Selective Gradient Boosting for Effective Learning to Rank

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

Franco Maria Nardini
ISTI-CNR, Pisa, Italy
f.nardini@isti.cnr.it

Raffaele Perego
ISTI-CNR, Pisa, Italy
r.perego@isti.cnr.it

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

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

ABSTRACT

Learning an effective ranking function from a large number of query-document examples is a challenging task. Indeed, training sets where queries are associated with a few relevant documents and a large number of irrelevant ones are required to model real scenarios of Web search production systems, where a query can possibly retrieve thousands of matching documents, but only a few of them are actually relevant. In this paper, we propose Selective Gradient Boosting (SelGB), an algorithm addressing the Learning-to-Rank task by focusing on those irrelevant documents that are most likely to be mis-ranked, thus severely hindering the quality of the learned model. SelGB exploits a novel technique minimizing the mis-ranking risk, i.e., the probability that two randomly drawn instances are ranked incorrectly, within a gradient boosting process that iteratively generates an additive ensemble of decision trees. Specifically, at every iteration and on a per query basis, SelGB *selectively* chooses among the training instances a small sample of negative examples enhancing the discriminative power of the learned model. Reproducible and comprehensive experiments conducted on a publicly available dataset show that SelGB exploits the diversity and variety of the negative examples selected to train tree ensembles that outperform models generated by state-of-the-art algorithms by achieving improvements of NDCG@10 up to 3.2%.

CCS CONCEPTS

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

KEYWORDS

Learning to Rank, Multiple Additive Regression Trees, Boosting

ACM Reference Format:
Claudio Lucchese, Franco Maria Nardini, Raffaele Perego, Salvatore Orlando, and Salvatore Trani. 2018. Selective Gradient Boosting for Effective Learning to Rank. In *SIGIR '18: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, July 8–12, 2018, Ann Arbor, MI, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209978.3210048

SIGIR '18, July 8–12, 2018, Ann Arbor, MI, USA

ACM ISBN 978-1-4503-5657-2/18/07…$15.00
https://doi.org/10.1145/3209978.3210048

1 INTRODUCTION

In the last ten years several effective machine learning solutions explicitly tailored for ranking problems have been designed, giving rise to a new research field called Learning-to-Rank (LtR). Web search is one of the most important applications of these techniques, as complex ranking models learned from huge gold standard datasets are necessarily adopted to effectively identify the documents that are relevant for a given user query among the billions of documents indexed. Given a gold standard dataset where each query-document example is modeled by hundreds of features and a label assessing the relevance of the document for the query, a LtR algorithm learns how to exploit the features to provide a query-document scoring model that optimizes a metric of ranking effectiveness, such as NDCG, MAP, ERR, etc. [19].

In a large-scale Web search system a user query can match thousands or millions of documents, but only a few of them are actually relevant for the user [23]. Therefore, learning effective ranking functions in this scenario requires large gold standard datasets where each training query is associated with a few relevant documents (positive examples) and a large amount of irrelevant ones (negative examples). Indeed, several studies confirm that a number of examples in the order of thousands per query is required [4, 5, 18].

Research in this field has focused on designing efficient [6] and effective algorithms that improve the state of the art, or on engineering new classes of features allowing to better model the relevance of a document to a query. Less effort has been spent in understanding how to deal with the unbalanced classes of positive and negative examples in the gold standard so as to maximize the effectiveness and robustness of the learned ranking model. This aspect has not been fully investigated mainly because publicly available datasets contain a relatively low number of negative examples per query, thus preventing in-depth studies on the impact of class imbalance on LtR algorithms.

To investigate the issue of class imbalance in real-world LtR datasets, in this paper we contribute and study a new dataset with about 2.7K examples per query on average. We first investigate how the volume of negative examples impacts on a state-of-the-art LtR algorithm such as λ-Mart [21]. Experimental results confirm that a large number of negative examples is required to train effective models. We also show that λ-Mart reaches a plateau, where increasing class imbalance neither harms or improves the accuracy achieved. However we observe that not all the negative examples are equally important for the training process, and that it is hard for an algorithm do properly identify and exploit the most informative negative instances. We thus present a novel LtR algorithm, named

Selective Gradient Boosting (SelGB) focusing during learning on the negative examples that are likely to be the most useful to improve the model learned so far. To this purpose, we introduce a novel *negative selection* phase within a gradient boosting learning process. Specifically, SelGB is designed as a variant of the λ-Mart algorithm that at each iteration limits the training set used to grow the tree ensemble to all the positive examples and to a sample of negative ones. The negative examples chosen at each step are the most informative ones, those that are most useful to reduce the misranking risk, i.e., the probability that a method ranks two randomly drawn instances incorrectly. Results of the exhaustive experimental assessment conducted confirm that SelGB is able to train models that significantly improve NDCG@10 over the ones generated by the reference λ-Mart algorithm.

In summary, we improve the state of the art in the LtR field with the following contributions:

- we propose SelGB, a new gradient boosting algorithm designed as a variant of λ-Mart that iteratively selects the most "informative" negative examples from the golden standard dataset. The proposed technique allows the SelGB algorithms to focus during training on those negative examples that are likely to be the most useful to reduce the ranking risk of the scoring model learned so far and adjust the model correspondingly. We provide a comprehensive experimental comparison showing that SelGB outperforms the reference λ-Mart algorithm by up to +3.2% in terms of NDCG@10.
- we release the SelGB source code and a new public LtR dataset to foster the research in this field and to allow the reproducibility of our results. The dataset is made up of 26,791,447 query-document pairs, produced starting from 10,000 queries sampled from a query log of a real-world search engine. On average, the dataset contains 2,679 documents per query. To the best of our knowledge this is the largest public LtR dataset ever released, in terms of number of documents per query.

The rest of the paper is structured as follow: Section 2 discusses the related work while Section 3 introduces the notation and the preliminaries needed to present the Selective Gradient Boosting algorithm in Section 4. We provide a comprehensive evaluation of Selective Gradient Boosting against state-of-the-art competitors in Section 5. Finally, Section 6 concludes the work and outlines future work.

2 RELATED WORK

Research in the LtR field in the last years mainly focused on developing effective LtR algorithms [3, 14–16] and on extracting and engineering relevant features from query-document pairs. A relatively lower attention was reserved to study how to choose queries and documents to include in LtR gold standard datasets, or the effect of these choices on the ability of LtR algorithms to learn effective and efficient scoring models. Yilmaz and Robertson observe that the number of judgments in the training set directly affects the quality of the learned system [22]. Given that collecting relevance judgments from human assessors to build gold standard datasets is expensive, the main problem is how to well distribute this judgment effort. Authors thus investigate the trade-off between the number of queries and the number of judgments per query when building training sets. In particular, they show that training sets with more queries but less judgments per query are more cost effective than training sets with less queries but more judgments per query.

Aslam *et al.* investigate different document selection methodologies, i.e., depth-k pooling, sampling (infAP, statAP), active-learning (MTC), and on-line heuristics (hedge) [1]. The proposed techniques result in gold standard datasets characterized by different properties. They show that infAP, statAP and depth-k pooling are better than hedge and the LETOR method (depth-k pooling using BM25) for building efficient and effective LtR collections. The study conducted deals with both i) the proportion of positive and negative examples, and ii) the similarity between positive and negative examples in the datasets. Results confirm that both characteristics highly affect the quality of the LtR collections, with the latter having more impact. As a side result, the authors observe that some LtR algorithms, RankNet and λ-Mart, are more robust to document selection methodologies than other, i.e., Regression, RankBoost, and Ranking SVM.

