Is Interpretable Machine Learning Effective at Feature Selection for Neural Learning-to-Rank?

Lijun Lyu[1 [0000−0002−7268−4902]], Nirmal Roy[1 [0000−0003−0860−5269]], Harrie Oosterhuis[2 [0000−0002−0458−9233]], and Avishek Anand[1 [0000−0002−0163−0739]]

[1] Delft University of Technology, Delft, The Netherlands
{L.Lyu, N.Roy, Avishek.Anand}@tudelft.nl
[2] Radboud University, Nijmegen, The Netherlands
harrie.oosterhuis@ru.nl

Abstract. Neural ranking models have become increasingly popular for real-world search and recommendation systems in recent years. Unlike their tree-based counterparts, neural models are much less interpretable. That is, it is very difficult to understand their inner workings and answer questions like *how do they make their ranking decisions?* or *what document features do they find important?* This is particularly disadvantageous since interpretability is highly important for real-world systems. In this work, we explore feature selection for neural learning-to-rank (LTR). In particular, we investigate six widely-used methods from the field of interpretable machine learning (ML) and introduce our own modification, to select the input features that are most important to the ranking behavior. To understand whether these methods are useful for practitioners, we further study whether they contribute to efficiency enhancement. Our experimental results reveal a large feature redundancy in several LTR benchmarks: the local selection method TabNet can achieve optimal ranking performance with less than 10 features; the global methods, particularly our G-L2x, require slightly more selected features, but exhibit higher potential in improving efficiency. We hope that our analysis of these feature selection methods will bring the fields of interpretable ML and LTR closer together.

1 Introduction

Learning-to-rank (LTR) is at the core of many information retrieval (IR) and recommendation tasks [38]. The defining characteristic of LTR, and what differentiates it from other machine learning (ML) areas, is that LTR methods aim to predict the optimal ordering of items. This means that LTR methods are not trying to estimate the exact relevance of an item, but instead predict relative relevance differences, i.e., whether it is more or less relevant than other items. Traditionally, the most widely adopted and prevalent LTR methods were based on Gradient Boosted Decision Trees (GBDT) [12, 29, 65]. However, in recent years, neural LTR methods have become increasingly popular [21, 46, 47].

Recently, [51] have shown that neural models can provide ranking performance that is comparable, and sometimes better, than that of state-of-the-art GBDT LTR models on established LTR benchmark datasets [14, 17, 49]. It thus seems likely that the prevalence of neural LTR models will only continue to grow in the foreseeable future.

Besides the quality of the results that ranking systems return, there is an increasing interest in building trustworthy systems through interpretability, e.g., by understanding which features contribute the most to ranking results. Additionally, the speed at which results are provided is also highly important [3, 5, 7]. Users expect ranking systems to be highly responsive and previous work indicates that even half-second increases in latency can contribute to a negative user experience [7]. A large part of ranking latency stems from the retrieval and computation of input features for the ranking model. Consequently, *feature selection* for ranking systems has been an important topic in the LTR field [22, 23, 45, 48, 52, 59, 68]. These methods reduce the number of features used, thereby helping users understand and greatly reduce latency and infrastructure costs, while maintaining ranking quality as much as possible. In line with the history of the LTR field, existing work on feature selection has predominantly focused on GBDT and support-vector-machine (SVM) ranking models [21, 26], but has overlooked neural ranking models. To the best of our knowledge, only two existing works have looked at feature selection for neural LTR [48, 52]. This scarcity is in stark contrast with the importance of feature selection and the increasing prevalence of neural models in LTR.

Outside of the LTR field, feature selection for neural models has received much more attention, for the sake of efficiency [35,36], and also to better understand the model behaviours [4, 72]. Those methods mainly come from the *interpretable* ML field [18,44], where the idea is that the so-called *concrete* feature selection can give insights into what input information a ML model uses to make decisions. This tactic has already been successfully applied to natural language processing [71], computer vision [6], and tabular data [4, 69]. Accordingly, there is a potential for these methods to also enable *embedded feature selection* for neural LTR models, where the selection and prediction are optimized simultaneously. However, the effectiveness of these interpretable ML methods for LTR tasks is currently unexplored, and thus, it remains unclear whether their application can translate into useful insights for LTR practitioners.

The goal of this work is to investigate whether six prevalent feature selection methods – each representing one of the main branches of interpretable ML field – can be applied effectively to neural LTR. In addition, we also propose a novel method with minor modifications. Our aim is to bridge the gap between the two fields by translating the important concepts of the interpretable ML field to the LTR setting, and by demonstrating how interpretable ML methods can be adapted for the LTR task. Moreover, our experiments consider whether these methods can bring efficiency into the practical application by reducing irrelevant input features for neural ranking models.

Our results reveal a large feature redundancy in LTR benchmark datasets, but this redundancy can be understood differently for interpretability and for efficiency: For understanding the model, feature selection can vary per document and less than 10 features are required to approximate optimal ranking behavior. In contrast, for practical efficiency purposes, the selection should be static, and then 30% of features are needed. We conclude that – when adapted for the LTR task – not all, but a few interpretable ML methods lead to effective and practical feature selection for neural LTR.

To the best of our knowledge, this is the first work that extensively studies embedded feature selection for neural LTR. We hope our contributions bring more attention to the potential of interpretable ML for IR field. To stimulate future work and enable reproducibility, we have made our implementation publicly available at: `https://github.com/GarfieldLyu/NeuralFeatureSelectionLTR` (MIT license).

