IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 22 September 2023, accepted 31 October 2023, date of publication 8 November 2023, date of current version 15 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3331088

RESEARCH ARTICLE

Early Exit Strategies for Learning-to-Rank Cascades

FRANCESCO BUSOLIN[1], CLAUDIO LUCCHESE[1], FRANCO MARIA NARDINI[2], SALVATORE ORLANDO[1], RAFFAELE PEREGO[2], AND SALVATORE TRANI[2]
[1]Department of Environmental Science, Informatics and Statistics, Ca' Foscari University of Venice, 30172 Venice, Italy
[2]Institute of Information Science and Technologies (ISTI), National Research Council (CNR), 56124 Pisa, Italy

Corresponding author: Francesco Busolin (francesco.busolin@unive.it)

This work was supported in part by the SEcurity and RIghts In the CyberSpace (SERICS) Project under Grant NRRP M4C2 Inv.1.3 PE00000014; in part by the Interconnected Nord-Est Innovation Ecosystem (iNEST) Project through the Next Generation EU (EU-NGEU) under Grant NRRP M4C2 Inv.1.5 ECS00000043; in part by the Future Artificial Intelligence Research (FAIR)-Spoke 1 "Human-Centered Artificial Intelligence (AI)" funded by the European Commission through the NextGeneration EU Program, Piano Nazionale di Resistenza e Resilienza, research guidelines M4C2 (PNRR-M4C2)-Investimento 1.3, under Grant Partenariato Esteso PE00000013; in part by the Horizon Europe Research and Innovation Action (RIA) "Extreme Food Risk Analytics (EFRA)" under Grant 101093026.

ABSTRACT The ranking pipelines of modern search platforms commonly exploit complex machine-learned models and have a significant impact on the query response time. In this paper, we discuss several techniques to speed up the document scoring process based on large ensembles of decision trees without hindering ranking quality. Specifically, we study the problem of document early exit within the framework of a cascading ranker made of three components: 1) an efficient but sub-optimal ranking stage; 2) a pruner that exploits signals from the previous component to force the early exit of documents classified as not relevant; and 3) a final high-quality component aimed at finely ranking the documents that survived the previous phase. To maximize speedup and preserve effectiveness, we aim to increase the accuracy of the pruner in identifying non-relevant documents without early exiting documents that are likely to be ranked among the final top-k results. We propose an in-depth study of heuristic and machine-learning techniques for designing the pruner. While the heuristic technique only exploits the score/ranking information supplied by the first sub-optimal ranker, the machine-learned solution named LEAR uses these signals as additional features along with those representing query-document pairs. Moreover, we study alternative solutions to implement the first ranker, either a small prefix of the original forest or an auxiliary machine-learned ranker explicitly trained for this purpose. We evaluated our techniques through reproducible experiments using publicly available datasets and state-of-the-art competitors. The experiments confirm that our early-exit strategies achieve speedups ranging from 3× to 10× without statistically significant differences in effectiveness.

INDEX TERMS Query processing, efficiency/effectiveness trade-offs, learning-to-rank.

STATEMENTS AND DECLARATIONS

This manuscript is an extension of our ACM SIGIR 2021 short paper entitled "Learning Early Exit Strategies for Additive Ranking Ensembles" [4]. In this extension we investigate different solutions for designing and training an early-exit component in our document early exit framework. At the end of the analysis, we propose a solution that improves our preliminary model and the heuristic-based method described by Cambazoglu et al. [5].

The associate editor coordinating the review of this manuscript and approving it for publication was Hai Dong.

I. INTRODUCTION

Query processors on modern search platforms rely on sophisticated ranking pipelines to optimize precision-oriented list-wise metrics at small cutoffs. Learning-to-rank (LtR) techniques are widely used to train models that can precisely re-rank a set of candidate documents in the last stages of the pipeline. Among the state-of-the-art solutions for candidate

re-ranking, particular relevance have the models based on additive ensembles of regression trees learned by gradient boosting algorithms such as MART [12] and λ-Mart [3], [29]. As such ensembles may include hundreds of regression trees to be visited for scoring each candidate document, the tight constraints on query response time typical in web-scale platforms require suitable solutions able to provide an optimal trade-off between document scoring time and ranking effectiveness [6], [26]. Several techniques have been proposed in recent years to target this efficiency-effectiveness trade-off [2], [9], [23], [33]. Among them, one line of research investigates early termination heuristics aimed to reduce, on a document- or query-level basis, the number of trees traversed during the scoring process [5], [20]. These works study the impact of the proposed early termination strategies on scoring latency and ranking accuracy.

In this paper, we investigate document-level *early exit* (EE) strategies for additive ranking ensembles by generalizing and building upon the state-of-the-art method introduced by Cambazoglu et al. [5]. Since documents are re-ranked by traversing the whole ensemble to accumulate the final scores, [5] proposed some heuristic techniques to speed up query processing by forcing documents to early exit the ensemble if they are unlikely to be included in the top-k results. The techniques rely upon simple thresholding strategies exploiting the partial scores/rankings computed at a *sentinel* point of the ensemble, i.e., after having evaluated a limited number of trees. Here, we build upon this simple approach. First, we propose an in-depth investigation of the partial document scorings/rankings as additional features to represent query-document pairs. We use this data representation along a suited ground truth to train LEAR, a machine learning (ML) solution in charge of deciding which documents should early exit the ranking pipeline because they will be unlikely ranked among the final top-k results. Second, we study the advantage of using, rather than an initial portion of the forest until the sentinel, an *auxiliary ranker*, i.e., an alternative compact ranking forest, specifically trained to provide high-quality early-exit signals with a reduced scoring cost. We guess this auxiliary forest can supply more reliable information at the sentinel, thus allowing a more accurate pruning of non-promising documents.

Our contribution allows the early exit process to be framed as a *cascade* of three components, namely $Ranker_{\text{pre}}$, *Pruner*, and $Ranker_{\text{post}}$, as illustrated in Figure 1. The main goal of the $Ranker_{\text{pre}}$ component, implemented as a small ranking forest, is to supply scoring/ranking information to the *Pruner* component. Specifically, $Ranker_{\text{pre}}$ assigns a score S to each query-document pair (q, d) and passes the triple (q, d, S) to the next stage *Pruner*. The goal of *Pruner* is to maximize the overall speedup at inference time without hindering the overall ranking quality of the query processor. To obtain significant speedups, *Pruner* aims to reduce the number of candidates traversing $Ranker_{\text{post}}$, a computationally expensive forest optimized for high precision. In other terms, the goal is to *minimize* the number of query-document pairs following the **continue** arrow (see Figure 1) to complete their scoring by traversing $Ranker_{\text{post}}$.

This work extends a previous contribution by [4] that proposes a preliminary machine learning (ML) solution for early terminating document scoring. In this paper, we build upon the previous investigation and extend it with the following novel contributions:

- Given a ranking forest $\mathcal{E}$, we study the possibility of exploiting a trained ensemble $\mathcal{E}_{\text{aux}}$ as $Ranker_{\text{pre}}$, where $\mathcal{E}$ thus plays the role of $Ranker_{\text{post}}$. Moreover, we deepen the investigation of how to split $\mathcal{E}$ into $\mathcal{E}_{\text{pre}}$ and $\mathcal{E}_{\text{post}}$, where $\mathcal{E}_{\text{pre}}$ is a small sub-forest of $\mathcal{E}$, providing to *Pruner* the scores accumulated up to the *sentinel* point, whereas $\mathcal{E}_{\text{post}}$ is the remaining part of the whole forest $\mathcal{E}$. We then study and assess how to fruitfully exploit either $\mathcal{E}_{\text{aux}}$ or $\mathcal{E}_{\text{pre}}$ as $Ranker_{\text{pre}}$ in our cascading framework for document early exit introduced in Figure 1.
- We study the solution space for designing and training the *Pruner* component of our document early exit framework. To learn the classification model, we exploit the available representation of query-document pairs and approximate document scores/ranks, returned by $Ranker_{\text{pre}}$, i.e., either $\mathcal{E}_{\text{pre}}$ or $\mathcal{E}_{\text{aux}}$. *Pruner* predicts whether a document should exit the ranker because it will unlikely be ranked among the top-k documents returned for a given query or it should continue the traversal of the rest of the cascading ranker, namely $Ranker_{\text{post}}$. We provide an in-depth investigation of the possible solutions for training an effective classifier, providing a good trade-off between classification quality and cost.
- We conduct an extensive experimental analysis of the effectiveness and the speedup achieved by the introduction of cascading of three components for early exit in a query processor exploiting state-of-the-art ranking models and scoring algorithm [18], [29]. Reproducible experiments conducted on two well-known public LtR datasets, namely MSN-1 and ISTELLA, show that our machine-learned solutions for document-level early exit achieve speedups ranging from 3× to 10× without statistically significant differences in terms of NDCG@10.

The article is organized as follows. Section II discusses the relevant related work. Section III formally introduces the early exit problem. Section IV presents our contribution addressing the early exit problem while Section V discusses the results of our experimental analysis on public datasets. Finally, Section VI concludes the work and drafts some future work.