In this paper we focus on selecting dynamically, i.e., at training time, samples of negative examples improving the accuracy of the scoring model learned. Conversely, the work by Aslam *et al.* investigate *a priori* document selection methodologies that do not consider the document class. Moreover, Aslam *et al.* apply document selection methodologies on depth-100 pools from TREC 6,7, and 8 adhoc tracks, i.e., a collection of 150 queries in total. In this paper, we evaluate our proposal on a new dataset with 10,000 queries and about 27M examples specifically built for investigating this problem.

In a later contribution, Kanoulas *et al.* propose a large-scale study on the effect of label distribution in gold standard datasets across the different grades of relevance [11]. The authors propose a methodology to generate a large number of datasets with different label distributions. The datasets are then used to fit different ranking models learned by using three LtR algorithms. The study concludes that the distribution in the training set of relevance grades is an important factor for the effectiveness of LtR models. Qualitative advises are provided regarding the construction of a gold standard: distributions with a balance between the number of documents in the extreme grades should be favored, as the middle relevance grades play less important role than the extreme ones.

Ibrahim and Carman investigate the imbalanced nature of LtR training sets. They observe that these datasets contain very few positive examples as compared to the number of negative ones [9]. The authors study how many negative examples are needed in order to learn effective ranking functions. They exploits random and deterministic under-sampling techniques to reduce the number of negative documents. The reduction of the size of the dataset decreases the training time, which is an important factor in large scale search environments. The study shows that under-sampling techniques can be successfully exploited for large-scale LtR tasks to reduce training time with negligible effect on effectiveness. Lucchese *et al.* also contribute in the same direction, by investigating a new technique to sample documents so to improve both efficiency and effectiveness of LtR models [17]. The improved efficiency comes from a reduced size of the sampled dataset, as

the ranking algorithm is trained on a smaller number of query-document pairs. Experiments on a real-world LtR dataset show that an effective sampling technique improves the effectiveness of the resulting model by also filtering out noise and reducing redundancy of training examples.

Our proposal is different from the work by Ibrahim and Carman as we do not study sampling techniques aimed at reducing the number of examples without hindering effectiveness. Conversely, our SelGB algorithm focuses the learning process on the "most informative" negative examples to maximize the effectiveness of the learned model. Moreover, the selection of negative examples is performed dynamically, i.e., during the iterative growing of the scoring model and not *a priori*, (i.e., once, before the learning process starts) as described in both the works.

Long *et al.* address the selection of the examples which minimize the expected DCG loss over a training set in an Active Learning framework [13]. Authors motivate the task with the need to reduce the cost associated with manual labeling of documents. Some other works, such as Yu [24] and Donmez *et al.* [7], also try to find the examples which can improve ranking accuracy if added to the training set. These papers aims at improving the quality of the training set while keeping its size small and do not distinguish between positive and negative examples. Differently, we explicitly deal with dataset imbalance and we exploit the "most informative" negative examples in a large-scale LtR scenario.

Another approach that is related to our proposal is the Gradient-based One-Sided Sampling (GOSS) technique employed within LightGBM [12]. At each iteration of the boosting process, GOSS considers only a subset of the training examples: those with the largest gradient and a random sample of the remaining instances, representative of the whole dataset. This strategy is howewer finalized at reducing the computational cost of the training. Indeed, the authors report that GOSS provides also a negligible improvement of ≈ 0.003 in NDCG@10. Since GOSS does not provide any statistically significant improvement over the reference algorithm we do not include it among the baselines in the experimental analysis of this work.

Another important related contribution is *Stochastic Gradient Boosting* by Friedman [8]. Friedman observes that a randomization step within gradient boosting allows to increase robustness against over-fitting and to reduce the variance of the model. In particular, at each iteration [8] proposes to fit a weak learner on a random sample of the training dataset. The solution proposed by Friedman is similar to ours because the sampling is repeated at each iteration and is part of a gradient boosting process, thus favouring the generalization power of the model without compromising its effectiveness. The main difference with our approach concerns the selection of the sample dataset. SelGB selects the negative examples that are likely to be mis-classified by the scoring model learned so far in order to learn from them. *Stochastic Gradient Boosting* selects instead the sample at random without exploiting any knowledge of the model learned so far.

3 NOTATION AND PRELIMINARIES

Let $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{|\mathcal{D}|}, y_{|\mathcal{D}|})\}$ be a gold standard training set, where $\mathbf{x}_i \in \mathbb{R}^{|\mathcal{F}|}$ is the real valued vector of features in $\mathcal{F} = \{f_1, f_2, \ldots\}$, and $y_i \in \mathbb{R}$ the target label. Gradient boosting is a greedy stage-wise technique that aims at learning a function $F(\mathbf{x})$ that minimizes the prediction loss $L(y_i, F(\mathbf{x}_i))$ averaged over $\mathbf{x}_i \in \mathcal{D}$.

For this work function $F(\mathbf{x})$ is an additive ensemble of regression trees (*weak learners*), denoted by $\mathcal{E} = \{t_1, \ldots, t_{|\mathcal{E}|}\}$, where each tree t_i tries to approximate the negative gradient direction. After $m-1$ trees, the gradient g_m of the current function $F_{m-1}(\mathbf{x})$ is defined at each data instance $\mathbf{x}_i$ as:

$$g_m(\mathbf{x}_i) = \left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)} \right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}$$

The negative gradient $-g_m$ is approximated by fitting a regression tree t_m on the pairs $\{\mathbf{x}_i, -g_m(\mathbf{x}_i)\}$, and it is then used to update $F_{m-1}(\mathbf{x})$. Let $h_m(\mathbf{x})$ be the prediction given by tree t_m on instance $\mathbf{x}$. Thus we have:

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu \cdot h_m(\mathbf{x})$$

where ν is the *shrinkage* factor (or *learning rate*) which acts as a regularization factor by shrinking the size of the minimization step along the steepest descent direction.

When the loss L is Mean Squared Error (MSE), i.e., $L(y_i, F(\mathbf{x}_i)) = \frac{1}{2}(y_i - F(\mathbf{x}_i))^2$, the negative gradients can be easily computed as $-g_m(\mathbf{x}_i) = y_i - F_{m-1}(\mathbf{x}_i)$. This is usually denoted with r_i and named *pseudo-response*.

Without loss of generality, hereinafter we refer to a Web search scenario where the goal is to learn a scoring function to rank Web documents in response to a user query. In such LtR framework gradient boosting algorithms are considered the state of the art and the gold standard $\mathcal{D}$ is indeed made of many *ranked lists*. For each assessed query q, $\mathcal{D}$ includes in fact multiple query-document pairs representing both positive (relevant) and negative (not relevant) examples. In turn, each query-document pair represented by $\mathbf{x}_i \in \mathbb{R}^{|\mathcal{F}|}$ is labeled with a relevance judgment y_i (usually a positive integer in the range $[0, 4]$ where 4 means *highly relevant* and 0 *not relevant*). Such relevance labels induce a per-query partial order corresponding to the *ideal ranking* we aim to learn from the gold standard. We denote by $\mathcal{D}^-$ the *negative* examples in $\mathcal{D}$, and by $\mathcal{D}^+$ the *positive* instances $\mathbf{x}_i \in \mathcal{D}$ such that $y_i > 0$. It is worth remarking that in the LtR scenario, the amount of negative examples is typically much larger than the number of positive ones, i.e., $|\mathcal{D}^-| \gg |\mathcal{D}^+|$. Hereinafter, we will call *query list size* the cardinality of the list of training examples referred to a query q, where the list in $\mathcal{D}$ comprises both relevant and not relevant examples.