2 Related Work

Learning-to-Rank (LTR). Traditional LTR algorithms mainly rely on ML models, such as SVMs and decision trees to learn the correlation between numerical input features and human-annotated relevance labels [15, 20, 27, 31, 37, 65, 67, 70]. Neural approaches [10, 11, 13, 54, 61, 66] have also been proposed, but did not show significant improvements over traditional non-neural models. Inspired by the transformer architecture [63], recent works have also adapted self-attention [46, 47, 51] and produced the neural LTR methods that outperform LambdaMART [65], albeit with a relatively small difference. It shows that neural rankers can provide competitive performance, consequently, the interest and effort towards neural models for LTR are expected to increase considerably in the near future.

Efficiency is crucial in real-world systems since users expect them to be highly responsive [3, 5, 7]. Aside from model execution, the latency of ranking system is largely due to feature construction, as it happens *on-the-fly* for incoming queries. Thus, efficiency is often reached by reducing (expensive) features. Previous works [21, 64] apply a cascading setup to reduce the usage of expensive features. Another growing trend in LTR is to design *interpretable models*. Existing methods rely on specific architecture design, such as general additive model (GAM) [73] or a feature-interaction constrained and depth-reduced tree model [39].

Feature Selection for LTR. Feature selection can achieve both efficiency and interpretability [4, 35, 36, 72]. By selecting a subset of input features, the input complexity of models is reduced while maintaining competitive performance. This helps with (1) efficiency as it avoids unnecessary construction of features [35, 36], and (2) interpretability as fewer input features are involved in prediction [4, 72].

Existing feature selection methods in LTR are classified commonly as *filter*, *wrapper* and *embedded* methods [22, 23, 48]. Filter and wrapper methods are applied to given static ranking models which are not updated in any way; filter methods are model-agnostic [22] while wrapper methods are designed

for a particular type of model [23]. In this work we will focus on the third category, embedded methods, where feature selection is performed simultaneously with model optimization. Most embedded methods are limited to particular model designs such as SVMs [32–34] or decision trees [40, 45, 68]. To the best of our knowledge, only two methods are designed for neural LTR [48, 52]: one applies group regularization methods [52] to reduce both input and other model parameters; the other [48] uses the gradients of a static ranking model to infer feature importance, and thus it belongs to the *filter* category. We do not investigate these two methods further, as the focus of this work is on *embedded* input feature selection methods.

Interpretable Machine Learning. The earliest work in interpretable ML attempted to explain a trained model in *post-hoc* manner, mainly relying on input perturbations [41, 53], gradients [58, 60] and so on [57]. In parallel, more recent works advocated intrinsically interpretable models, that are categorized as *interpretable-by-design* methods [1, 2, 56]. For neural networks, explaining the decision path is challenging due to the large set of parameters. Therefore, the more prevalent choice for intrinsic interpretable neural models is to shift the transparency to the input features. Namely, the final prediction comes from a subset selection of input elements, e.g., words or pixels and the rest irrelevant features are masked out [16, 36, 71]. Importantly, this selection decision can be learned jointly with the predictive accuracy of a model. Thereby, we limit our research focus in intrinsic interpretable ML models.

Due to the discrete nature of selection, many approaches such as L2X [16], *Concrete AutoEncoders* (CAE) [6], *Instance-wise Feature grouping* (IFG) [43] apply Gumbel-Softmax sampling [24] to enable backpropagation through the feature selection process. Alternatively, regularization is also a commonly-used feature selection approach in traditional ML algorithms [31, 62], and is applicable to neural models, i.e., with INVASE [69] or LassoNet [35]. Moreover, TabNet [4] applies both regularization and the sparsemax activation function [42] to realize sparse selection. These approaches have been successfully applied in language, vision and tabular domains, and suggested that the resulting feature selections substantially improved the user understanding of models and datasets [28, 55].

Despite their success in other domains, we find that the above-mentioned feature selection methods for neural models (L2X, CAE, IFG, INVASE, LassoNet and TabNet) have not been studied in the LTR setting. In response, we hope to bridge this gap between the interpretable ML and the LTR field by adapting and applying these methods to neural ranking models.

3 Background

3.1 Learning-to-Rank (LTR)

The LTR task can be formulated as optimizing a scoring function f that given item features x predicts an item score $f(x) \in \mathbb{R}$, so that ordering items according to their scores corresponds to the optimal ranking [38]. Generally, there are

Table 1: Properties of feature selection methods from the interpretable ML field as discussed in Section 3.

Method	Global	Local	Sampling	Regularization	Fixed-Budget	Composable
L2X [16]		✓	✓		✓	✓
INVASE [69]		✓	✓	✓		✓
CAE [6]	✓		✓		✓	✓
IFG [43]		✓	✓			✓
LASSONET [35]	✓			✓		✓
TABNET [4]		✓		✓		
G-L2X (ours)	✓		✓		✓	✓

relevance labels y available for each item, often these are labels provided by experts where $y \in \{0, 1, 2, 3, 4\}$ [14,49,50]. Given a training set $\mathcal{D}_q = \{(x_i, y_i)\}_{i=1}^{N_q}$ for a single query q, optimization is done by minimizing a LTR loss, for instance, the *softmax cross entropy loss* [9, 13]:

$$\mathcal{L}(f \mid \mathcal{D}_q) = -\frac{1}{|\mathcal{D}_q|} \sum_{(x,y)\in\mathcal{D}_q} \sum_{i=1}^{N_q} y_i \log \sigma(x_i \mid f, \mathcal{D}_q), \tag{1}$$

where σ is the softmax activation function:

$$\sigma(x \mid f, \mathcal{D}_q) = \frac{\exp(f(x))}{\sum_{x'\in\mathcal{D}_q} \exp(f(x'))}. \tag{2}$$

The resulting f is then commonly evaluated with a ranking metric, for instance, the widely-used normalized discounted cumulative gain metric (NDCG) [25].