II. RELATED WORK

Several techniques have been proposed to target efficiency-effectiveness trade-offs in query processing. Among the main contributions in the area, we cite the algorithms for the efficient traversal of tree ensembles [9], [16], [18], [33]. Alternative methods are concerned with strategies for pruning

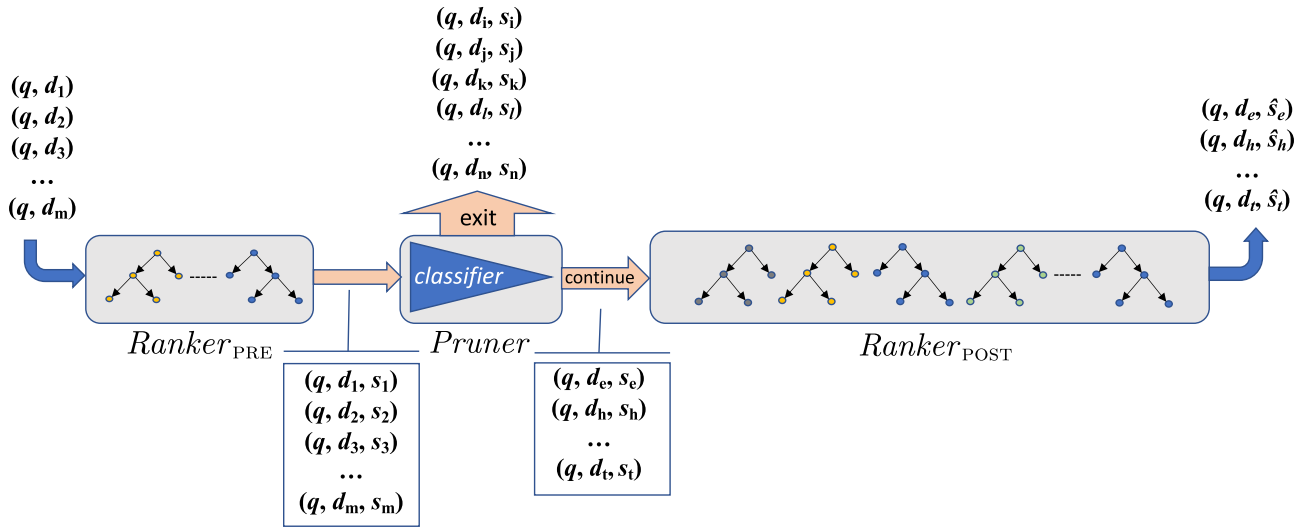


FIGURE 1. Our cascading framework.

the ensemble during or after the training phase [19], [22], [23], budget-aware LtR algorithms [2], [7], [13], [28], [32] and novel approaches for feature engineering and selection [14], [21]. A separate line of research investigates early exit optimizations in modern query processing exploiting machine-learned ranking models. The first work in this line is the one by [5], where authors introduce the notion of *sentinel* for additive ensembles of trees – a popular learning to rank technique used in two-stage query processors of modern search engines. A sentinel is a specific point s of the ensemble used to compute the partial score of a document. The authors introduce four heuristics that exploit the partial score calculated at a given sentinel to decide whether to early exit or not the score computation during document ranking. The four heuristics are built upon the concepts of document score/rank. The main intuition here is to use these two ingredients to define policies governing the early exit of documents during ranking. Among the four heuristics, the best performing strategy is the *Early Exits using Proximity Thresholds* (EPT). In EPT the pivot document rank is permanently fixed to k. The decision to eliminate a document is based on an offline-computed score proximity threshold p. The idea is to keep scoring the documents with a score close enough to the document's score at the k-th rank. Therefore, only the documents that are within the first k ranks, as well as documents that are within a score proximity of the k-th document's score, continue to be further scored. Authors evaluate their performance using a state-of-the-art machine learning system based on gradient-boosted decision trees. The proposed strategies achieve considerable speedups in online execution times of additive ensembles and, with the EPT approach, the improvement is up to four times with almost no loss in quality with respect to a baseline ranker that does not employ early exit strategies.

Later, Busolin et al. [4] build upon the state-of-the-art method introduced by Cambazoglu et al. [5] by proposing query-based and document-based machine-learned techniques for early exit. Authors propose LEAR (*Learned EArly exit Ranking*), a machine learning framework (ML) for early terminating document scoring. LEAR exploits a binary classifier based on query-document features and document score/rank computed at a given sentinel to predict if a given document is likely (unlikely) to be ranked among the top-k final ones, and it thus need to continue (exit) the traversal of the rest of the ensemble. Authors show effective techniques for building early-exiting classifiers and show on public data that the approach is practical and improves the state of the art by Cambazoglu et al. A different solution to this problem is investigated by Lucchese et al., which generalize the problem to a different level of granularity [20], i.e., on a query level. In detail, authors investigate the problem of *query-level* early exiting that asks for deciding the profitability of early stopping the traversal of the ranking ensemble for all the candidate documents to be scored for a query by returning a ranking based on the additive scores computed by a limited portion of the ensemble. In doing so, authors study the contribution of portions of the tree ensemble to the ranking of the top-k documents scored for a given query. Experiments on public data show that queries exhibit different behaviors as scores are accumulated during the traversal of the ensemble. Moreover, they show that query-level early stopping can remarkably improve ranking quality with an overall gain of up to 7.5% in NDCG@10 with a speedup of the scoring process of up to 2.2×.

Recently, following the growing interest in the field of Neural Ranking [8], [15], [24], efforts have been devoted to the study of early exit strategies for pre-trained transformers [17], [30], [31]. Xin et al. propose "Dynamic Early Exiting for BERT" (DeeBERT), where authors accelerate

BERT inference by inserting extra classification layers between each transformer layer of BERT [31]. Classification layers are jointly fine-tuned along with BERT on a given downstream dataset. At inference time, after a sample goes through a transformer layer, it is passed to the following classifier and returned only if the prediction shows low confidence; otherwise, the sample is sent to the next transformer layer. Experiments on six GLUE datasets show that, when applied to BERT and RoBERTa, DeeBERT can accelerate model inference by up to 40% with minimal model quality degradation. More recently, Xin et al. extend the approach above to other tasks than classification by defining BERxiT, where authors propose an effective fine-tuning method that allows to take advantage of the pre-trained model's effectiveness fully and a learning-to-exit approach that generalizes early exiting to other tasks [30]. Experiments on eight datasets for both classification and regression tasks show the effectiveness of BERxiT as a way to speed up the inference on pre-trained transformers. At the same time, Zhu proposes a novel training mechanism called "Learned Early Exiting for BERT" (LeeBERT) [34]. Unlike previous approaches, the central intuition behind LeeBERT is that different layers extract features of varying granularity. Layers thus provide different perspectives of the sentence. Zhu exploits this fact by learning early exits from each other to improve the expressiveness of lower exits and alleviate the over-fitting of the later exits. Second, their weights are treated as parameters and are learned along with model parameters. The optimization of the learnable weights is formulated as a bit-level optimization problem and optimized with gradient descent. Similarly, Soldaini and Moschitti propose a method to train a cascade model for question answering [25]. In their proposal, the authors build a sequence of increasingly complex re-rankers that process the candidates in a pipeline. In short, each re-ranker takes the set of candidates selected by the previous re-ranker and provides a subset of candidates to the next re-ranker. Naturally, this approach saves computation time from the more expensive re-rankers by progressively reducing the number of candidates at each step.

More general approaches to the same problem have been developed in the machine learning community. Viola and Jones introduce a similar algorithm for speeding up real-time face detection [27] based on a cascade of simple classifiers. Each classifier of the cascade is executed, and if its prediction is positive (i.e., a face is detected), then another classifier is triggered. Otherwise, the execution is short-circuited. This process continues until all classifiers agree that a face is detected. This approach is shown to save computational resources. Another approach based on a similar idea has been developed in the context of nonlinear support vector machines. Here, two works have been proposed to reduce online decision-making cost [10], [11]. Both solutions are based on the idea that the points that are far away from the decision boundary can be classified very quickly with high confidence.

III. THE EARLY EXIT PROBLEM

In this section we discuss and formalize the early exit problem within the framework of a cascading ranker as introduced in Section I. For easy reading, we include in Table 1 the list of symbols used for the rest of this work.

Let $\mathcal{E}$ be an additive ensemble of n regression trees $\{T_1, \ldots, T_n\}$, trained by algorithm $\mathcal{A}$ on a ground-truth dataset containing *query-document* pairs (q, d) associated with multi-level graded relevance labels. Given a test instance (q, d), we denote by $s_i(q, d)$ the score contribution induced by a tree T_i. The final prediction for (q, d) is obtained by accumulating the score contributions by all the n trees in the ensemble, and we refer to it as full-model score:

$$S(q, d) = \sum_{l=1}^{n} s_l(q, d) \tag{1}$$

In ad-hoc retrieval tasks we are mostly interested in *precision* at small cutoffs [1]; thus, the quality measure adopted to evaluate ranking effectiveness usually takes into account only the head of the ranked list, i.e., the top-k documents according to the ranking induced by the decreasing order of predicted scores. For this work, we adopt 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 predicted relevance $S(q, d)$ and discounts their contribution according to their rank position.

Given a test query q, a set D_q of candidate documents, and a cutoff value k (e.g., $k = 10$ in this work) the quality measure used to evaluate the ranking of D_q is not affected by the documents ranked in positions greater than k. Let $D_q^+ \subset D_q$ be the subset of documents ranked in the first k positions by the ranking ensemble, and let $D_q^- = D_q \setminus D_q^+$ the rest of D_q ranked after position k, thus not affecting the quality measure. Since computing the scores of documents in D_q^-, by traversing the whole ensemble $\mathcal{E}$, is a waste of time and computing resources, to improve efficiency without hindering effectiveness an optimal technique should *early* identify the documents in D_q^+ and D_q^-, and force the *exit* of documents in D_q^- from the ranking pipeline.

Referring to the cascade of $Ranker_{\text{pre}}$, *Pruner*, and $Ranker_{\text{post}}$ components illustrated in Figure 1, *Pruner* can be modeled as a binary classifier aimed at deciding if candidate triples (q, d, S) are likely to belong to the set D_q^+ or to the set D_q^-. To this end, it is crucial for *Pruner* to precisely identify the documents in D_q^+ to avoid discarding potentially relevant documents and to maximize the efficiency boost deriving from early exiting irrelevant documents. However, since discarding relevant documents severely impact effectiveness, *Pruner* should implement a conservative strategy, putting in D_q^+ all the promising documents that are likely to be ranked among the first k.

Pruner can exploit both query-document features but also score-dependent features computed by $Ranker_{\text{pre}}$. The latter in particular could be extracted in a point-wise fashion, e.g.,

TABLE 1. List of symbols.