The Multiple Additive Regression Trees (MART) algorithm is an implementation of the above framework: a forest of regression trees is grown through gradient boosting by optimizing MSE. In the LtR scenario, rank-based quality measures are used to evaluate the quality of a ranked document list. Hereinafter, we use NDCG@k as the reference quality measure to be maximized. Given a ranked document list, NDCG@k is a normalized measure that only weights the top-k ranked documents according to their relevance and discounts their contribution according to their rank position. MART can be used as a basic strategy for the LtR problem, but minimizing MSE does not provide guarantees on NDCG optimization.

Algorithm 1 Selective Gradient Boosting.

1: **function** Selective Gradient Boosting($\mathcal{D}, N, n, p$)
   input:
2: $\mathcal{D}$: *training dataset*
3: N : *ensemble size*
4: n : *# of iterations between consecutive sampling steps*
5: p : *% of irrelevant samples kept in $\mathcal{D}$ at each sampling step*
   output:
6: $\mathcal{E}$: *trained ensemble*
7:
8: $\mathcal{E} \leftarrow \emptyset$ ▷ the current ensemble model
9: $\mathcal{D}^* \leftarrow \mathcal{D}$
10: **for** $m = 1$ to N **do**
11: **if** $(m \mod n) = 0$ **then** ▷ Every n iterations
12: $\mathcal{D}^* \leftarrow$ Sel_Sampl($\mathcal{D}$, $\mathcal{E}$, p)
13: $\{\lambda_i\} \leftarrow \lambda$*-gradients for each* $\mathbf{x}_i \in \mathcal{D}^*$
14: $\mathcal{R}^* = \{(\mathbf{x}_i, \lambda_i)\}$, *for all* $\mathbf{x}_i$ *occurring in* $\mathcal{D}^*$
15: $t_m \leftarrow$ *fit a regression tree to* $\mathcal{R}^*$
16: $\mathcal{E} \leftarrow \mathcal{E} \cup t_m$
17: **return** $\mathcal{E}$

The state-of-the-art LtR algorithm is λ-Mart [2], a variant of MART aimed at optimizing rank-based quality measures. Indeed, a function such as NDCG@k is not differentiable, and thus it cannot be used as the loss function of a gradient boosting framework. λ-Mart thus introduces a smooth approximation of the gradient, which is called λ-gradient. For each training instance $\mathbf{x}_i$, λ-Mart estimates the benefit of increasing or decreasing the currently predicted score $F_m(\mathbf{x}_i)$, by computing the change of NDCG@k value occurring when a variation of $F_m(\mathbf{x}_i)$ causes a change in the rank position. This estimate, denoted by λ_i, is used in place of the gradient $g_m(\mathbf{x}_i)$.

We remark that, at learning time, MART processes training examples independently of one another. In fact, for optimizing MSE knowing whether two examples are referred to the same query or not is totally irrelevant. The λ-Mart is instead a listwise algorithm, which evaluates the whole ranked list of training examples referred to a given query in order to estimate their gradients.

Finally, we note that gradient boosting algorithms require some kind of regularization to avoid over-fitting. Besides the shrinkage parameter (ν) that reduces the variance of each tree added to the ensemble, we can also control the complexity of each additive tree t_m by limiting, for example, the number of levels or leaves of the trees. Another regularization technique, which is related to the approach discussed in the next Section 4, consists in subsampling of the training dataset. For example, at each iteration of gradient boosting, Stochastic Gradient Boosting [8] fits a tree on a random sample of the training dataset, thus introducing randomization to increase robustness and avoid over-fitting. As a side effect, this approach reduces the computational cost of the fitting phase due to the decreased amount of training data used.

4 SELECTIVE GRADIENT BOOSTING

In this section we introduce Selective Gradient Boosting (SelGB). The pseudo-code in Algorithm 1 shows that SelGB is a gradient boosting algorithm similar to λ-Mart.

The core of the algorithm is the novel function Sel_Sampl($\mathcal{D}$, $\mathcal{E}$, p) (line 12), which selects a subset $\mathcal{D}^*$ of the original dataset $\mathcal{D}$ to be used during the fitting of the next regression tree. In particular, SelGB maintains in $\mathcal{D}^*$ all the relevant examples $\mathcal{D}^+$, whereas keeps in $\mathcal{D}^*$ only those irrelevant instances $\mathcal{D}^-$ that are scored the highest by the model $\mathcal{E}$ learned so far. Specifically, for every query q occurring in $\mathcal{D}$, all the *positive* examples $(\mathbf{x}_i, y_i)$, $y_i > 0$ are inserted into $\mathcal{D}^*$, while the *negative* instances $(\mathbf{x}_j, y_j)$, $y_j = 0$ are scored with the current model $\mathcal{E}$ and sorted according to the estimated score $F_m(x_j)$. Only the fraction $p\%$ of the top-ranked negative instances $\mathbf{x}_j$ are then inserted into $\mathcal{D}^*$. This data *selection* process leads to a pruned training set $\mathcal{D}^*$ of cardinality $|\mathcal{D}^*| = |\mathcal{D}^+| + p\% \cdot |\mathcal{D}^-|$. Note that the subset $\mathcal{D}^*$ is used for the next n iterations, until a new $\mathcal{D}^*$ is selected on the basis of the updated model (line 11).

Unlike Stochastic Gradient Boosting, which randomly samples $\mathcal{D}$ to increase the robustness to over-fitting and to reduce variance, our approach, which selectively chooses a small sample of the irrelevant instances to be kept in the training set, aims to minimize the mis-ranking risk. Due to the characteristics of the NDCG metric, we need to be very accurate in discriminating the few positive instances that must be pushed in the top positions of the scored lists, from the plenty of negative instances present in $\mathcal{D}$. In the context of ranking, this means to discriminate between the relevant documents and the highest scored irrelevant documents for any query in the dataset. Indeed, the negative instances with the highest scores are exactly those being more likely to be ranked above relevant instances, thus severely hindering the ranking quality measured by NDCG. On the other hand, the low-scored negative instances can hardly affect remarkably NDCG, and can be safely discarded.

In the following, we discuss in more detail the pseudo-code of SelGB (see Figure 1). SelGB builds iteratively $\mathcal{E}$ until a maximum number of trees N, which is another hyper-parameter of our learning algorithm. At each iteration, first we compute the λ-gradients (line 13) of $\mathcal{D}^*$. Note that at the first iteration the pseudo-responses λ_i just correspond to the original y_i labeling the instances $\mathbf{x}_i$ in $\mathcal{D}$. These λ-gradients are then used to build the training set $\mathcal{R}^*$ (line 14), on which we fit the next tree t_m (line 14) used to grow the tree ensemble $\mathcal{E}$.

Finally, although not shown in the pseudo-code for sake of simplicity, SelGB can early stop the growth of $\mathcal{E}$, by using a validation set. The stop occurs when the NDCG@k measured over the validation set does not improve for a fixed number of iterations. Moreover, at the first iteration $\mathcal{D}$ is not yet pruned and $\mathcal{D}^*$ is initialized with a copy of $\mathcal{D}$ (line 9).