3.2 Properties of Feature Selection Methods

As discussed before, feature selection is used in the interpretable ML field to better understand which input features ML models use to make their predictions. Furthermore, feature selection is also important to LTR for increasing the efficiency of ranking systems. However, selecting a subset of input features without compromising the model performance is an NP-hard problem, since the number of possible subsets grows exponentially with the number of available features [23, 48]. As a solution, the interpretable ML field has proposed several methods that approach feature selection as an optimization problem. We will now lay out several important properties that can be used to categorize these methods, which will be elaborated in next section.

Global vs. local. Global methods select a single subset of features for the entire dataset, whereas local methods can vary their selection over different items.

Composable vs. non-composable. Non-composable methods are designed for a specific model architecture, and therefore, they can only perform feature

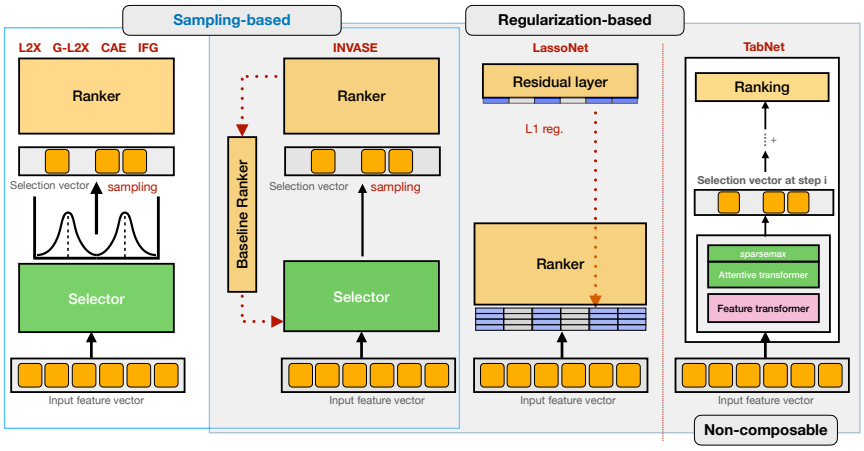

Fig. 1: Methods overview, as described in Section 4.

selection for those models. Conversely, composable methods are not constrained to specific architectures, and thus, they work for any (differentiable) model.

Fixed-budget vs. budget-agnostic. Fixed-budget methods work with a predefined selection budget, i.e., what number of features should be selected in total, or a cost per feature and a maximum total cost for the selection. Their counterparts are budget-agnostic methods that do not use an explicit budget, consequently, one has to carefully fine-tune their hyper-parameters to achieve a desired performance-sparsity trade-off.

Sampling-based vs. regularization-based. As their names imply, sampling-based methods optimize a sampling procedure to perform the feature selection, whereas regularization-based methods use an added regularization loss to stimulate sparsity in feature selection. While these groups apply very different approaches, whether one is significantly more useful for LTR purposes than the other remains unknown.

4 Feature Selection from Interpretable ML for LTR

In this section, we present a brief technical overview of our selection of six interpretable ML methods and their adaption to neural LTR models, and propose our G-L2x method based on a minor modification. While any ranking loss can be chosen, we use a listwise softmax cross entropy (Eq. 1) with all methods, for the sake of simplicity. Therefore, the training of each query is conducted after generating the output of all documents associated with the query. Table 1 highlights the properties of all methods and Figure 1 provides a visual overview to accompany this section.

4.1 Sampling-based Feature Selection

Sampling-based approaches use a two-stage architecture consisting of a *selector* that generates a sparse selection over the input features; and a *ranker* that only takes selected features as its input, in the form of a masked vector $\hat{x}$.

The training of a ranker follows conventional LTR, i.e., Eq 1 with x_i replaced by $\hat{x_i}$. But the optimization of a selector (ζ) is not as straightforward; Usually, ζ constructs a probability distribution $\mathbf{p} = [p_1, p_2, \cdots p_d]$, indicating a selection probability per feature. However, the ranker uses a concrete selection $m \in \{0,1\}^d$ from the probability distribution, and this concrete operation does not allow optimization of the selector via backpropagation. The common solution is to generate a differentiable approximation $\tilde{m}$, by *concrete relaxation* or the Gumbel-Softmax trick [24]. Namely, the selection of p_i can be approximated with the differentiable c_i as:

$$c_i = \frac{exp\{(\log p_i + g_i)/\tau\}}{\sum_{j=1}^{d} exp\{(\log p_j + g_i)/\tau\}}, \tag{3}$$

where g is the Gumbel noise and $\tau \in \mathbb{R}^{>0}$ is the temperature parameter. Now, the selector ζ can be optimized with stochastic gradient descent by using $\tilde{m}$. The following four sampling-based methods apply this overall procedure, but differ in how they generate $\mathbf{p}$ and $\tilde{m}$.

Learning to explain (L2X). L2X [16] is a local selection method since its neural selector generates a probability distribution $\mathbf{p}$ for each individual input instance. To generate $\tilde{m}$, L2X repeats the sampling procedure k times (Eq. 3), and subsequently, uses the maximum c_i out of the k repeats for the i_{th} element in $\tilde{m}$. The intention behind this maximization step is to make the top-k important features more likely to have high probability scores (ideally close to 1).

Global learning to explain (G-L2X, ours). As a counterpart, we propose a global method G-L2Xbased on L2X. Our change is straightforward, where L2X generates a different distribution $\mathbf{p}$ for each item, we apply the same $\mathbf{p}$ to all items. In other words, G-L2X includes a global selector layer ζ ($\zeta \in \mathbb{R}^d$) to simulate $\mathbf{p}$, and sampling is conducted in the same way as L2X on the selector weights. Thereby, G-L2X will select the same features for all items in the dataset.