Symbol	Description	Symbol	Description
$d ; d_j$	Document	$q ; q_i$	Query
$(q, d) ; (q_i, d_j)$	Query-document pair	$S ; S(q, d)$	Full-model score of a pair
$s_l(q, d)$	Score of tree l for pair (q, d)	$\mathcal{E}$	Ensemble of regression trees
$\mathcal{E}_{pre}$	Portion of $\mathcal{E}$ before an Early Exit sentinel	$\mathcal{E}_{post}$	Portion of $\mathcal{E}$ after an Early Exit sentinel
$\mathcal{E}_{aux}$	Auxiliary ensemble used by $Ranker_{pre}$	$\mathcal{Q}$	Set of all queries
t_{pre}	Size of $Ranker_{pre}$	t_{post}	Size of $Ranker_{post}$
k	Top-k documents cutoff used by the metric	Δ	Change in metric
D_q	Candidate documents for query q	D_q^+	First k ranked documents by $\mathcal{E}$ for query q
D_q^-	Documents ranked after position k by $\mathcal{E}$ for query q	θ	Average tree traversing cost
c	Confidence threshold for LEAR	p	Proximity threshold for EPT

the raw score of each document, or query-wise, i.e., features derived from an overall evaluation of the scores on a per-query basis, by looking at all the candidate documents for the query and taking into account their scores and the induced ranking.

The document-level early exit problem can thus be thought as that of co-designing $Ranker_{pre}$ and *Pruner* to *i)* minimize the effectiveness drop resulting from erroneously filtering out documents in D_q^+; *ii)* maximize the efficiency gain by filtering out most of the documents belonging to D_q^-; and *iii)* minimize the computational cost of $Ranker_{pre}$ and *Pruner* as to avoid overheads wasting the advantages of the early exit strategy implemented.

In this work, we consider two different options for the implementation of $Ranker_{pre}$. One option is that of splitting the full ranking model $\mathcal{E}$ into $\mathcal{E}_{pre}$ and $\mathcal{E}_{post}$, where $\mathcal{E}_{pre}$ is a small sub-forest of $\mathcal{E}$ providing relevant scoring features to *Pruner*, whereas $\mathcal{E}_{post}$ is the remaining part of the whole forest $\mathcal{E}$ and is used by $Ranker_{post}$. This solution, adopted also by Cambazoglu et al. [5], allows to easily integrate the original ensemble $\mathcal{E}$ in the document-level early exit strategy. This is simply done by defining the position of a sentinel point splitting $\mathcal{E}$ in $\mathcal{E}_{pre}$ and $\mathcal{E}_{post}$. An orthogonal option is that of using, rather than an initial portion of the full forest until the sentinel, an auxiliary model $\mathcal{E}_{aux}$, i.e, an alternative compact ranking forest, specifically trained to provide high-quality early-exit signals with a reduced scoring cost to *Pruner*, and finally exploiting the full ranking model $\mathcal{E}$ as $Ranker_{post}$. We guess this auxiliary model can supply more reliable information to *Pruner*, thus allowing a more accurate pruning of non-promising documents.

For both solutions $\mathcal{E}_{pre}$ and $\mathcal{E}_{aux}$, i.e., $Ranker_{pre}$ implemented as either a prefix of the full ensemble or an auxiliary ranker, $Ranker_{pre}$ must be at the same time computationally efficient and effective in providing high-quality signals to the *Pruner* component. There is thus a trade-off in designing $Ranker_{pre}$. Indeed, implementing $Ranker_{pre}$ with a very compact ranking forest has the effect of providing to *Pruner* less informative features. On the other hand, a more precise model in terms of ranking quality would make the *Pruner* more effective, but it would impact negatively on the overall scoring cost.

In the following, let t_{pre} be the number of trees used by ranking ensemble $Ranker_{pre}$ and t_{post} the number of trees used by $Ranker_{post}$. If we use $\mathcal{E}_{pre}$ as $Ranker_{pre}$ model, t_{pre} is the sentinel point, and $t_{post} = n - t_{pre}$ is the size of $\mathcal{E}_{post}$. Alternatively, if we adopt $\mathcal{E}_{aux}$ as $Ranker_{pre}$ model, t_{pre} is the number of trees of the auxiliary ranker $\mathcal{E}_{aux}$ and $t_{post} = n$, i.e., $Ranker_{post}$ exploits the full ranking model $\mathcal{E}$.

We assume that the costs of traversing a single tree of both $\mathcal{E}$ and $\mathcal{E}_{aux}$ are similar, and we denote by θ this averaged cost. The per-query scoring cost $\mathcal{C}_F$ of the full ranking model $\mathcal{E}$ can be approximated with:

$$\mathcal{C}_F(q, D_q) = \sum_{d_q \in D_q} \|\mathcal{E}\| \cdot \theta = \|D_q\| \cdot n \cdot \theta \tag{2}$$

Let $D_q^* \subset D_q$ be the subset of promising documents identified by *Pruner*. It is worth noting that the perfect EE strategy would be ensured by an oracle always choosing $D_q^* = D_q^+$.

We denote by $\phi^q = \frac{\|D_q^*\|}{\|D_q\|}$ the fraction of candidate documents deemed as relevant for query q by the *Pruner*'s strategy, while b_c denotes the classification cost paid to apply *Pruner* to each candidate documents. The scoring cost $\mathcal{C}_{EE}$ resulting from the introduction of an early exit strategy can be approximated with:

$$\begin{aligned}\mathcal{C}_{EE}(q, D_q) &= \underbrace{\sum_{d_q \in D_q} t_{pre} \cdot \theta}_{Ranker_{pre}\ \text{cost}} + \underbrace{\sum_{d_q \in D_q} b_c}_{Pruner\ \text{cost}} + \underbrace{\sum_{d_q \in D_q^*} t_{post} \cdot \theta}_{Ranker_{post}\ \text{cost}} \\ &= \|D_q\| \cdot (t_{pre} \cdot \theta + b_c + \phi^q \cdot t_{post} \cdot \theta)\end{aligned} \tag{3}$$

where the first term models the cost for scoring all the documents by $Ranker_{pre}$, the second term represents the classification cost of *Pruner*, and the third term refers to the scoring cost of $Ranker_{post}$ only for the documents not filtered out by *Pruner*. The ratio between Equation 2 and Equation 3 models the speed-up achieved by the early exit strategy $\mathcal{S}$ compared to the cost of scoring with the full ensemble:

$$\text{speed-up}(q, D_q) = \frac{\mathcal{C}_F(q, D_q)}{\mathcal{C}_{EE}(q, D_q)} \tag{4}$$

Let us now suppose that the classification cost b_c can be expressed in terms of θ, for example, if the binary classifier

is implemented as an ensemble of b decision trees, with characteristics similar to that of the other ranking ensembles (i.e., similar depth and number of leaves). Thus, we have that $b_c = b \cdot \theta$. The speed-up can consequently be simplified as follows:

$$\text{speed-up}(q, D_q) = \frac{n}{t_{\text{pre}} + b + \phi^q \cdot t_{\text{post}}} \tag{5}$$

whereas the overall speedup for all the queries in a test set $\mathcal{Q}$ is obtained by macro-averaging as follows:

$$\text{speed-up}(\mathcal{Q}) = \frac{\sum_{q \in \mathcal{Q}} \|D_q\| \cdot n}{\sum_{q \in \mathcal{Q}} \|D_q\| \cdot (t_{\text{pre}} + b + \phi^q \cdot t_{\text{post}})} \tag{6}$$

We now perform a simple theoretical analysis of the possible speed-ups we can expect from the introduction of our EE strategy in the scoring process. We postpone to the experimental section a detailed discussion on the tradeoff between efficiency and effectiveness, i.e., the analysis of the impact of the observed speed-up, obtained the cascading pipeline in a real deployment, on the quality of the document ranking.

Considering Equation 5, we can exemplify the effects on the final speed-up of our ranking cascading by deeming realistic values for its hyper-parameters, namely t_{pre}, b, and ϕ^q. In this context, speed-up mainly depends on the proportion ϕ^q of promising documents (identified by *Pruner*) that must be fully scored by $Ranker_{\text{post}}$, along with the size t_{pre} of $Ranker_{\text{pre}}$. In particular, for the case $Ranker_{\text{pre}} = \mathcal{E}_{\text{pre}}$, the sentinel position t_{pre} plays an important role in our setting, as we can trade efficiency (lowering t_s) for the effectiveness of the whole ranking cascading, or vice versa (raising t_{pre}) to improve the precision of $Ranker_{\text{pre}}$ and feed *Pruner* with more reliable information. Note that a similar tradeoff also exists for the alternative setting $Ranker_{\text{pre}} = \mathcal{E}_{\text{aux}}$, although in that case, we have $t_{\text{post}} = n$, i.e., the size of $Ranker_{\text{post}}$ remains unchanged.

As an example of speed-up level we can achieve, let us suppose that the full ranking ensemble $\mathcal{E}$ includes $n = 1,000$ trees, while $Ranker_{\text{pre}}$ is made up of 100 trees (e.g., in case of $\mathcal{E}_{\text{pre}}$, the sentinel is placed at tree $t_{\text{pre}} = 100$ or, in case of $\mathcal{E}_{\text{aux}}$, it is composed of 100 trees). In addition, we suppose that *Pruner* selects 20% of D_q ($\phi^q = 0.2$) as promising ones, thus stopping the scoring of the remainder 80%. When we apply Equation 5, we consider negligible the cost of classification, as we assume that *Pruner* uses a very small number b of trees in comparison to the size of $\mathcal{E}$, i.e., $b \ll n$. We verified that for suitable values of b, such as $b = 10$, speed-up values lower of tiny amounts, only affecting the hundreds of these values.

For this setting, when $Ranker_{\text{pre}} = \mathcal{E}_{\text{pre}}$, we observe a speed-up equal to 3.6. The speedup lowers to 3.3× for the $\mathcal{E}_{\text{aux}}$ solution, although it is supposed to make *Pruner* more effective, thus possibly lowering the percentage of documents selected as promising and reversing the gap with $\mathcal{E}_{\text{pre}}$.