5 EXPERIMENTS

The experimental scenario we focus on is a two-stage query processing architecture common in large-scale Web IR systems [4, 5, 23]. In this scenario a few relevant documents have to be selected from a huge and noisy collection of Web documents. To this end, a first query processing stage retrieves from the index a large number of candidate documents matching the user query. These candidate documents are then re-ordered by a subsequent, complex and accurate ranking stage. We target in particular the process of learning an effective ranking function for such a second stage. Experimental

Table 1: Datasets properties.

Properties	Istella-X^{5k}	Istella-X$^{2.5k}$	Istella-X^{1k}	Istella-X^{500}	Istella-X^{100}
# queries	10,000				
# features	220				
# query-doc pairs	26,791,447	15,778,399	7,363,902	3,995,063	906,782
max # docs/query	5,000	2,500	1,000	500	100
avg. # docs/query	2,679	1,578	736	400	91
# pos. query-doc pairs	46,371 (0.17%)	46,371 (0.29%)	46,371 (0.63%)	46,371 (1.16%)	46,371 (5.11%)
# neg. query-doc pairs	26,745,076 (99.83%)	15,732,028 (99.71%)	7,317,531 (99.37%)	3,948,692 (98,84%)	860,411 (94,89%)

results show that, by exploiting a large number of negative examples, SelGB is able to build ranking models that result to be more accurate than those learned with state-of-the-art ranking algorithms.

This section is organized as follow. First we introduce the methodology used for the experimental evaluation, and the challenging LtR dataset used for the experiments. We then analyze the performance achieved on this dataset by the λ-Mart state-of-the-art algorithm [2], and by two variants of the same algorithm, which exploit samples of the training instances. Finally, we discuss the performance achieved by the proposed Selective Gradient Boosting algorithm. The effectiveness metric adopted throughout the experiments is NDCG@10 [10]. To ease the reproducibility of the results, we release to the public the new dataset employed, along with the source code of our implementation of Selective Gradient Boosting[1].

5.1 Datasets and Methodology

We aim to mimic a real-world production environment employing a multi-stage ranking pipeline. Unfortunately, the available LtR public datasets are not suited to deeply investigate the impact of the volume of negative examples on state-of-the-art LtR algorithms. In fact, public available datasets provide a small amount of assessed examples per query. The most popular LtR datasets, i.e., MSN Learning to Rank[2] and Yahoo! LETOR Challenge (sets 1 and 2)[3], provide on average 120 and 20 documents per query, respectively. To investigate this research line, we thus built a new dataset from a subset of 10,000 queries sampled from a log of a real-world Web search engine. For each query, up to 5,000 results were retrieved from a collection of 44,830,467 Web documents after being ranked according to a BM25F scoring function [25]. The dataset, hereinafter named Istella-X^{5k} (eXtended), contains in total 26,791,447 query-document pairs. Each query-document pair is represented by 220 features, and is labeled by a integer relevance judgment ranging from 0 (not relevant) to 4 (perfectly relevant). On average, the dataset contains 2,679 documents per query, but only 4.64 of them are relevant/positive examples, i.e., only 4.64 query-document pairs are associated with a relevance judgment in the range $[1, 4]$. Indeed, the setting we adopted for the dataset creation is similar to that of [4] where BM25 was used to retrieve the set of documents to which a multi-stage ranking pipeline is applied.

[1] http://quickrank.isti.cnr.it
[2] http://research.microsoft.com/en-us/projects/mslr/
[3] http://learningtorankchallenge.yahoo.com

To investigate to which extent the presence of negative examples may influence the performance of LtR algorithms, we also created three scaled-down variants of Istella-X^{5k}. For each query in the training set, we first sorted all negative examples in descending order of BM25F scores, and then we discarded everything beyond a given rank. We used this methodology to produce three datasets, Istella-X$^{2.5k}$, Istella-X^{1k} and Istella-X^{500}, having at most 2,500, 1,000 and 500 documents per query, respectively. Note that this scaling down does not affect positive examples that were always preserved. The four datasets thus share the same positive examples but they differ in the proportion of positive versus negative examples. The largest Istella-X^{5k} presents a fraction of 0.17% positive query-document pairs, while in the smallest Istella-X^{100} this ratio is almost 30 times higher (5.11%). Table 1 reports some properties of the new proposed datasets. Each dataset was then split in three sets: train (60%), validation (20%), and test (20%). We built the three partitions of each dataset by always including the same queries in each of them, so as to allow a direct comparison of the performance achieved by the learned models. Although we exploit the train and validation sets of a given scaled-down dataset to learn a LtR model, at testing time we always used the test split of the complete Istella-X^{5k}, to fairly compare the effectiveness of the learned models, as Istella-X^{5k} best matches the reference scenario [4, 23]. The reduced datasets Istella-X$^{\{100,500,1k,2.5k,5k\}}$ are used in Section 5.2 to comprehensively evaluate the performance of the λ-Mart algorithm.

We run each algorithm to train up to 1,000 trees. To avoid overfitting, all the algorithms employed an "early stop" condition during training that allows to halt the learning process if no improvement in terms of NDCG@10 on the validation set is observed on the last 100 trees trained. When comparing different methods, we also evaluated statistical significance by using the randomization test with 10,000 permutations and $p\text{-}value \leq 0.05$ [20].

5.2 Gradient Boosting for Ranking: λ-Mart

We first investigate the behavior of λ-Mart [21], a state-of-the-art gradient boosting algorithm that exploits NDCG as loss function. The goal of this study is to understand whether λ-Mart is able to exploit a large number of negative examples at training time and to evaluate its robustness to high class imbalance.

We trained several λ-Mart models on the train split of the five datasets Istella-X$^{\{100,500,1k,2.5k,5k\}}$ and we evaluated their performance in terms of NDCG@10 on the test split of Istella-X^{5k}. In the following we refer to these models with the names λ-Mart^{100},

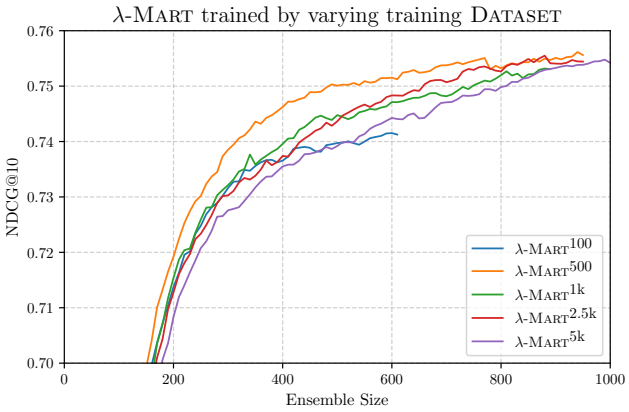

Figure 1: Effectiveness of λ-MART, trained on ISTELLA-X$^{\{500,1k,2.5k,5k\}}$, computed on the test split of ISTELLA-X^{5k}.

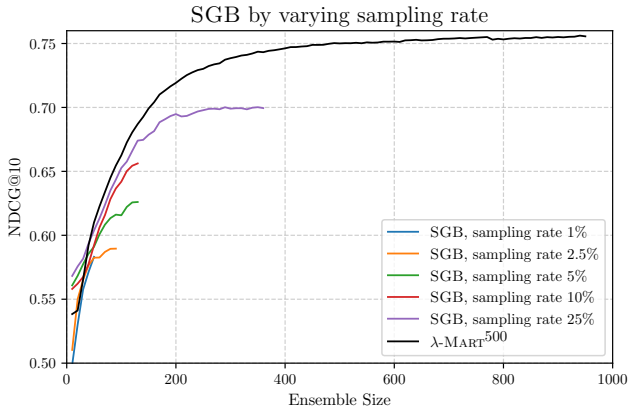

Figure 2: Effectiveness of STOCHASTIC GRADIENT BOOSTING trained and tested on ISTELLA-X^{5k} at different sampling rates.