Concrete autoencoder (CAE). CAE [6] is a global method where the selector is the encoder part of an auto-encoder model [30]. Specifically, the selector compresses the input into a smaller representation $\hat{x}$, by linearly combining selected features, i.e. $x^\top \tilde{m}$, where $\tilde{m} \in \mathbb{R}^{k \times d}$ can be viewed as approximated k-hot concrete selection, sampled from the selector weights ($\zeta \in \mathbb{R}^{k \times d}$). Therefore, CAE might result in repetitive selection, and the input dimension to the predictor is reduced to k.

Instance-wise Feature Grouping (IFG). IFG [43] applies a similar approach as L2X, but clusters features into groups and then selects k feature groups for prediction. IFG first assigns a group for each feature via Gumbel-sampling, and then makes a feature selection by Gumbel-sampling k out of the resulting

groups. This grouping decision is also guided by how rich the selected features are to recover the original input. Therefore, apart from the ranking objective, IFG jointly optimizes an additional input restoring objective as well (similar to auto-encoders [30]). IFG is agnostic to the number of selected features and the group sizes, it can produce oversized groups and very large selections.

4.2 Regularization-based Feature Selection

Instead of the budget-explicit feature selection, regularization-based methods induce sparsity through implicit constraints enforced by regularization terms in the training objective. We propose modifications to three existing methods to make them applicable to the LTR setting.

INVASE. We consider INVASE [69] to be a hybrid approach involving both sampling and regularization. Built on the same structure as L2X, the selector of INVASE generates a *boolean/hard mask* m (instead of the approximation $\tilde{m}$) via Bernoulli sampling to train the predictor. Since this disables backpropagation, INVASE uses a customized loss function that does not need the gradients from the predictor to train the selector. The idea is to apply another individual baseline predictor model that takes the full-feature input, simultaneously with the predictor that takes the masked input. The loss difference between the two predictors is used as a scale to train the selector. Meanwhile, the L1 regularization is applied to the selector output to enforce selection sparsity. Ultimately, INVASE will push the selector to output a small set of selections which leads to the most similar predictions as using all features.

LASSONET. As the name suggests, LASSONET [35] adapts a traditional Lasso (L1 regularization) [62] on the first layer of a neural model to eliminate unnecessary features. The challenge with neural models is, all weights corresponding to a particular feature entry in the layer have to be zero in order to mask out the feature. Towards this, LASSONET adds a residual layer with one weight per input feature to the original model to perform as the traditional Lasso. Then, after every optimization step, LASSONET develops a proximal optimization algorithm to adjust the weights of the first layer, so that all absolute elements of each row are smaller than the respective weight of residual layer corresponding to a specific feature. Thereby, LASSONET performs global selection and the sparsity scale is adjusted by the L1 regularization on the residual layer weights.

TABNET. Unlike the previous methods, TABNET [4] is non-composable and tied to a specific tree-style neural architecture. It imitates a step-wise selection process before it outputs the final prediction based only on the selected features. Each step has the same neural component/block but with its own parameters, thus the model complexity and selection budget grow linearly with the number of steps. At each step, the full input is transformed by a feature transformer block first, and then an attentive transformer block conducts feature selection by sparsemax activation [42], as the weights in the resulting distribution corresponding to unselected features are zeros. The final prediction is aggregated from all steps to

simulate ensemble models. A final mask m is a union of selections from all steps, and the entropy of the selection probabilities is used as the sparsity regularization.

5 Experimental Setup

Since feature selection can be applied in various manners and situations, we structure our experiments around three scenarios:

- *Scenario 1: Simultaneously train and select.* Both the ranking model and the feature selection are learned once and jointly. The methods are evaluated by the performance-sparsity trade-off. It is the standard setup for evaluating embedded feature selection methods in the interpretable ML field [43, 52].
- *Scenario 2: Train then select with an enforced budget.* Practitioners generally set hard limits to the computational costs a system may incur and the efficiency of the system can be greatly enhanced if it only requires a much smaller amount of features to reach competitive performance. Following the previous scenario, we evaluate the trained model with test instances where only a fixed amount of features (which the method deems important and selects frequently during training) are presented and the rest are masked out. The resulting ranking performance and the costs of computing the required features indicate how practical the method is in efficiency improvements.

Datasets and preprocessing. We choose three public benchmark datasets: MQ2008 (46 features) [50], Web30k (136 features) [49] and Yahoo (699 features) [14], to cover varying numbers of available features. We apply a $\log_{1p}$ transformation to the features of Web30k, as suggested in [51]. Yahoo contains cost labels for each feature, for Web30k we use cost estimates suggested by previous work [21].[3] All reported results are evaluated on the held-out test set partitions of the datasets.

Models. We use a standard feed-forward neural network with batch normalization, three fully-connected layers and tanh activation as the ranking model, denoted as DNN. According to the findings in [51], this simple model performs closely to the most effective transformer-based models, but requires much less resources to run. The selector models of L2X and INVASE have the same architecture, and as the only exception, TABNET is applied with its own unalterable model (see Section 4).

Implementation. Our experimental implementation is done in *PyTorch Lightning* [19]. For TABNET and LASSONET existing implementations were used.[4] We created our own implementations for the rest of the methods.

[3] MQ2008 is omitted from cost analysis since no associated cost information is available.

