* PMLR, 6226–6236. http://proceedings.mlr.press/v119/liu20e.html

[17] Claudio Lucchese, Franco Maria Nardini, Raffaele Perego, Salvatore Orlando, and Salvatore Trani. 2018. Selective Gradient Boosting for Effective Learning to Rank. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, Kevyn Collins-Thompson, Qiaozhu Mei, Brian D. Davison, Yiqun Liu, and Emine Yilmaz (Eds.). ACM, 155–164. https://doi.org/10.1145/3209978.3210048

[18] Duc Tam Nguyen, Chaithanya Kumar Mummadi, Thi-Phuong-Nhung Ngo, Thi Hoai Phuong Nguyen, Laura Beggel, and Thomas Brox. 2020. SELF: Learning to Filter Noisy Labels with Self-Ensembling. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020.* OpenReview.net. https://openreview.net/forum?id=HkgsPhNYPS

[19] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 Datasets. *CoRR* abs/1306.2597 (2013). http://arxiv.org/abs/1306.2597

[20] J. Ross Quinlan. 1993. *C4.5: Programs for Machine Learning.* Morgan Kaufmann.

[21] Scott E. Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. 2015. Training Deep Neural Networks on Noisy Labels with Bootstrapping. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6596

[22] Qiang Wu, Christopher J. C. Burges, Krysta M. Svore, and Jianfeng Gao. 2010. Adapting boosting for information retrieval measures. *Inf. Retr.* 13, 3 (2010), 254–270. https://doi.org/10.1007/s10791-009-9112-1

[23] Xin Wu, Qing Liu, Jiarui Qin, and Yong Yu. 2022. PeerRank: Robust Learning to Rank With Peer Loss Over Noisy Labels. *IEEE Access* 10 (2022), 6830–6841. https://doi.org/10.1109/ACCESS.2022.3142096

[24] Zhilu Zhang and Mert R. Sabuncu. 2018. Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 8792–8802. https://proceedings.neurips.cc/paper/2018/hash/f2925f97bc13ad2852a7a551802feea0-Abstract.html