λ-MART^{500}, λ-MART^{1k}, λ-MART$^{2.5k}$ and λ-MART^{5k}, respectively. The training process of the λ-MART algorithm was finely tuned by sweeping its learning parameters on the train split of ISTELLA-X^{5k} and by exploiting the "early stop" condition on the validation split of the same dataset. We varied the maximum number of tree leaves in the set $\{8, 16, 32, 64\}$, and the learning rate ν in $\{0.05, 0.1, 0.5, 1.0\}$. The best performance of λ-MART^{5k} was obtained when employing a learning rate equal to 0.05 and 64 leaves. We applied this combination of parameters in all the experiments we report in the following.

Figure 1 reports the performance in terms of NDCG@10 of the aforementioned λ-MART models as a function of the number of trees in the learned ensembles. We first highlight that the model λ-MART^{100} performs significantly worse than the others. Moreover, the best performance achieved by the other models range from 0.7532 (λ-MART^{1k}) to 0.7562 (λ-MART^{500}), with the latter being the best performing overall. However the differences of performance among these models are not statistically significant.

We also highlight that the λ-MART^{500} model dominates the others with interesting gains when employing only a fraction of the trees in the ensemble model. For example, when scoring the testing set with the first 400 trees of the ensemble, the λ-MART^{500} model scores 0.7462, while the second better is λ-MART^{1k} that scores 0.7396. In this case, the difference is statistically significant. The results above lead to two considerations. First, ISTELLA-X^{100} does not allow to learn models that are more effective than the ones obtained with larger datasets. Although ISTELLA-X^{100} is made up of a large number of negative instances (about 95% of all query-document pairs), it still misses some negative examples that are important to allow the resulting models to increase its generalization power. Therefore, a high number of negative instances provides useful information for training accurate ranking models. Second, the similar performance achieved by the best performing models suggests that λ-MART is quite robust with respect to the class imbalance of the examples, as its performance does not degrade significantly when increasing the number of negative examples.

The above experiments suggest that negative instances are very informative during the learning process of a λ-MART algorithm. Moreover, the use of a dataset providing about 500 negative instances per query is sufficient to achieve the best performance in terms of NDCG@10 in this experimental setting. However, the above experiment does not allow to conclude that ISTELLA-X^{500} is as informative as ISTELLA-X^{5k}, or rather that λ-MART is not capable of exploiting larger datasets. It is also worth reminding that no other `LtR` dataset released to the public so far provides a number of documents per query allowing to investigate the above issues. Our publicly available dataset thus contributes by allowing the reproducibility of our experiments and further investigation of this research line.

5.3 STOCHASTIC GRADIENT BOOSTING

STOCHASTIC GRADIENT BOOSTING (SGB) [8] is a natural competitor of our SELGB algorithm. At each iteration, SGB fits a weak learner (λ-MART in our case) by using a random sample of the training dataset. The introduction of a randomization step that samples data instances allows for an increased robustness to over-fitting and a reduction of the variance. Moreover, it provides computational savings since each iteration of the learning process deals with a reduced amount of data.

The experiments reported in this section are performed on the ISTELLA-X^{5k} dataset. We trained a STOCHASTIC GRADIENT BOOSTING λ-MART model with different sampling rates in the set {1%, 2.5%, 5%, 10%, 25%}. As we did for λ-MART, we used the validation set to avoid over-fitting. We also employed the same hyper-parameters, i.e., learning rate and the maximum number of leaves, found to be optimal for λ-MART.

Figure 2 reports the performance of the SGB models learned in terms of NDCG@10. Interestingly, none of the models perform better than λ-MART. The model with the least aggressive sampling, i.e., 25% is the best performing one, even if it is far below λ-MART (0.6990 versus 0.7548). A possible explanation of this phenomenon is that the original version of SGB [8] is not query-aware. SGB samples instances independently of queries, which may lead to queries associated with a significantly decreased number of candidate documents after sampling. Second, positive instances are sampled with the same uniform probability, therefore removing important information from the dataset. To solve these issues, we implemented a

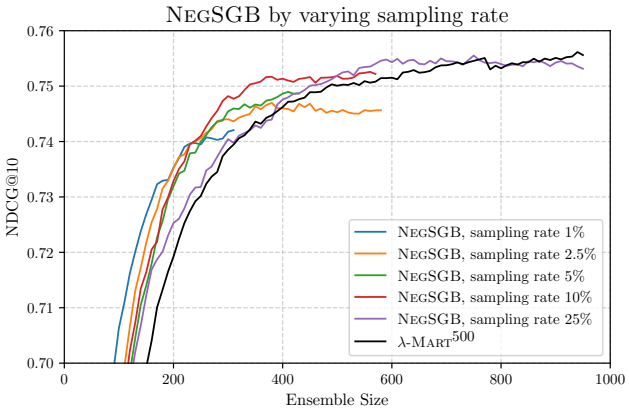

Figure 3: Effectiveness of Negative Stochastic Gradient Boosting trained and tested on Istella-X5k at different sampling rates.

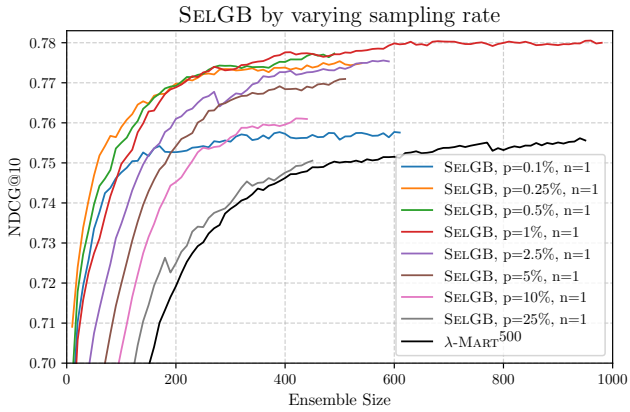

Figure 4: Effectiveness of Selective Gradient Boosting trained and tested on Istella-X5k at different sampling rates.

modified version of Stochastic Gradient Boosting that adopts a different sampling strategy. The new algorithm, named Negative Stochastic Gradient Boosting (NegSGB), samples instances at query level, by explicitly considering the lists of documents associated with each query in the training set. Moreover, it performs sampling only on the negative examples. Positive examples of each list are always included in the sample used. This is indeed similar to the common technique of under-sampling the most frequent class to reduce imbalance, but conducted at the query level rather than at the dataset level. To some extent, NegSGB is inspired by the work of Ibrahim and Carman [9]. In that work, authors propose to apply randomized undersampling techniques to deal with high class imbalance of examples and they do this at query level. NegSGB can thus be seen as the extension of the work by Ibrahim and Carman to gradient boosting as the method we propose perform selection of negative instances at query level during training. Compared to Stochastic Gradient Boosting, the advantage of this sampling strategy is that it both preserves i) all the positive instances and ii) the per-query examples distribution.