Finally, let us analyze the effect of adopting a much smaller model for $Ranker_{\text{pre}}$, specifically $t_{\text{pre}} = 50$ instead of $t_{\text{pre}} = 100$. We observe that speed-up raises from 3.6× to 4.2× for $\mathcal{E}_{\text{pre}}$, and from 3.3× to 4.0× for $\mathcal{E}_{\text{aux}}$, thus pushing for the adoption of a very compact model for $Ranker_{\text{pre}}$ if we only consider the overall efficiency, without trading-off with the effectiveness of the solution.

IV. LEAR: LEARNED EARLY EXIT RANKING

The first strategies for document early exit from deep ranking ensembles were proposed by [5]. They are based on a heuristic approach and can be used to implement a very efficient *Pruner* component in our ranking cascading. Indeed, we employ a solution inspired by [5] to realize a *baseline* for our ML-based solution for implementing *Pruner*. In this paper, we thus discuss in detail LEAR, an ML binary classifier used to implement *Pruner*. The aim of LEAR is to select a small fraction ϕ_q of the candidate documents D_q for continuing the scoring up to the end of the ensemble. Indeed, LEAR must precisely identify the documents in D_q^+, by hopefully discarding all the documents in D_q^-. The goal is to boost the speedup and provide a ranking quality as close as possible to the one achieved by fully evaluating all documents in D_q. Note that, along with the precision of the classifier, it is also important to keep its complexity under control to avoid vanishing the speed-up benefits deriving from document EE.

This design of LEAR poses the following challenges: *(i)* how to build the training set for the classifier; *(ii)* how to cope with imbalance of selected versus discarded documents; *(iii)* how to manage at the best the trade-off between efficiency and effectiveness.

Building the training set. The examples for training the binary classifier are generated by exploiting the LtR training dataset and the scoring information obtained from the full-model $\mathcal{E}$. We use *Continue* and *Exit* to label training instances, i.e., (q, d) pairs. Since the class *Continue* is the rarest one, and thus the dataset is unbalanced, in the following, we also refer to *Continue*/*Exit* classes as the positive/negative ones. We assign the *Continue* class label to all the documents in D_q^+ that are associated with positive relevance labels in the LtR dataset. The complementary documents assigned to the *Exit* class are all those in D_q^-, plus the documents in D_q^+ that are labeled as *not relevant* in the LtR dataset, if any. The *Continue* set, which includes only relevant documents, has a cardinality not larger than $|D_q^+|$, where $|D_q^+| \leq k$ (the cutoff value k is equal to 10 in our experimental setting). Conversely, the size of the *Exit* set is comparatively larger, as $|D_q|$ is typically much larger that k.

Note that since the *Continue* set includes only the relevant documents of D_q^+, a perfect *Pruner* trained on examples labeled in such a way might drop some irrelevant documents erroneously placed by $\mathcal{E}$ in D_q^+, thus possibly leading to a better ranking accuracy than the full ensemble (without EE).

To train *Pruner* with *Continue* and *Exit* examples, we use an augmented representation for the query-document pairs, by including information provided by $Ranker_{\text{pre}}$. Specifically, besides the features used by $\mathcal{E}$ for each pair (q, d), we use the following additional features:

1) the rank of document d computed by $Ranker_{\text{pre}}$;
2) the score of document d computed by $Ranker_{\text{pre}}$;
3) the per-query min-max normalized score value;
4) the number of candidates for query q, i.e, $|D_q|$.

This implies that to build a training set for LEAR, we have to select a portion of our LtR dataset (not used to train $\mathcal{E}$ or $\mathcal{E}_{\text{aux}}$) and feed $Ranker_{\text{pre}}$ (either $\mathcal{E}_{\text{pre}}$ or $\mathcal{E}_{\text{aux}}$) with this portion of the dataset so as to collect document scores and finally augment the feature set modeling (q, d).

Handling imbalance. We observe that in the dataset for training *Pruner* the *Continue* instances are less than 10% of the total examples, resulting in a highly imbalanced dataset, possibly hindering the performance of the binary classifier. This depends on the size of D_q in LtR datasets which is typically in the hundreds, along with the choice of the cutoff value $k = 10$ of quality metrics. Moreover, when using quality metrics such as NDCG@k, documents contribute differently to the quality of the resulting ranking, as it depends on document relevance labels and rank positions. We tackle this issue by exploiting a cost-sensitive policy for training *Pruner*, where an instance d, with relevance label r_d and classification label $l_d \in \{Continue, Exit\}$, is associated with a weight $w_d = 2^{r_d}/f_q(l_d)$, where $f_q(l_d)$ is the frequency, among the candidate documents D_q, of the classification label l_d. This pushes the classifier to prioritize loss reduction on documents with large relevance judgments (the weight is proportional to their contribution to the NDCG metric), and on the infrequent *Continue* documents (the weight is inversely proportional to the class frequency). Note that this weighting scheme is query-based and allows the training of *Pruner* to adapt to the characteristics of the different queries.

Efficiency vs. effectiveness trade-off. Accuracy is not the metric we target for the classifier. Our goal is in fact to maximise the recall over *Continue* documents without hindering precision. To this end, we fine-tune a threshold on the probability of belonging to class *Continue* predicted by *Pruner*. By varying this threshold, we can find the sweet spot between *precision* and *recall*. Finally, the quality of the $Ranker_{\text{pre}}$ model impacts the accuracy of the classifier and the efficiency of the LEAR framework. Less reliable score-dependent features, coming from very simple $Ranker_{\text{pre}}$ ranking models, potentially harm the classifier accuracy. On the other hand, they may produce large speedups thanks to the reduced scoring cost. In the experimental section, we investigate the impact of varying the complexity of $Ranker_{\text{pre}}$, both in terms of size and in using $\mathcal{E}_{\text{pre}}$ vs $\mathcal{E}_{\text{aux}}$, on the performance of the whole ranking system.

V. EXPERIMENTS

We conduct extensive experiments on two public LtR datasets to evaluate the impact of introducing LEAR into a state-of-the-art ranking pipeline, analyzing both the efficiency gain in terms of speed-up and the implications on the ranking quality. We also perform various studies to better understand LEAR and the hyper-parameters driving its performance.

A. EXPERIMENTAL SETUP

Datasets. We conduct our experiments on two publicly available datasets: the MSN-1[1] (Fold 1) and the ISTELLA[2] datasets. The MSN-1 dataset consists of 31,351 queries and 136 features extracted from 3,771,125 query-document pairs, while the ISTELLA dataset is composed of 33,018 queries and 220 features extracted from 10,454,629 query-document pairs. They thus differ in the number of average documents per query, ranging from the 120 of MSN-1 to the 317 of ISTELLA. The query-document pairs in both datasets are labeled with relevance judgments ranging from 0 (irrelevant) to 4 (perfectly relevant). ISTELLA comes with about 96% of non-relevant documents and a normal distribution among the relevant ones centered on label 2, while MSN-1 shows a power law distribution with 51% of non-relevant documents. Both the datasets have been split in *four* partitions with proportions 60%-15%-5%-20%:

1) the first partition is used to train the λ-Mart ranking model, namely the full ensemble $\mathcal{E}$ and $\mathcal{E}_{\text{aux}}$;
2) the second is the validation set for tuning the hyper-parameter of λ-Mart and for training the *Pruner* binary classifier;
3) the third is used to fine-tune *Pruner* for both LEAR and the competitors EE strategies;
4) finally, the fourth partition is used as test set to evaluate the efficiency and effectiveness of the ranking solutions exploiting the full model $\mathcal{E}$ as well as the various EE strategies considered.

Ranking models. The reference ranking models are trained with λ-Mart. We use the LightGBM framework.[3] We fine-tune its hyper-parameters by maximizing NDCG@10 and using the Bayesian approach provided by HyperOpt.[4] The hyper-parameters tuned are: learning rate, minimum number of instances in leaves, minimum sum of hessian, minimum gain to split, while the number of leaves and the max depth are fixed and are set to 64 and 8, respectively. The training process is early stopped when the ranking quality measured on the validation set does not show improvements for the last 100 trees. The resulting ensembles $\mathcal{E}$ have 1,129 and 1,481 trees for MSN-1 and ISTELLA, respectively. For $\mathcal{E}_{\text{aux}}$ we further restrict the size of the ensemble to be composed of at most 50 trees with a patience of 5 trees. The more compact $\mathcal{E}_{\text{aux}}$ ensemble means that, besides the number of trees, the main difference between the full model and $\mathcal{E}_{\text{aux}}$ is the learning rate, which is 0.05 for $\mathcal{E}$ and 0.32 for $\mathcal{E}_{\text{aux}}$, with all the other parameters being almost identical.

Competitor EE Strategy. We evaluate LEAR against the EPT heuristic strategy, i.e. the one achieving the best performance among those proposed in [5] and discussed in Section II. Consistently with the original proposal for EPT, we use $k = 10$. Once $Ranker_{\text{pre}}$ has produced the scores for

[1] http://research.microsoft.com/en-us/projects/mslr/
[2] http://blog.istella.it/istella-learning-to-rank-dataset/
[3] https://github.com/microsoft/LightGBM
[4] https://github.com/hyperopt/hyperopt

all documents in D_q (for any $q \in \mathcal{Q}$), these scores and the induced ranks are exploited by the EPT strategy. The decision about EE is made by combining the use of k and a proximity threshold that is computed offline. Indeed, only documents that are within the first k ranks as well as documents that are within a given proximity threshold with respect to the score of k-th document score continue in $Ranker_{\text{post}}$. To assess whether an Early Exit technique has a significant impact in terms of ranking quality we performed a Paired T-Test for Equivalence between the whole evaluation and the one employing Early Exit.