[4] `https://github.com/dreamquark-ai/tabnet`; `https://github.com/lasso-net/lassonet`

Table 2: Results of ranking performance and feature sparsity for methods applied in Scenario 1. For comparison, we also include GBDT [8, 29] and DNN baselines without feature selection as upper bound. #F denotes the number of selected features. Reported results are averaged over 5 random seeds (*std* in parentheses). Bold font indicates the highest performing selection method; the $\star$ and underlines denote scores that are *not* significantly outperformed by GBDT and the bold-score method, respectively ($p > 0.05$, paired t-tests using Bonferonni's correction).

Listwise loss	**MQ2008 NDCG@k** (%) @1	@10	#F	**Web30k NDCG@k** (%) @1	@10	#F	**Yahoo NDCG@k** (%) @1	@10	#F
Without feature selection.									
GBDT	69.3 (2.5)	80.8 (1.7)	46	50.4 (0.1)	52 (0.1)	136	72.2 (0.1)	79.2 (0.1)	699
DNN	66.2* (2)	80.2* (0.6)	46	46.1 (0.6)	47.7 (0.2)	136	69.4 (0.3)	76.9 (0.1)	699
Fixed-budget feature selection using the DNN ranking model.									
CAE	63.0 (1.1)	78.7 (0.5)	4	32.9 (2.9)	36.6 (2.2)	13	59.2 (0.2)	69.5 (0.1)	6
G-L2X	63.8* (1.3)	79.1 (0.4)	4	41.1 (0.9)	44.4 (0.3)	13	65.4 (0.1)	74.0 (0.0)	6
L2X	63.0 (2.1)	78.7 (0.7)	4	34.5 (2.4)	39.7 (1.9)	13	61.9 (1.1)	73.2 (0.3)	6
Budget-agnostic feature selection using the DNN ranking model.									
INVASE	62.5 (2.2)	77.5 (2.1)	5 (2)	15.1 (0.0)	22.1 (0.0)	0	38.7 (0.0)	57.8 (0.0)	0
IFG	**66.4*** (0.9)	**80.4*** (0.5)	20 (2)	32.5 (5.3)	37.5 (5.3)	72 (30)	69.6 (0.2)	77.1 (0.2)	58 (3)
LASSONET	64.7* (2.2)	79.3 (1.2)	6 (3)	39.4 (0.8)	42.1 (0.3)	8 (2)	63.1 (2.3)	72.4 (1.5)	12 (4)
Budget-agnostic feature selection using a method-specific ranking model.									
TABNET	64.7* (2.7)	78.2 (1.2)	7 (3)	**47.0** (0.4)	**49.2** (0.1)	8 (1)	**70.2** (0.4)	**77.7** (0.1)	6 (1)

6 Results

We report the findings in this section, aiming to answer two questions: (1) *how effective are investigated methods in the ranking setup?* and (2) *can those methods improve efficiency?* Each question corresponds to one of the scenarios described in Section 5.

Simultaneous Optimization and Selection. We begin by investigating the effectiveness of the feature selection methods when applied to Scenario 1, where feature selection and model optimization are performed simultaneously. The results for this scenario are displayed in Table 2 and Figure 2.

Table 2 shows the ranking performance and the respective feature sparsity of all feature selection methods and two baselines without any selection as the upper-bound reference. For fixed-budget methods, the budgets were set to 10% of the total number of features for MQ2008 and Web30k, and 1% for Yahoo (the results with varying budgets are displayed in Figure 2). Since the sparsity of budget-agnostic methods is more difficult to control, we performed an extensive grid search and used the hyper-parameters that produced the highest ranking performance with a comparable feature sparsity as the other methods.

The results in Table 2 show that not all feature selection methods are equally effective, and their performance can vary greatly over datasets. For instance, on MQ2008 all methods perform closely to the baselines, with only a fraction of the features. However, this is not the case for bigger datasets like Web30k and

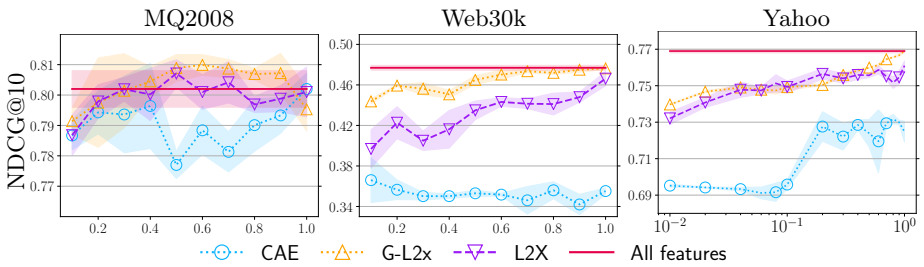

Fig. 2: Results of three fixed-budget methods applied to scenario 1. The x-axis indicates the pre-specified percentile of selected features (k). The shaded area shows the standard deviation over 5 random seeds.

Yahoo. In particular, INVASE selects no features at all due to big uncertainty in selection (for this reason, we omit INVASE from all further comparisons). On the other hand, IFG performs poorly in inducing sparsity, mainly because of its input reconstruction objective, whereas the ranking performance is not substantially better than the rest of methods. Additionally, CAE does not seem effective either, and furthermore, increasing the selected features does not always result in better ranking performance (cf. Figure 2). This is most-likely because CAE samples with replacement, and thus the same features can be selected repeatedly.

In contrast, the other two sampling-based methods L2X and G-L2X are designed to avoid the repetitive selection issue. Overall, the global selection G-L2X outperforms the local counterpart L2X, possibly because global selection generalizes better to unseen data. Another global method LASSONET is also inferior to G-L2X, mainly due to the difficulties in sparsity weight tuning and manually adjusting weights in the input layer.