Figure 3 reports the performance of NegSGB trained and tested on Istella-X5k. The new algorithm achieves a remarkably better performance over SGB, thus confirming that per-query sampling of instances belonging to the negative class is an effective technique to improve the effectiveness achieved by the randomization step. Moreover, NegSGB outperforms λ-Mart^{5k} with valuable gains in terms of NDCG@10 when partial ensembles with a limited number of trees are considered. However, NegSGB is not able to outperform λ-Mart on the full ensemble. Indeed, the best performing NegSGB model, i.e., the one using a sampling rate of 25%, achieves a NDCG@10 of 0.7531 vs. 0.7556 achieved by λ-Mart^{500}, and the difference is not statistically significant. On the other hand, the adoption of smaller sampling rates leads over-fitting, thus causing the early stop condition based on the validation set to halt the training process quite early.

In summary, the analysis reveals that standard sampling approaches do not help λ-Mart in exploiting all the information available in the largest dataset Istella-X5k. Indeed, the best performance figures achieved so far were obtained by λ-Mart trained on Istella-X500, and by Negative Stochastic Gradient Boosting trained on Istella-X5k, despite the latter makes use of much more training data. Therefore, we can conclude that i) the λ-Mart^{500} model provides the best performance, and ii) learning λ-Mart^{500}-based models requires to process less data than the best performing NegSGB model, as the latter employs a sampling rate of 25% of each per-query list, thus generating lists composed of an average number of 1,250 documents compared to the 500 ones of Istella-X500.

5.4 Selective Gradient Boosting

Unlike other algorithms discussed above, the query level samples produced by Selective Gradient Boosting are rank-aware. During a training iteration, it focuses on on a small sample of the negative instances, accurately chosen on the basis of the current ranks as computed by the scoring model learned so far.

5.4.1 Hyper Parameters Tuning. SelGB shares several hyper parameters with λ-Mart, i.e., the learning rate, the total number of trees, and maximum number of leaves. It also introduces two new parameters, namely n and p, that actively drive the learning process and characterize the newly proposed algorithm. As stated in Section 4, the former drives the frequency of the selective sampling step while the latter drives the fraction of per-query negative samples to keep. In the following experiments, we report the effectiveness of SelGB by varying these two parameters independently of each other, so as to provide interesting insights regarding the behavior of the algorithm.

Figure 4 reports the performance achieved by SelGB as a function of the sampling rate p. Compared with λ-Mart, the performance improvement is apparent. The best performance is achieved with a sampling rate of 1%, which is equivalent to using a maximum of 50 negative documents per query. Under these settings, SelGB achieves an effectiveness of 0.7800 in terms of NDCG@10, with a gain of 0.0244 (+3.23%) over the best performing λ-Mart^{500} model (NDCG@10 = 0.7556). Even when employing lower sampling rates (0.25% and 0.5%), the resulting performance is similar to the best NDCG achieved with sampling rate of 1%. On the other hand, the higher is the sampling rate, the more the performance of the

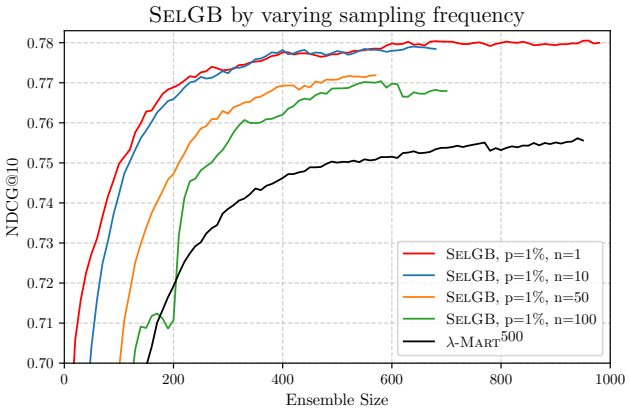

Figure 5: Effectiveness of SELECTIVE GRADIENT BOOSTING trained and tested on ISTELLA-X5k by varying the number of iterations between two consecutive sampling steps.

SELGB algorithm converge to the one of λ-MART. As an example, when employing a sampling rate of 25% the SELGB model performs similarly to the λ-MART^{500} one.

It is also worth highlighting that such good performance is achieved quite early during the training. About 100 trees of the 1% sampled model provide almost the same effectiveness of the full λ-MART^{5K}, and 600 trees provide almost optimal performance. Thus not only SELGB can generate more effective models, but it also remarkably improves the scoring efficiency as a side effect, as it can grant an effectiveness similar to that of λ-MART^{500} with much smaller ensembles.

Figure 5 reports the performance of SELGB when the number of iterations between two consecutive selective sampling steps is varied, i.e., acting on parameter n. In this analysis, the sampling rate (p) has been fixed to 1%, as this is the value providing the best performance in the previous experiment. The model obtained by selectively sampling instances at each iteration ($n = 1$) is the best one, along with the one performing sampling every 10 trees learned. Indeed, when sampling less frequently, the effectiveness of the models start decreasing ($n = 50$, $n = 100$). This behavior is apparent when we look at the performance curve of the model obtained when sampling every 100 iterations. The behavior of the curve between 100 and 200 trees reveals an important increase in performance in the first part. Suddenly the curve starts to decrease due to over-fitting as more trees are added. This degraded behavior continues until a new sampling is generated. From that point on, the performance increases sharply from 200 to 220 trees, and then the curve start becoming always more flattened, until it converges to the final effectiveness of the model.

5.4.2 Effectiveness Analysis. We now present a comparison of the effectiveness of SELGB against five competitors. Table 2 reports the performance of all the algorithms tested in terms of NDCG@10. Each algorithm was trained by exploiting the hyper parameters combination that maximize its performance. We also report the difference in performance over the most effective baseline λ-MART^{500}, and we highlight with the * symbol the performances that are statistically different with respect to the one obtained by the model

Table 2: Effectiveness of SELGB and five competitors tested on ISTELLA-X5k in terms of NDCG@10. We report the relative difference of performance over λ-MART^{500}. The * symbol highlights statistically significant differences.

Algorithm	150 trees	Full ensemble
SELGB	$0.7628^{*}_{+9.1\%}$	$0.7800^{*}_{+3.2\%}$
λ-MART^{500}	0.6992	0.7556
λ-MART^{5k}	$0.6855^{*}_{-2.0\%}$	$0.7542_{-0.0\%}$
SGB	$0.6787^{*}_{-2.9\%}$	$0.6990^{*}_{-7.5\%}$
NEGSGB	$0.7122^{*}_{+1.9\%}$	$0.7531_{-0.0\%}$
Lucchese *et al.* [17]	$0.6982_{-0.0\%}$	$0.7583_{+0.0\%}$

learned by λ-MART^{500}. Among the baselines reported, we also include the work by Lucchese *et al.* [17], as we share a similar research goal, i.e., to sample instances for improving efficiency and effectiveness of LtR models. However, they propose a *a priori* sampling of negative instances, which is not embedded in the learning process. On the contrary, SELGB selects dynamically the negative examples to be used at training time, i.e., during the iterative growing of the scoring model.

Interestingly, SELGB achieves an absolute gain of 0.0244 in terms of NDCG@10 over λ-MART^{500}, which accounts for a significant +3.2% improvement in effectiveness. The solution by Lucchese *et al.* shows a marginal gain, as to highlight the deficiencies of a static sampling approach. Conversely, SGB and NEGSGB are both not particularly effective in training their models.