Assessing the efficiency. We drive the selection of the parameters for our EE ranking solution with the cost estimation model introduced in section III (see equations 2, 3 and 4). In the last evaluation phase, however, we assessed experimentally the efficiency of the various EE settings using QuickScorer (QS) [9], [18], the state-of-the-art algorithm for the traversal of ensembles of regression trees. To this purpose, we integrate in the QS scoring process the LEAR-based and EPT-based EE strategies, exploiting the three cascading components $Ranker_{\text{pre}}$ (either $\mathcal{E}_{\text{aux}}$ or $\mathcal{E}_{\text{pre}}$), *Pruner* (either LEAR or EPT), and $Ranker_{\text{post}}$ (either $\mathcal{E}$ or $\mathcal{E}_{\text{post}}$). We performed all experiments[5] on a single machine equipped with two Intel Xeon Platinum 8276L CPUs clocked at 2.20GHz for a total of 112 virtual cores. Memory-wise, the system has three layers of cache: 32 KB of distinct data and instruction memory for L1, 1024 KB for L2, and 3942 KB for L3, alongside 504MB of general RAM. In section V-B are reported and discussed the efficiency results computed by considering the total latency of the whole scoring process, i.e., the time needed to score the documents with $Ranker_{\text{pre}}$ (either based on $\mathcal{E}_{\text{pre}}$ or $\mathcal{E}_{\text{aux}}$ models) and $Ranker_{\text{post}}$ plus the time required to *Pruner* to decide about document EE.

B. RESEARCH QUESTIONS

Our experiments aim to answer the following research questions by evaluating the performance of LEAR under different and incremental testing scenarios so as to have a better understanding of the characteristics and hyper-parameters driving its utility:

- **RQ1**: Which is the best binary classification model for the *Pruner*?
- **RQ2**: Considering the case $Ranker_{\text{pre}} = \mathcal{E}_{\text{pre}}$, which is the best position in the ensemble where to place the sentinel? It is worth recalling that the earlier the placement of the sentinel, the highest will be the observed speed-up (see Equation 5), but on the contrary, the lower will be the expected classification accuracy. How does a solution based on which $Ranker_{\text{pre}} = \mathcal{E}_{\text{aux}}$, where $\mathcal{E}_{\text{aux}}$ is a small ensemble of trees of the same size as the smallest $\mathcal{E}_{\text{pre}}$, behave with respect to the alternative solution using a prefix of the whole ensemble $\mathcal{E}$, i.e., $Ranker_{\text{pre}} = \mathcal{E}_{\text{pre}}$?
- **RQ3**: How does LEAR compare to existing state-of-the-art algorithm for ensemble traversal, namely EPT, in terms of observed speed-up and possible drop in ranking quality?

[5] code available at https://github.com/hpclab/earlyexit-ltr

RQ1: WHAT IS THE BEST BINARY CLASSIFICATION MODEL TO ADOPT BY PRUNER FOR DOCUMENT EE?

Several options are available for building a binary classifier, e.g., Logistic Regression (LR), Support Vector Machine (SVM), Neural Network (NN), Gradient Boosted Decision Trees (GBDT), and many others. Note that the classification task performed for each document is a potential overhead that we introduce, according to the cost b_c modeled in Equation 3.

To answer this RQ, we limited our tests to a specific instance of our cascading ranker, where $Ranker_{\text{pre}} = \mathcal{E}_{\text{pre}}$ and $Ranker_{\text{post}} = \mathcal{E}_{\text{post}}$, and the sentinel is placed at tree $t_{\text{pre}} = 50$. In this context, we trained different classifiers of small complexity, suitable for our scenario, where the per-document overhead at inference time should be minimal for not impacting significantly on the overall efficiency. We used the features discussed in section IV with the same setup described in paragraph V-A. For each trained model, we also performed a grid search to find the best hyper-parameters of each model. Table 2 summarizes the performances of four different binary classifiers by evaluating them in terms of both Precision and Recall (typical of a classification problem), but also in terms of NDCG@10 (typical of a ranking problem). Precision and Recall are measured on the validation set by considering the *Continue* class as the positive (rare) one, and the *Exit* class as the negative one. On the other hand, NDCG@10 measures the effectiveness of the cascading ranker on the validation set, where Pruner is implemented as one of four classifiers. To produce the final ranking of documents in D_q before applying NDCG@10, we combine the scores provided by $\mathcal{E}_{\text{pre}}$ on the documents that early exit the pipeline, along with the scores provided by the $\mathcal{E}_{\text{post}}$ on the documents that continue on the pipeline. Indeed, we produce two separate ranked lists for documents marked as either *Exit* or *Continue*; then, the ranked list of the *Exit* documents is appended to the list of the *Continue* ones before computing the NDCG@10 metric. We note that, from the classification perspective, the Neural Network achieves the best Precision/Recall trade-off among the four methods tested. However, when we consider the ranking metric NDCG@10, the GBDT ensemble performs better.

This controversial result also derives from the training process that uses a dataset where a positive/continue class label is assigned to all relevant documents appearing in the top-k results returned by $\mathcal{E}$, without considering their degrees of relevance. In contrast, these different degrees are used to weigh the instances in the training dataset. Conversely, the NDCG metric fully exploits this information of nuanced relevance to evaluate a given ranking. So, due to the slightly lower recall of the NN solution in comparison to GBDT, NN forces some highly relevant documents to exit the cascading ranker, thus inducing a high penalization in

TABLE 2. Performances on the validation set of different choices for the LEAR classifier.

Model type	MSN-1				ISTELLA			
	NDCG@10	Precision	Recall	F1	NDCG@10	Precision	Recall	F1
GBRT Ensemble	0.5329	0.28	0.97	0.43	0.7281	0.26	0.97	0.41
Neural Network	0.5252	0.59	0.95	0.73	0.7194	0.44	0.98	0.61
Logistic Model	0.5253	0.43	0.92	0.59	0.7205	0.23	0.97	0.37
Linear SVM	0.5194	0.67	0.84	0.75	0.6871	0.62	0.80	0.70

TABLE 3. Performances on the validation set of several LEAR classifiers trained without weight balancing.

Model type	MSN-1				ISTELLA			
	NDCG@10	Precision	Recall	F1	NDCG@10	Precision	Recall	F1[6]
GBRT Ensemble	0.5143	0.71	0.44	0.54	0.6636	N.D.	0.00	N.D.
Neural Network	0.5133	0.61	0.50	0.55	0.6636	N.D.	0.00	N.D.
Logistic Model	0.4996	0.57	0.01	0.02	0.6636	N.D.	0.00	N.D.
Linear SVM	0.4572	0.32	0.08	0.13	0.6636	N.D.	0.00	N.D.

the NDCG metric. Finally, table 3 reports the performances of the same four classifiers, trained without the weighting schema of the instances, as discussed in Section III. We thus perform an ablation study, aimed at understanding the contribution of this strategy to the overall performance. Note that Precision/Recall as well as NDCG get much worse. For dataset ISTELLA, we observe a total collapse towards classifiers that predict only the *Exit* (majority) class. Hence, Precision (and F1 as a consequence) cannot be computed and we observe a Recall equal to 0.00 with respect to the positive/minority class *Continue*, as well as a much smaller NDCG. Since for this dataset all four classifiers, trained without imbalance mitigation, cause all the documents to exit, NDCG is equal to the one observed at the sentinel t_{pre}, but the speed-up is obviously maximum.

On the other hand, by analyzing the different models at inference time from an efficiency perspective, as already mentioned in Section V-A, we have very efficient algorithms to traverse ensembles of regression trees, so the cost b_c of running the classifier is very limited even for moderately sized GBRT ensembles. Furthermore, employing a tree-based classifier has the added benefit that it can be easily added to the scoring pipeline that always uses the same algorithm, namely QuickScorer in our scenario. To answer RQ1, we adopted a GBDT ensemble of $b = 10$ regression trees for classification purposes, given its higher recall/ranking quality and lower impact on the classification cost. In particular, as already observed from the analysis of Equation 5, with $b \ll n$, where $n > 1{,}000$ is the size of $\mathcal{E}$, the classification cost is almost negligible with respect to the final speed-up achieved by the whole ranking pipeline introduced by LEAR. The classification model adopted by *Pruner* was trained by optimizing the logistic loss and using the same implementation framework of λ-Mart (LightGBM + HyperOpt).

[6] Column set to N.D. due to undefined Precision.

RQ2: WHAT IS THE BEST POSITION IN THE ENSEMBLE WHERE TO PLACE THE SENTINEL? HOW DOES A CASCADING RANKER EXPLOITING A SMALL TREE ENSEMBLE $\mathcal{E}_{\text{AUX}}$ BEHAVE WITH RESPECT TO THE ALTERNATIVE SETTING BASED ON AN ENSEMBLE $\mathcal{E}_{\text{PRE}}$ OF COMPARABLE SIZE?

The first question of RQ2 still refers to a cascading ranker, which entails that $Ranker_{\text{pre}} = \mathcal{E}_{\text{pre}}$ and $Ranker_{\text{post}} = \mathcal{E}_{\text{post}}$. It aims at evaluating how the placing of the sentinel at different positions, with $t_{\text{pre}} \in \{50, 100, 200\}$, impacts on both LEAR and EPT. Indeed, by evaluating different sentinel positions, we aim to find the best tradeoff between speed-up and ranking quality. Regarding LEAR, for each t_{pre}, we build a different training set and train a GBDT classification model that fully exploits the rank-based and score-based features provided by $\mathcal{E}_{\text{pre}}$. For the same positions t_{pre}, we also evaluated EPT. Figure 2 and Figure 3 reports the performance of EPT and LEAR, respectively, in terms of speed-up (x-axis) and ranking quality NDCG@10 (y-axis). We recall all the tests, including the ones conducted for RQ1, were performed by exploiting the validation set because we first tune the hyper-parameter p of the ranking cascading technique before evaluating using the test set. Observe that the plots report a distinct curve per each t_{pre}. To obtain the plots for LEAR and EPT, we vary the classifier confidence and the proximity thresholds of the heuristic techniques using 20 evenly spaced points, in range [0.1, 0.9] and [0.3, 1.5] for LEAR and EPT respectively. In detail, going from left to right in the curves, we change confidence and threshold values to make both LEAR and EPT stricter in selecting the documents that have to continue: this implies that more documents early exit the cascading ranker, including relevant documents, thus reducing the ranking quality, but mainly increasing the overall speedup.