Lastly, TABNET shows the best performance-sparsity balance across all datasets, and even outperforms the DNN baseline. Although, the comparison between TABNET and DNN is not completely fair, as they optimize different neural architectures. It does reveal large feature redundancies in these datasets: TABNET uses <10% of features on Web30k and 1% on Yahoo, yet still beats the DNN baseline with all features.

To summarize, we find that the local method TABNET is the most effective at balancing ranking performance and sparsity. Slightly inferior but competitive enough is the global method G-L2X, which reached > 95% of baseline performance with only 1% features on Yahoo and > 93% with 10% on Web30k.

Feature Selection for Trained Ranking Models. Next, we evaluate the methods in Scenario 2, where only a specified budget (i.e., a given number of features) of features are present in the test input. Figure 3 displays both the ranking performance and the total feature cost for varying degrees of sparsity. The costs represent the time it requires to retrieve the selected feature sets, and allow us to estimate the actual efficiency improvements they provide.

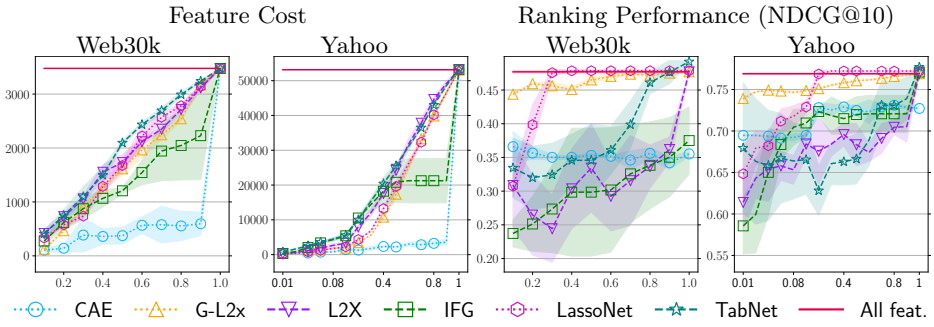

Fig. 3: Scenario 2. Feature cost (left two) and ranking performance (right two) under incomplete input. The x-axis indicates how many percentages of features are present in the input, to test the trained ranking model. Note this differs from specifying k during training for fixed-budget methods in scenario 1.

Unlike previous scenario, all local methods including TabNet, are no longer able to maintain superior performance. This is because for local methods, the selection is made conditioned on full input information, and an incomplete input could affect the selection and thus disrupt its prediction performance.

In contrast, global methods are immune to input changes. Therefore, Cae is still not performing well as it did in Scenario 1; G-L2x and LassoNet provide the best overall performance under small costs. LassoNet maintains baseline performance with less than 40% of features on both datasets, while G-L2x outperforms LassoNet when selected features are less than 30%. Meanwhile, it also shows LassoNet tends to select more costly features than G-L2x.

To conclude, we find that global methods G-L2x and LassoNet perform the best in Scenario 2, where upcoming query inputs are masked under enforced feature budgets. Particularly, G-L2x is superior in both ranking and computing cost when the feature budget is small. This translates to substantial efficiency improvements in practical terms, as ranking performance is maintained by selected features only.

7 Conclusion

The main goal of this work is to bring the interpretable ML and the LTR fields closer together. To this end, we have studied whether feature selection methods from the interpretable ML are effective for neural LTR, for both interpretability and efficiency purposes.

Inspired by the scarcity of feature selection methods for neural ranking models in previous work, we adapted six existing methods from the interpretable ML for the neural LTR setting, and also proposed our own G-L2x approach. We discussed different properties of these methods and their relevance to the LTR task. Lastly, we performed extensive experiments to evaluate the methods in terms of their

trade-offs between ranking performance and sparsity, in addition, their efficiency improvements through feature cost reductions. Our results have shown that several methods from interpretable ML are highly effective at embedded feature selection for neural LTR. In particular, the local method TabNet can reach the upper bound with less than 10 features; the global methods, in particular G-L2x, can reduce feature retrieval costs by more than 70%, while maintaining 96% of performance relative to a full feature model.

We hope our investigation can bridge the gap between the LTR and interpretable ML fields. The future work can be developing more interpretable and efficient ranking systems, and how that interpretability could support both practitioners and the users of ranking systems.

Acknowledgements This work is partially supported by German Research Foundation (DFG), under the Project IREM with grant No. AN 996/1-1, and by the Netherlands Organisation for Scientific Research (NWO) under grant No. VI.Veni.222.269.

References

1. Abdul, A.M., von der Weth, C., Kankanhalli, M.S., Lim, B.Y.: COGAM: measuring and moderating cognitive load in machine learning model explanations. In: Bernhaupt, R., Mueller, F.F., Verweij, D., Andres, J., McGrenere, J., Cockburn, A., Avellino, I., Goguey, A., Bjøn, P., Zhao, S., Samson, B.P., Kocielnik, R. (eds.) CHI '20: CHI Conference on Human Factors in Computing Systems, Honolulu, HI, USA, April 25-30, 2020. pp. 1–14. ACM (2020). https://doi.org/10.1145/3313831.3376615, https://doi.org/10.1145/3313831.3376615
2. Alvarez-Melis, D., Jaakkola, T.S.: Towards robust interpretability with self-explaining neural networks. In: Bengio, S., Wallach, H.M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada. pp. 7786–7795 (2018), https://proceedings.neurips.cc/paper/2018/hash/3e9f0fc9b2f89e043bc6233994dfcf76-Abstract.html
3. Arapakis, I., Bai, X., Cambazoglu, B.B.: Impact of response latency on user behavior in web search. In: Geva, S., Trotman, A., Bruza, P., Clarke, C.L.A., Järvelin, K. (eds.) The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014. pp. 103–112. ACM (2014). https://doi.org/10.1145/2600428.2609627, https://doi.org/10.1145/2600428.2609627
4. Arik, S.Ö., Pfister, T.: Tabnet: Attentive interpretable tabular learning. In: Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021. pp. 6679–6687. AAAI Press (2021). https://doi.org/10.1609/AAAI.V35I8.16826, https://doi.org/10.1609/aaai.v35i8.16826