We provide additional insights on the effectiveness of the newly proposed algorithm by reporting the performance obtained by scoring functions that only exploit the first 150 trees of the ensembles under analysis. Here, SELGB scores 0.7628 in terms of NDCG@10. Despite the usage of only 15% of the trees, SELGB performs better than the full λ-MART^{500} model, composed of 1,000 trees. Moreover, when comparing the effectiveness of the two models employing only the first 150 trees, SELGB shows an important absolute gain of 0.0636, corresponding to a performance improvement of +9.1%.

We are also interested in analyzing how the difference in performance provided by SELGB against λ-MART is spread across queries with a different list size, i.e., the number of per-query training examples, also called *query list size*. We perform this analysis by bucketing queries by the number of documents they are associated with. Figure 6 (top half) reports the average per-query difference of NDCG@10 achieved by SELGB against λ-MART^{500} (y-axis), while the bottom half of the figure presents the number of queries (y-axis) falling into each bucket. The x-axis reports the query list size binned at intervals of 100 documents.

First, we observe that ISTELLA-X5k contains 303 (15%) queries with at most 100 documents, and 787 (39%) queries with more than 4,900 documents. An improvement in this last bucket of queries having the largest list size has a significant impact on the overall performance. Indeed, SELGB always provides an improvement over λ-MART for queries with list size greater than 3,700, with a significant average gain in NDCG@10 of about 0.05 for the last bin of queries. For smaller query with list sizes up to 600 documents, the

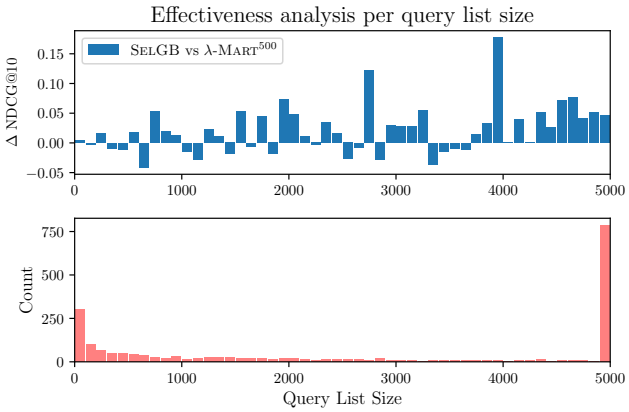

Figure 6: Effectiveness of SelGB vs. λ-Mart^{500} as a function of the query list size (top) and distribution of sizes of the query lists (bottom).

performance of SelGB is similar to that of λ-Mart, still using a much smaller number of negative instances thanks to the adopted sampling rate $p = 1\%$.

To conclude, SelGB remarkably outperforms competitors. The largest gain of performance is achieved for queries having larger list sizes. These are very likely to be difficult queries, potentially matching several thousands of documents beyond the 5,000 limit we imposed. The above experiments confirm that, especially for queries associated with a large number of candidate documents, SelGB is able to select and exploit the most informative negative examples.

5.4.3 Document Selection Analysis. To better understand the dynamics of SelGB, we conducted an analysis of which negative examples are actually selected by SelGB during training. We report the results of this analysis in Figure 7. The heatmap shows the SelGB selection of negative instances by displaying, on the y-axis, from the top to the bottom of the figure, queries sorted in ascending order of their list size. The x-axis reports the ranked list of documents per query, sorted in descending order of their original value of BM25F scores. For each query, the brighter the color associated to each document, the higher the number of times the document has been selected as a negative instance by SelGB during training. The figure was built by using a threshold on the number of times a document is selected to reduce noise. We employed a threshold equal to 10, meaning that a bright yellow color refers to documents used in at least 10 different iterations of the learning. The outcome of this analysis is two-fold: i) the most frequent negative examples selected are those with the highest BM25F score, meaning that there is a slight correlation between the ranking metric and BM25F; ii) despite that, and especially for queries with more than 4,900 documents, SelGB achieves better performance by selecting negative instances with lower rank, thus proving the usefulness of examples with lower BM25F scores.

To provide a more quantitative analysis, we report in Figure 8, the cumulative distribution of the fraction of negative instances selected during the training of SelGB as a function of their BM25F rank. Interestingly, 80% of the negative instances were selected in the top 1,000 positions. The remaining 20% were selected from instances

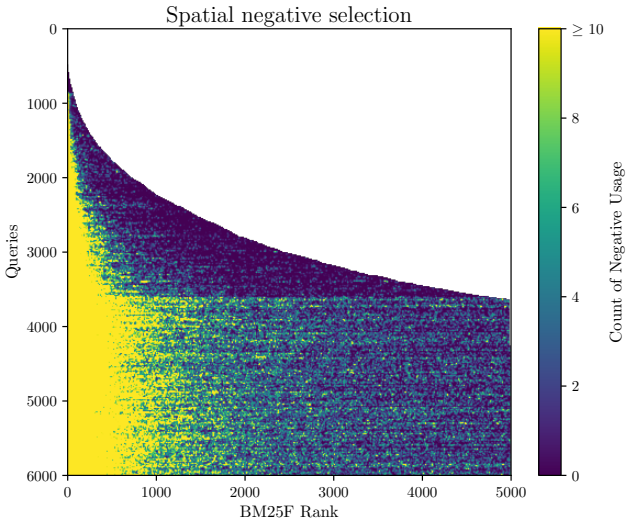

Figure 7: Selection of negative examples for each query in the train split of Istella-X^{5k}. Queries are sorted in decreasing order of the size of their associated lists. On the y-axis, from the top to the bottom of the plot, we report queries with increasing number of documents. On the x-axis we report the document ranks induced by the associated BM25F scores. The brighter the color associated with each position, the higher the number of times the document has been selected by SelGB at training time.

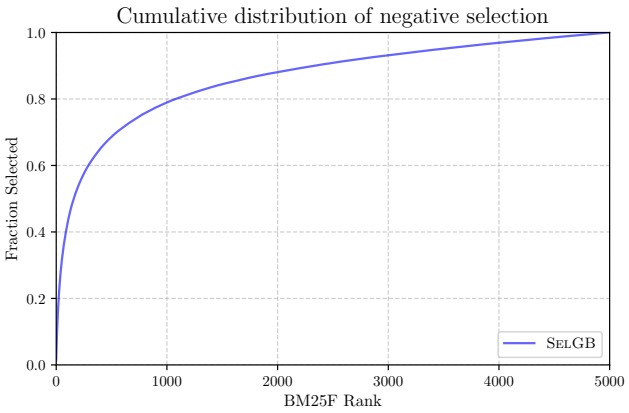

Figure 8: Cumulative distribution of negative selection of SelGB on the train split of Istella-X^{5k}.

with lower ranks. The analysis reveals that training a model by using only a few hundreds negative examples per query strongly limits the effectiveness of the resulting model, see the performance of λ-Mart^{500} in Table 2. The proposed SelGB algorithm achieves a significant gain by selecting 20% of its training instances among those beyond rank 1,000, despite using a small number of training instances thanks to the adopted sampling rate. This confirms the importance of properly exploiting negative examples during the training process.

6 CONCLUSION

The aim of this work was to learn LtR models that are able to effectively rank the huge number of candidate documents retrieved per each query in a multi-stage retrieval system. In this context, the first stage of the system aims to maximize recall, and thus many candidate documents must be retrieved to avoid missing relevant results. These documents have then to be re-ranked by a precise and accurate LtR model in the following ranking stage.