On the MSN-1 dataset (see Figure 2.a), the baseline heuristic EPT achieves the best performance by placing the sentinel at $t_{\text{pre}} = 200$, since for all the other options we observe a rapid degradation of ranking quality, even

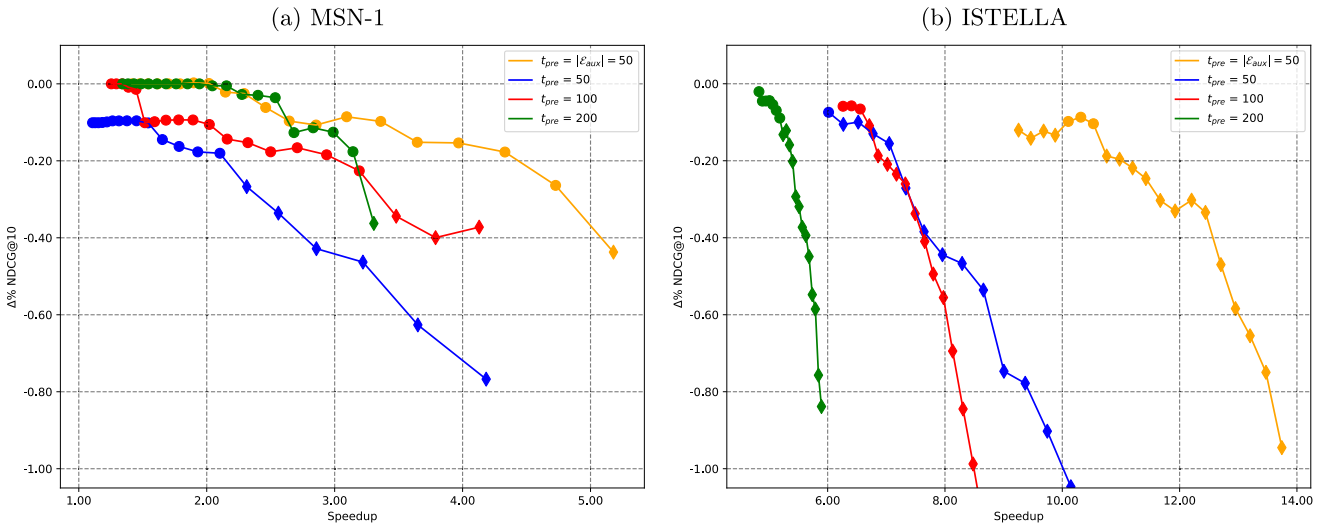


FIGURE 2. Estimated impacts on the validation set of different sentinel positions for EPT on MSN-1 and ISTELLA. Points are marked with a circle if there is statistical equivalence between the NDCG values of the EE solution and those from the baseline and with a diamond otherwise.

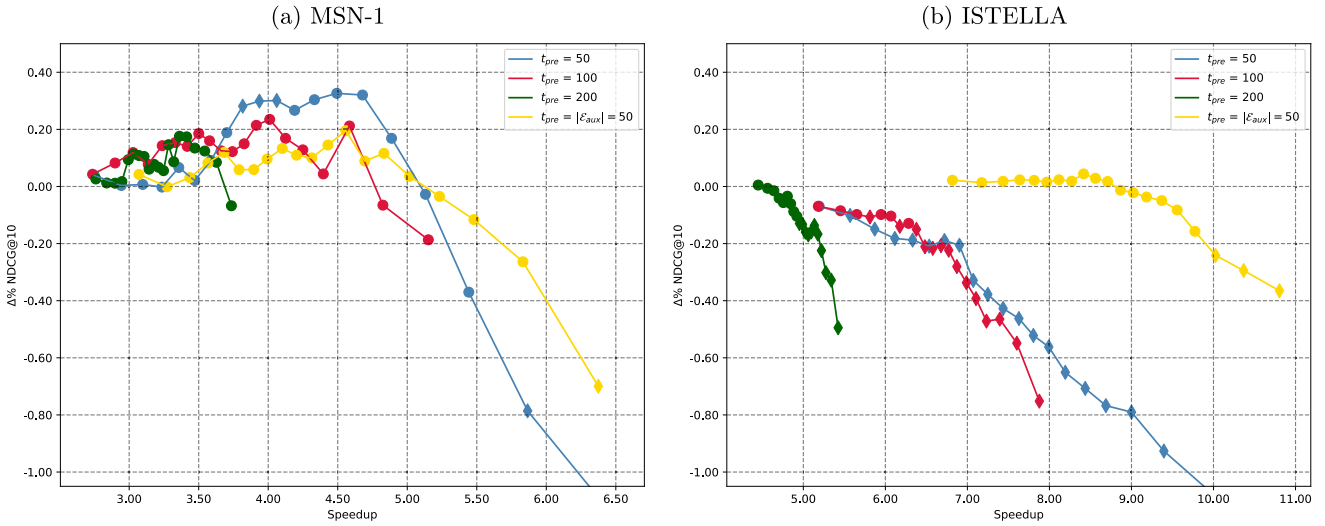


FIGURE 3. Estimated impacts on the validation set of different sentinel positions for LEAR on MSN-1 and ISTELLA. Points are marked with a circle if there is statistical equivalence between the NDCG values of the EE solution and those from the baseline and with a diamond otherwise.

with speedups less than 2@. On the ISTELLA dataset (see Figure 2.b) the best candidate appears to be $t_{\text{pre}} = 100$, since in the first part of the curve, for several threshold points, we are able to achieve good speedups (greater than 6x) with no significant difference in terms of NDCG compared to the baseline, in contrast with the other choices for t_{pre}. The LEAR technique on the MSN-1 dataset (see Figure 3.a) achieves the best performance by placing the sentinel at $t_{\text{pre}} = 50$, while on the ISTELLA dataset (see Figure 3.b) we observe a significant overlap between $t_{\text{pre}} = 50$ and $t_{\text{pre}} = 100$, with the latter again showing some threshold choices that induce no significant loss in NDCG.

We can now explore the second part of this research question, i.e., the possible advantages in quality/speedup of a solution based on $\mathcal{E}_{\text{aux}}$ compared with the alternative setting based on $\mathcal{E}_{\text{pre}}$. For EPT_{aux} (see Figure 2), on MSN-1 we can observe no significant degradation of NDCG up to a speedup of about 4.75@, while on the ISTELLA dataset the maximum speedup without significant degradation raises

TABLE 4. Summary of parameter selection.

EE Strategy	MSN-1		ISTELLA	
	t_{pre}	threshold	t_{pre}	threshold
$\text{EPT}_{\text{prefix}}$	200	0.42	100	1.50
EPT_{aux}	50	0.36	50	1.25
$\text{LEAR}_{\text{prefix}}$	50	0.65	100	0.31
LEAR_{aux}	50	0.61	50	0.44

TABLE 5. Precision, Recall and F1 of EPT and LEAR, exploiting either $\mathcal{E}_{\text{aux}}$ or $\mathcal{E}_{\text{pre}}$, on MSN-1 and ISTELLA.

Method	Class	MSN-1			ISTELLA		
		Precision	Recall	F1	Precision	Recall	F1
$\text{EPT}_{\text{prefix}}$	Exit	1.00	0.86	0.92	1.00	0.93	0.96
	Continue	0.30	0.99	0.46	0.24	0.99	0.39
EPT_{aux}	Exit	1.00	0.88	0.94	1.00	0.96	0.98
	Continue	0.33	0.97	0.50	0.35	0.99	0.52
$\text{LEAR}_{\text{prefix}}$	Exit	1.00	0.87	0.93	1.00	0.93	0.96
	Continue	0.31	0.95	0.46	0.24	0.99	0.39
LEAR_{aux}	Exit	1.00	0.88	0.94	1.00	0.94	0.97
	Continue	0.33	0.97	0.49	0.30	1.00	0.46

TABLE 6. Best strategies in [5] vs Full and LEAR (PREFIX and AUX) on MSN-1. Equivalent measures with Full are highlighted (p-value = 0.05).

Method	NDCG@10	Δ	Speedup	QS	μ_q	σ_q
Full	0.5233	-0.00%	1.00×	1.00×	119.51	71.83
$\text{EPT}_{\text{prefix}}$ ($t_{pre} = 200$, $p = 0.42$)	0.5226	-0.14%	3.00×	1.74×	22.70	9.94
$\text{LEAR}_{\text{prefix}}$ ($t_{pre} = 50$, $c = 0.65$)	0.5226	-0.13%	4.50×	3.14×	21.17	8.41
EPT_{aux} ($t_{pre} = 50$, $p = 0.36$)	0.5225	-0.16%	4.75×	3.78×	19.85	7.69
LEAR_{aux} ($t_{pre} = 50$, $c = 0.61$)	0.5231	-0.03%	4.71×	3.50×	20.07	6.88

to 10.36@. The same methodology for LEAR_{aux} (see Figure 3) allows to observe a speedup of 4.71@on MSN-1 and 9.05@on ISTELLA.

In general, the auxiliary model $\mathcal{E}_{\text{aux}}$ permits larger speedups without hindering too much the ranking quality, proving the benefit of adopting an alternative compact ranking forest as a $\textit{Ranker}_{\text{pre}}$ model for both the EPT and LEAR EE strategies. This is due to the better stability of the sub-optimal $\mathcal{E}_{\text{aux}}$ scores compared to the ones produced by $\mathcal{E}_{\text{pre}}$.

To complete the answer to RQ2 and conclude the discussion on Pruner placement and threshold selection, we report in Table 4 the thresholds (either of proximity or confidence) that will be used for the remainder of this paper. The values chosen are the ones that resulted in the highest NDCG among the ones described in Figures 2 and 3.

RQ3: PERFORMANCE OF LEAR COMPARED TO EPT AND STATE-OF-THE-ART TRAVERSAL ALGORITHMS

Table 5 reports the performance in terms of precision, recall and F1 measures of the LEAR and EPT classifiers trained using $\mathcal{E}_{\text{pre}}$ and $\mathcal{E}_{\text{aux}}$ respectively on the two datasets. Although EPT is not tuned to guess the *Exit/Continue* labels as LEAR, and thus is not properly a classifier, this allows us to contrast the performance of LEAR against EPT in the early-exit setting.

By comparing the two LEAR variants exploiting $\mathcal{E}_{\text{pre}}$ and $\mathcal{E}_{\text{aux}}$, we observe that both exhibit a high recall for the *Continue* class, with the latter outperforming the former (95% vs. 97% on MSN-1, and 99% vs. almost 100% on ISTELLA, respectively for $\text{LEAR}_{\text{prefix}}$ and LEAR_{aux}). This implies that both classifiers are able to correctly identify almost all documents that should continue the forest traversal, as they will be included in the top-k results. We remind that having a large recall on the *Continue* class is necessary for a high-quality final ranking.

The second objective of the LEAR classifiers is to minimize the number of false positives, i.e., the number of *Exit* documents incorrectly classified as *Continue*. These misclassified documents do not contribute to the final top-k results and only add to the overall evaluation cost. In this regard, the recalls on the *Exit* class are 87% and 88% for MSN-1 and 93% and 94% for ISTELLA in the case of $\text{LEAR}_{\text{prefix}}$ and LEAR_{aux}, respectively. This finding highlights that both LEAR variants early exit the vast majority of the irrelevant documents. We remind you that in both datasets, the irrelevant documents are more than 90% of the total. In Table 5, we compared the classification performances of EPT and LEAR. Interestingly, both EPT variants show similar results to LEAR, with the AUX approach showing the best performances on both datasets. In particular, we can observe that EPT_{aux} is the best classifier among the four options, achieving the highest F1 scores for the two classes in both MSN-1 and ISTELLA.

We now compare the performance of LEAR and EPT by assessing the trade-off between efficiency and effectiveness when both solutions are employed within a state-of-the-art traversal algorithm. Table 6 reports results for MSN-1 and Table 7 for ISTELLA. In both tables, the fourth column indicates the theoretical speedup calculated using Equation 6. In contrast, the fifth column shows the speedup observed when implementing Early Exit within QuickScorer (QS). The reduced speedups observed in the real-world scenario can be attributed to the omission of hardware-related overhead

TABLE 7. Best strategies in Cambazoglu et al. [5] vs Full and LEAR (PREFIX and AUX) on ISTELLA. Equivalent measures with Full are highlighted (p-value = 0.05).

Method	NDCG@10	Δ	Speedup	QS	μ_q	σ_q
Full	0.7320	-0.00%	1.00×	1.00×	319.32	126.44
$\text{EPT}_{\text{prefix}}$ ($t_{pre} = 100$, $p = 1.50$)	0.7311	-0.12%	6.47×	5.51×	29.80	15.39
$\text{LEAR}_{\text{prefix}}$ ($t_{pre} = 100$, $c = 0.31$)	0.7311	-0.12%	6.18×	5.14×	30.00	7.81
EPT_{aux} ($t_{pre} = 50$, $p = 1.25$)	0.7308	-0.16%	10.36×	9.37×	20.03	8.26
LEAR_{aux} ($t_{pre} = 50$, $c = 0.44$)	0.7311	-0.12%	9.05×	7.70×	24.49	6.00

costs in the initial estimates. The last two columns, namely μ_q and σ_q, highlight respectively the average number of *Continue* documents selected by the classifier and its standard deviation. We remind that by increasing the LEAR confidence threshold, the EE strategy becomes more aggressive. This translates into higher speedups at the cost of higher degradation of the ranking quality. On the other hand, the proximity threshold of EPT has an opposite behavior, i.e., its increase implies a more conservative strategy that results in a better preservation of the ranking quality at the expense of a higher evaluation cost.

Results reported in Table 6 related to the MSN-1 dataset show that $\text{LEAR}_{\text{prefix}}$ and $\text{EPT}_{\text{prefix}}$ behave similarly regarding NDCG. However, the LEAR solution is almost two times faster, with a measured speedup of 3.14@versus 1.74@of EPT. On the other hand, the LEAR approach exploiting $\mathcal{E}_{\text{aux}}$ shows a statistical equivalent NDCG with the baseline with an observed speedup of 3.50@, approximately 0.4@faster than its counterpart using the PREFIX approach. EPT however balances a slightly lower effectiveness with a slight gain in efficiency. Table 7 reports the results on the ISTELLA dataset. We observe all the solutions behave similarly in terms of effectiveness, with a marginal drop with respect to the baseline. However, both the LEAR and EPT approaches exploiting $\mathcal{E}_{\text{aux}}$ achieve much higher speedups, with an increase of 2.5@between $\text{LEAR}_{\text{prefix}}$ and LEAR_{aux} and of 3.8@between $\text{EPT}_{\text{prefix}}$ and EPT_{aux}. The reason for this big efficiency gap is to be found in the reduced number of *Continue* documents selected by the classifiers exploiting $\mathcal{E}_{\text{aux}}$ compared to $\mathcal{E}_{\text{pre}}$. To answer RQ3, on the MSN-1 dataset LEAR consistently outperforms EPT, showcasing a 1.5@higher speedup when using $\mathcal{E}_{\text{pre}}$ and resulting in an equal measure of NDCG. On the other hand, when employing $\mathcal{E}_{\text{aux}}$, we observe close values of efficiency gains, but we LEAR_{aux} achieves statistically equivalent NDCG values to the *Full* ranking evaluation. In contrast, when we shift our focus to the ISTELLA dataset, we observe that EPT slightly outperforms LEAR with both PREFIX and AUX strategies. The main novelty however is that the introduction of $\mathcal{E}_{\text{aux}}$ is always beneficial, regardless of which Early Exit strategy is used. Specifically, on the MSN-1 dataset we observe an increase of speedup ranging from 0.36 for LEAR up to 2.04 for EPT, with equal or better performances in terms of NDCG. Conversely, on ISTELLA we observe a considerable increase in speedups, with gains ranging from 2.56 for LEAR to 3.86 for EPT, similarly to MSN-1 we again observe little to no change of NDCG.

Finally, Figure 4 reports the feature importance analysis of the LEAR classifier, including the rank- and score-based features, exploiting $\mathcal{E}_{\text{pre}}$ and $\mathcal{E}_{\text{aux}}$, respectively. In both analyses, the feature that contributes the most in terms of gain is the normalized score, while the most used is the rank. Both these observations suggest that information about the *quality* of a document is fundamental for detecting the top-k documents. Another interesting aspect is that three out of four added features appear in the top-10 most important features, with only the query size being left out, confirming their invaluable importance for the classification capabilities of LEAR.

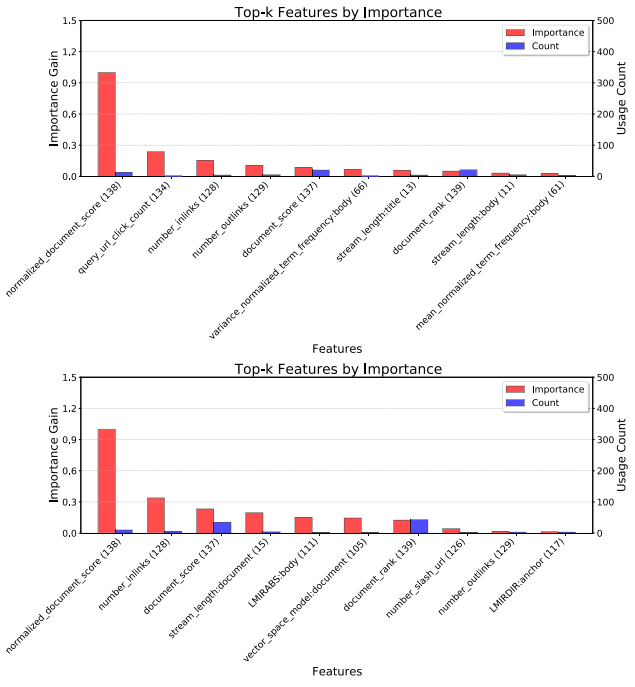


FIGURE 4. Feature importance for the $\text{LEAR}_{\text{prefix}}$ (top) and the LEAR_{aux} (bottom) classifiers on MSN-1.

VI. CONCLUSION

Results from this study have led to a deeper understanding of early exit strategies applied to document scoring through additive ensembles of regression trees. We have investigated

the early exit problem in the framework of a cascading ranker made of three components, where the middle one, *Pruner*, is in charge of the early exit strategy. One of the main results of this paper is that training and using a specialized ranker AUX for the first stage of the cascade allows to provide *Pruner* with good signals to make decisions about document early exit. We have also discussed in-depth alternative solutions for *Pruner*. In particular, we showed how to fine-tune EPT and train LEAR, a machine-learned classifier aimed to precisely identify documents that are unlikely to be ranked among the final top-k results, and thus that should early exit the ensemble to improve efficiency.

Experiments on two public datasets have shown that the early exit strategies studied achieve speedups ranging from $3\times$ to $10\times$, with a ranking quality measured in NDCG @10 statistically equivalent to the one of the baseline completing the scoring process for all the candidate documents. Our experiments showed that LEAR works better than EPT when we adopt $\mathcal{E}_{\text{pre}}$, a prefix of the original tree ensemble, as the first stage of the cascading ranker. Additionally, we highlighted that using AUX, an auxiliary machine-learned ranker explicitly trained to feed *Pruner*, significantly improves the performances of both LEAR and EPT.

REFERENCES

[1] T. G. Armstrong, A. Moffat, W. Webber, and J. Zobel, "Improvements that don't add up: Ad-hoc retrieval results since 1998," in *Proc. CIKM*. New York, NY, USA: Association for Computing Machinery, 2009, pp. 601–610.

[2] N. Asadi and J. Lin, "Training efficient tree-based models for document ranking," in *Advances in Information Retrieval*. Berlin, Germany: Springer, 2013, pp. 146–157.