We performed an in-depth investigation of this topic that led us to propose Selective Gradient Boosting (SelGB), a new stepwise algorithm introducing a tunable and dynamic selection of negative instances within λ-Mart. SelGB produces as λ-Mart an ensemble of binary decision trees but performs at training time a dynamic selection of the negative examples to be kept in the training set. In particular, the algorithm selects the top-scored negative instances within the lists associated with each query with the aim of minimizing the mis-ranking risk. Due to the characteristics of the NDCG metric used to evaluate the quality of the learned model, we need to discriminate the few positive instances that must be pushed in the top positions of the scored lists, from the plenty of negative instances in the training set. Indeed, top-scored negative instances are exactly those being more likely to be ranked above relevant instances, thus severely hindering the ranking quality.

Unlike other sampling methods proposed in the literature, our method does not simply aim at sampling the training set to reduce the training time without affecting the effectiveness of the trained model. Conversely, the proposed method is able to dynamically choose the "most informative" negative examples of the training set, so as to improve the final effectiveness of the learned model. A comprehensive experimental evaluation, based on a new very large dataset shows that SelGB achieves an astonishing NDCG@10 improvement of 3.2% over the reference λ-Mart state-of-the-art algorithm. To ease the reproducibility of our results and favor the research on this challenging topic, we released both the source code of Selective Gradient Boosting and the new dataset with 10,000 queries and thousands of assessed documents per query.

As future work, we plan to study more in deep different strategies, with the final goal of improving SelGB. First we aim at investigating an adaptive strategy for selecting the negative instances to be kept in the training set. Such strategy could choose on a per query basis the fraction of negative examples. This is motivated by the characteristics of real-world search system, where the distribution of the number candidate documents per query is skewed and some queries are more difficult than other to answer. Another interesting research direction regards the selection methodology used to dynamically choose the instances in the training set. Now it is only based on the scores of the negative examples computed by the model learned so far, without randomization and, more importantly, without considering the current λ-gradients, which in turn push modifications to these scores with the newly added trees. Therefore, we left as a future work the investigation of selection mechanisms favoring the examples with the largest λ-gradients. Finally, for a lack of space, we did not studied the improved efficiency of SelGB over λ-Mart introduced by the subsampling strategy, neither how we can obtain an efficiency/effectiveness trade-off by reducing the frequency of the selection step.

Acknowledgements. This paper is supported by the MASTER (grant agreement Nº777695), BIGDATAGRAPES (grant agreement Nº780751), SOBIGDATA (grant agreement Nº654024) and ECO-MOBILITY (project ID Nº10043082) projects that received funding from the European Union's Horizon 2020 research and innovation programme under the Marie-Slodowska Curie, Information and Communication Technologies, and Interreg Italy-Croatia CBC work programmes. We also acknowledge the support of Istella S.p.A., and, in particular, of Domenico Dato and Monica Mori.

REFERENCES

[1] Javed A. Aslam, Evangelos Kanoulas, Virgil Pavlu, Stefan Savev, and Emine Yilmaz. 2009. Document Selection Methodologies for Efficient and Effective Learning-to-rank. In *Proc. SIGIR*. ACM, 468–475.
[2] Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11, 23-581 (2010), 81.
[3] G. Capannini, C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, and N. Tonellotto. 2016. Quality versus efficiency in document scoring with learning-to-rank models. *Information Processing & Management* 52, 6 (2016), 1161 – 1177.
[4] Ruey-Cheng Chen, Luke Gallagher, Roi Blanco, and J. Shane Culpepper. 2017. Efficient Cost-Aware Cascade Ranking in Multi-Stage Retrieval. In *Proc. SIGIR*. ACM, 445–454.
[5] Van Dang, Michael Bendersky, and W Bruce Croft. 2013. Two-Stage Learning to Rank for Information Retrieval.. In *ECIR*. Springer, 423–434.
[6] D. Dato, C. Lucchese, F.M. Nardini, S. Orlando, R. Perego, N. Tonellotto, and R. Venturini. 2016. Fast ranking with additive ensembles of oblivious and non-oblivious regression trees. *ACM TOIS* 35, 2 (2016).
[7] Pinar Donmez and Jaime G Carbonell. 2008. Optimizing estimated loss reduction for active sampling in rank learning. In *Proc. ICML*. ACM, 248–255.
[8] J. H. Friedman. 2002. Stochastic gradient boosting. *Computational Statistics & Data Analysis* 38, 4 (2002), 367–378.
[9] Muhammad Ibrahim and Mark Carman. 2014. *Undersampling Techniques to Rebalance Training Data for Large Scale Learning-to-Rank*. Springer International Publishing, Cham, 444–457.
[10] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.* 20, 4 (Oct. 2002), 422–446.
[11] Evangelos Kanoulas, Stefan Savev, Pavel Metrikov, Virgil Pavlu, and Javed Aslam. 2011. A Large-scale Study of the Effect of Training Set Characteristics over Learning-to-rank Algorithms. In *Proc. SIGIR*. ACM, 1243–1244.
[12] 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*. 3149–3157.
[13] Bo Long, Olivier Chapelle, Ya Zhang, Yi Chang, Zhaohui Zheng, and Belle Tseng. 2010. Active Learning for Ranking Through Expected Loss Optimization. In *Proc. SIGIR*. ACM, 267–274.
[14] Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, and Trani Salvatore. 2018. X-CLEaVER: Learning Ranking Ensembles by Growing and Pruning Trees. *ACM Transactions on Intelligent Systems and Technology (TIST), to appear* (2018).
[15] Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, and Salvatore Trani. Post-Learning Optimization of Tree Ensembles for Efficient Ranking. In *Proc. SIGIR*. ACM, 949–952.
[16] Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, and Salvatore Trani. 2017. X-DART: Blending Dropout and Pruning for Efficient Learning to Rank. In *Proc. SIGIR*. ACM, 1077–1080.
[17] Claudio Lucchese, Franco Maria Nardini, Raffaele Perego, and Salvatore Trani. 2017. The Impact of Negative Samples on Learning to Rank. In *Proc. LEARning Workshop co-located with ACM ICTIR*. CEUR-WS.org.
[18] Craig Macdonald, Rodrygo LT Santos, and Iadh Ounis. 2013. The whens and hows of learning to rank for web search. *Information Retrieval* 16, 5 (2013), 584–628.
[19] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*. Cambridge University Press.
[20] M. D. Smucker, J. Allan, and B. Carterette. 2007. A Comparison of Statistical Significance Tests for Information Retrieval Evaluation. In *Proc. CIKM*. ACM.
[21] Q. Wu, C.J.C. Burges, K.M. Svore, and J. Gao. 2010. Adapting boosting for information retrieval measures. *Information Retrieval* (2010).
[22] Emine Yilmaz and Stephen Robertson. 2009. Deep Versus Shallow Judgments in Learning to Rank. In *Proc. SIGIR*. ACM, 662–663.
[23] D. Yin, Y. Hu, J. Tang, T. Daly, M. Zhou, H. Ouyang, J. Chen, C. Kang, and others. 2016. Ranking relevance in yahoo search. In *Proc. ACM SIGKDD*. ACM, 323–332.
[24] Hwanjo Yu. 2005. SVM Selective Sampling for Ranking with Application to Data Retrieval. In *Proc. ACM SIGKDD*. ACM, 354–363.
[25] H. Zaragoza, N. Craswell, M.J. Taylor, S. Saria, and S.E. Robertson. 2004. Microsoft Cambridge at TREC 13: Web and Hard Tracks.. In *TREC*, Vol. 4. 1–1.