[3] C. J. Burges, "From ranknet to lambdarank to lambdamart: An overview," *Learning*, vol. 11, nos. 23–581, p. 81, 2010.

[4] F. Busolin, C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, and S. Trani, "Learning early exit strategies for additive ranking ensembles," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.* New York, NY, USA: ACM, Jul. 2021, pp. 2217–2221.

[5] B. B. Cambazoglu, H. Zaragoza, O. Chapelle, J. Chen, C. Liao, Z. Zheng, and J. Degenhardt, "Early exit optimizations for additive machine learned ranking systems," in *Proc. WSDM*, 2010, pp. 411–420.

[6] G. Capannini, C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, and N. Tonellotto, "Quality versus efficiency in document scoring with learning-to-rank models," *Inf. Process. Manage.*, vol. 52, no. 6, pp. 1161–1177, Nov. 2016.

[7] R. Chen, L. Gallagher, R. Blanco, and J. S. Culpepper, "Efficient cost-aware cascade ranking in multi-stage retrieval," in *Proc. SIGIR*, N. Kando, T. Sakai, H. Joho, H. Li, A. P. de Vries, and R. W. White, Eds., 2017, pp. 445–454.

[8] N. Craswell, W. B. Croft, J. Guo, B. Mitra, and M. de Rijke, "Report on the SIGIR 2016 workshop on neural information retrieval (Neu-IR)," *ACM SIGIR Forum*, vol. 50, no. 2, pp. 96–103, Feb. 2017, doi: 10.1145/3053408.3053425.

[9] D. Dato, C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, N. Tonellotto, and R. Venturini, "Fast ranking with additive ensembles of oblivious and non-oblivious regression trees," *ACM Trans. Inf. Syst.*, vol. 35, no. 2, pp. 1–31, Apr. 2017.

[10] D. DeCoste, "Anytime interval-valued outputs for kernel machines: Fast support vector machine classification via distance geometry," in *Proc. 19th Int. Conf. Mach. Learn. (ICML)*, Sydney, NSW, Australia, Jan. 2002, pp. 99–106.

[11] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. ICML*, vol. 96, 1996, pp. 148–156.

[12] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, Oct. 2001.

[13] L. Gallagher, R. Chen, R. Blanco, and J. S. Culpepper, "Joint optimization of cascade ranking models," in *Proc. WSDM*, J. S. Culpepper, A. Moffat, P. N. Bennett, and K. Lerman, Eds., 2019, pp. 15–23.

[14] A. Gigli, C. Lucchese, F. M. Nardini, and R. Perego, "Fast feature selection for learning to rank," in *Proc. ICTIR*. New York, NY, USA: ACM, 2016, pp. 167–170.

[15] J. Guo, Y. Fan, L. Pang, L. Yang, Q. Ai, H. Zamani, C. Wu, W. B. Croft, and X. Cheng, "A deep look into neural ranking models for information retrieval," *Inf. Process. Manage.*, vol. 57, no. 6, Nov. 2020, Art. no. 102067.

[16] F. Lettich, C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, N. Tonellotto, and R. Venturini, "Parallel traversal of large ensembles of decision trees," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 9, pp. 2075–2089, Sep. 2019.

[17] M. Li, X. Zhang, J. Xin, H. Zhang, and J. Lin, "Certified error control of candidate set pruning for two-stage relevance ranking," 2022, *arXiv:2205.09638*.

[18] C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, N. Tonellotto, and R. Venturini, "QuickScorer: A fast algorithm to rank documents with additive ensembles of regression trees," in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Aug. 2015, pp. 73–82.

[19] C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, F. Silvestri, and S. Trani, "Post-learning optimization of tree ensembles for efficient ranking," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2016, pp. 949–952.

[20] C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, and S. Trani, "Query-level early exit for additive Learning-to-Rank ensembles," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2020, pp. 2033–2036.

[21] C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, and N. Tonellotto, "Speeding up document ranking with rank-based features," in *Proc. SIGIR*. New York, NY, USA: Association for Computing Machinery, 2015, pp. 895–898.

[22] C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, F. Silvestri, and S. Trani, "X-CLE a VER: Learning ranking ensembles by growing and pruning trees," *ACM Trans. Intell. Syst. Technol.*, vol. 9, no. 6, pp. 1–26, Nov. 2018.

[23] C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, and S. Trani, "X-DART: Blending dropout and pruning for efficient learning to rank," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Aug. 2017, pp. 1077–1080.

[24] L. Pang, Y. Lan, J. Guo, J. Xu, J. Xu, and X. Cheng, "DeepRank: A new deep architecture for relevance ranking in information retrieval," in *Proc. 2017 ACM Conf. Inf. Knowl. Manage. (CIKM)*. New York, NY, USA: Association for Computing Machinery, 2017, pp. 257–266, doi: 10.1145/3132847.3132914.

[25] L. Soldaini and A. Moschitti, "The cascade transformer: An application for efficient answer sentence selection," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics, Student Research Workshop*. Seattle, WA, USA: Association for Computational Linguistics, Jul. 2020, pp. 5697–5708.

[26] N. Tax, S. Bockting, and D. Hiemstra, "A cross-benchmark comparison of 87 learning to rank methods," *Inf. Process. Manage.*, vol. 51, no. 6, pp. 757–772, Nov. 2015.

[27] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, May 2004.

[28] L. Wang, J. Lin, and D. Metzler, "Learning to efficiently rank," in *Proc. SIGIR*. New York, NY, USA: ACM, 2010, pp. 138–145.

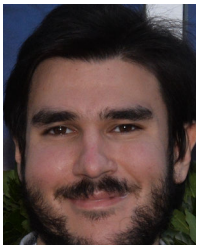
[29] Q. Wu, C. J. C. Burges, K. M. Svore, and J. Gao, "Adapting boosting for information retrieval measures," *Inf. Retr.*, vol. 13, no. 3, pp. 254–270, Jun. 2010.

[30] J. Xin, R. Tang, Y. Yu, and J. Lin, "BERxiT: Early exiting for BERT with better fine-tuning and extension to regression," in *Proc. 16th Conf. Eur. Chapter Assoc. Comput. Linguistics, Main Volume*, 2021, pp. 91–104.

[31] J. Xin, R. Tang, J. Lee, Y. Yu, and J. Lin, "DeeBERT: Dynamic early exiting for accelerating BERT inference," in *Proc. ACL*, D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, Eds. Stroudsburg, PA, USA: Association for Computational Linguistics, 2020, pp. 2246–2251.

[32] Z. Xu, M. Kusner, K. Weinberger, and M. Chen, "Cost-sensitive tree of classifiers," in *Proc. Mach. Learn. Res.*, Atlanta, GA, USA, Jun. 2013, pp. 133–141, vol. 28, no. 1.

[33] T. Ye, H. Zhou, W. Y. Zou, B. Gao, and R. Zhang, "RapidScorer: Fast tree ensemble evaluation by maximizing compactness in data level parallelization," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 941–950.
[34] W. Zhu, "LeeBERT: Learned early exit for BERT with cross-level optimization," in *Proc. ACL*. Stroudsburg, PA, USA: Association for Computational Linguistics, Aug. 2021, pp. 2968–2980.



FRANCESCO BUSOLIN is currently pursuing the Ph.D. degree with the Ca' Foscari University of Venice, Italy, under the supervision of Prof. Salvatore Orlando. His Ph.D. research aims to reduce the effort ranking pipelines need to perform to obtain adequate results to present to the final users. He is a Research Associate with the Italian National Research Council (CNR) in which he collaborates with the High-Performance Computing Laboratory (HPC Lab), Institute of Information Science and Technologies "Alessandro Faedo." His primary research interest includes information retrieval, particularly efficient learning to rank.

CLAUDIO LUCCHESE is currently a Professor with the Ca' Foscari University of Venice, where he is a member of the Data Mining and Information Retrieval Laboratory. Since 2018, he has been delegated as the Head of the Department for research activities. His main research interests include information retrieval, explainable AI, and data mining. He has published more than 100 papers on these topics in peer-reviewed international journals, conferences, and other venues. He was a recipient of the Best Paper Award from the ACM SIGIR Conference on Research and Development in Information Retrieval, in 2015.

FRANCO MARIA NARDINI is currently a Senior Researcher with the National Research Council (CNR), Italy. He has authored more than 80 papers in peer-reviewed international journals and conferences. His research interests include machine learning and web information retrieval. He received the Best Paper Award from ACM SIGIR, in 2015.

SALVATORE ORLANDO received the M.Sc. and Ph.D. degrees in computer science from the University of Pisa, in 1985 and 1991, respectively. He is a Full Professor with the Ca' Foscari University of Venice, where he is currently the Department Head. He is also a member of the Data Mining and Information Retrieval Laboratory, Ca' Foscari University of Venice. His research interests include data and web mining, information retrieval, and parallel/distributed systems. He has published more than 160 papers in journals and conference proceedings on these topics. He received the Best Paper Award from ACM SIGIR, in 2015.

RAFFAELE PEREGO is currently the Research Director with the Institute of Information Science and Technologies (ISTI), National Research Council (CNR). His main research interests include efficient algorithms and machine learning techniques for managing, analyzing, and searching large amounts of data. He has published more than 200 papers on these topics in peer-reviewed journals and proceedings of international conferences. He chaired ACM SIGIR, in 2016, and ECIR, in 2021. He serves in the senior program committees of the top-tier conferences in the research areas, such as ACM SIGIR, ACM CIKM, ACM WSDM, ECIR, and WWW, and the Ph.D. Board of the Italian National Ph.D. Program on Artificial Intelligence.

SALVATORE TRANI received the Ph.D. degree in computer science from the University of Pisa, in 2017. He is currently a Researcher with the National Research Council, Italy. His main research interests include information retrieval, web mining, and machine learning. He has authored more than 20 papers on these topics, published in peer-reviewed international journals and conferences. He serves regularly on the program committees of the top-tier conferences in the research areas.

Open Access funding provided by 'Università Ca' Foscari Venezia' within the CRUI CARE Agreement