5. Bai, X., Cambazoglu, B.B.: Impact of response latency on sponsored search. Inf. Process. Manag. **56**(1), 110–129 (2019). https://doi.org/10.1016/J.IPM.2018.10.005, https://doi.org/10.1016/j.ipm.2018.10.005
6. Balin, M.F., Abid, A., Zou, J.Y.: Concrete autoencoders: Differentiable feature selection and reconstruction. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA. Proceedings of Machine Learning Research, vol. 97, pp. 444–453. PMLR (2019), http://proceedings.mlr.press/v97/balin19a.html
7. Barreda-Ángeles, M., Arapakis, I., Bai, X., Cambazoglu, B.B., Pereda-Baños, A.: Unconscious physiological effects of search latency on users and their click behaviour. In: Baeza-Yates, R., Lalmas, M., Moffat, A., Ribeiro-Neto, B.A. (eds.) Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015. pp. 203–212. ACM (2015). https://doi.org/10.1145/2766462.2767719, https://doi.org/10.1145/2766462.2767719
8. Bruch, S.: An alternative cross entropy loss for learning-to-rank. In: Leskovec, J., Grobelnik, M., Najork, M., Tang, J., Zia, L. (eds.) WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021. pp. 118–126. ACM / IW3C2 (2021). https://doi.org/10.1145/3442381.3449794, https://doi.org/10.1145/3442381.3449794
9. Bruch, S., Wang, X., Bendersky, M., Najork, M.: An analysis of the softmax cross entropy loss for learning-to-rank with binary relevance. In: Fang, Y., Zhang, Y., Allan, J., Balog, K., Carterette, B., Guo, J. (eds.) Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval, ICTIR 2019, Santa Clara, CA, USA, October 2-5, 2019. pp. 75–78. ACM (2019). https://doi.org/10.1145/3341981.3344221, https://doi.org/10.1145/3341981.3344221
10. Burges, C.J.C., Ragno, R., Le, Q.V.: Learning to rank with nonsmooth cost functions. In: Schölkopf, B., Platt, J.C., Hofmann, T. (eds.) Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006. pp. 193–200. MIT Press (2006), https://proceedings.neurips.cc/paper/2006/hash/af44c4c56f385c43f2529f9b1b018f6a-Abstract.html
11. Burges, C.J.C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.N.: Learning to rank using gradient descent. In: Raedt, L.D., Wrobel, S. (eds.) Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005. ACM International Conference Proceeding Series, vol. 119, pp. 89–96. ACM (2005). https://doi.org/10.1145/1102351.1102363, https://doi.org/10.1145/1102351.1102363
12. Burges, C.J.: From ranknet to lambdarank to lambdamart: An overview. Learning **11**(23-581), 81 (2010)
13. Cao, Z., Qin, T., Liu, T., Tsai, M., Li, H.: Learning to rank: from pairwise approach to listwise approach. In: Ghahramani, Z. (ed.) Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007. ACM International Conference Proceeding Series, vol. 227, pp. 129–136. ACM (2007). https://doi.org/10.1145/1273496.1273513, https://doi.org/10.1145/1273496.1273513
14. Chapelle, O., Chang, Y.: Yahoo! learning to rank challenge overview. In: Chapelle, O., Chang, Y., Liu, T. (eds.) Proceedings of the Yahoo! Learning to Rank Challenge,

held at ICML 2010, Haifa, Israel, June 25, 2010. JMLR Proceedings, vol. 14, pp. 1–24. JMLR.org (2011), http://proceedings.mlr.press/v14/chapelle11a.html
15. Chapelle, O., Keerthi, S.S.: Efficient algorithms for ranking with svms. Inf. Retr. **13**(3), 201–215 (2010). https://doi.org/10.1007/S10791-009-9109-9, https://doi.org/10.1007/s10791-009-9109-9
16. Chen, J., Song, L., Wainwright, M.J., Jordan, M.I.: Learning to explain: An information-theoretic perspective on model interpretation. In: Dy, J.G., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018. Proceedings of Machine Learning Research, vol. 80, pp. 882–891. PMLR (2018), http://proceedings.mlr.press/v80/chen18j.html
17. Dato, D., Lucchese, C., Nardini, F.M., Orlando, S., Perego, R., Tonellotto, N., Venturini, R.: Fast ranking with additive ensembles of oblivious and non-oblivious regression trees. ACM Trans. Inf. Syst. **35**(2), 15:1–15:31 (2016). https://doi.org/10.1145/2987380, https://doi.org/10.1145/2987380
18. Du, M., Liu, N., Hu, X.: Techniques for interpretable machine learning. Commun. ACM **63**(1), 68–77 (2020). https://doi.org/10.1145/3359786, https://doi.org/10.1145/3359786
19. Falcon, W., et al.: Pytorch lightning. GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning **3**, 6 (2019)
20. Freund, Y., Iyer, R.D., Schapire, R.E., Singer, Y.: An efficient boosting algorithm for combining preferences. In: Shavlik, J.W. (ed.) Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Madison, Wisconsin, USA, July 24-27, 1998. pp. 170–178. Morgan Kaufmann (1998)
21. Gallagher, L., Chen, R., Blanco, R., Culpepper, J.S.: Joint optimization of cascade ranking models. In: Culpepper, J.S., Moffat, A., Bennett, P.N., Lerman, K. (eds.) Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019. pp. 15–23. ACM (2019). https://doi.org/10.1145/3289600.3290986, https://doi.org/10.1145/3289600.3290986
22. Geng, X., Liu, T., Qin, T., Li, H.: Feature selection for ranking. In: Kraaij, W., de Vries, A.P., Clarke, C.L.A., Fuhr, N., Kando, N. (eds.) SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, July 23-27, 2007. pp. 407–414. ACM (2007). https://doi.org/10.1145/1277741.1277811, https://doi.org/10.1145/1277741.1277811
23. Gigli, A., Lucchese, C., Nardini, F.M., Perego, R.: Fast feature selection for learning to rank. In: Carterette, B., Fang, H., Lalmas, M., Nie, J. (eds.) Proceedings of the 2016 ACM on International Conference on the Theory of Information Retrieval, ICTIR 2016, Newark, DE, USA, September 12- 6, 2016. pp. 167–170. ACM (2016). https://doi.org/10.1145/2970398.2970433, https://doi.org/10.1145/2970398.2970433
24. Jang, E., Gu, S., Poole, B.: Categorical reparametrization with gumble-softmax (2017)
25. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques. ACM Trans. Inf. Syst. **20**(4), 422–446 (2002). https://doi.org/10.1145/582415.582418, http://doi.acm.org/10.1145/582415.582418
26. Joachims, T.: Optimizing search engines using clickthrough data. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery

and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada. pp. 133–142. ACM (2002). https://doi.org/10.1145/775047.775067, https://doi.org/10.1145/775047.775067

27. Joachims, T.: Training linear svms in linear time. In: Eliassi-Rad, T., Ungar, L.H., Craven, M., Gunopulos, D. (eds.) Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006. pp. 217–226. ACM (2006). https://doi.org/10.1145/1150402.1150429, https://doi.org/10.1145/1150402.1150429
28. Kaur, H., Nori, H., Jenkins, S., Caruana, R., Wallach, H.M., Vaughan, J.W.: Interpreting interpretability: Understanding data scientists' use of interpretability tools for machine learning. In: Bernhaupt, R., Mueller, F.F., Verweij, D., Andres, J., McGrenere, J., Cockburn, A., Avellino, I., Goguey, A., Bjøn, P., Zhao, S., Samson, B.P., Kocielnik, R. (eds.) CHI '20: CHI Conference on Human Factors in Computing Systems, Honolulu, HI, USA, April 25-30, 2020. pp. 1–14. ACM (2020). https://doi.org/10.1145/3313831.3376219, https://doi.org/10.1145/3313831.3376219
29. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.: Lightgbm: A highly efficient gradient boosting decision tree. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA. pp. 3146–3154 (2017), https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html
30. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: Bengio, Y., LeCun, Y. (eds.) 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings (2014), http://arxiv.org/abs/1312.6114
31. Lai, H., Pan, Y., Liu, C., Lin, L., Wu, J.: Sparse learning-to-rank via an efficient primal-dual algorithm. IEEE Trans. Computers **62**(6), 1221–1233 (2013). https://doi.org/10.1109/TC.2012.62, https://doi.org/10.1109/TC.2012.62
32. Lai, H., Pan, Y., Liu, C., Lin, L., Wu, J.: Sparse learning-to-rank via an efficient primal-dual algorithm. IEEE Trans. Computers **62**(6), 1221–1233 (2013). https://doi.org/10.1109/TC.2012.62, https://doi.org/10.1109/TC.2012.62
33. Lai, H., Pan, Y., Tang, Y., Yu, R.: Fsmrank: Feature selection algorithm for learning to rank. IEEE Trans. Neural Networks Learn. Syst. **24**(6), 940–952 (2013). https://doi.org/10.1109/TNNLS.2013.2247628, https://doi.org/10.1109/TNNLS.2013.2247628
34. Laporte, L., Flamary, R., Canu, S., Déjean, S., Mothe, J.: Nonconvex regularizations for feature selection in ranking with sparse SVM. IEEE Trans. Neural Networks Learn. Syst. **25**(6), 1118–1130 (2014). https://doi.org/10.1109/TNNLS.2013.2286696, https://doi.org/10.1109/TNNLS.2013.2286696
35. Lemhadri, I., Ruan, F., Abraham, L., Tibshirani, R.: Lassonet: A neural network with feature sparsity. J. Mach. Learn. Res. **22**, 127:1–127:29 (2021), http://jmlr.org/papers/v22/20-848.html
36. Leonhardt, J., Rudra, K., Anand, A.: Extractive explanations for interpretable text ranking. ACM Trans. Inf. Syst. (dec 2022). https://doi.org/10.1145/3576924, https://doi.org/10.1145/3576924

37. Li, P., Burges, C.J.C., Wu, Q.: Mcrank: Learning to rank using multiple classification and gradient boosting. In: Platt, J.C., Koller, D., Singer, Y., Roweis, S.T. (eds.) Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007. pp. 897–904. Curran Associates, Inc. (2007), https://proceedings.neurips.cc/paper/2007/hash/b86e8d03fe992d1b0e19656875ee557c-Abstract.html
38. Liu, T.: Learning to rank for information retrieval. Found. Trends Inf. Retr. **3**(3), 225–331 (2009). https://doi.org/10.1561/1500000016, https://doi.org/10.1561/1500000016
39. Lucchese, C., Nardini, F.M., Orlando, S., Perego, R., Veneri, A.: ILMART: interpretable ranking with constrained lambdamart. In: SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022. pp. 2255–2259. ACM (2022). https://doi.org/10.1145/3477495.3531840, https://doi.org/10.1145/3477495.3531840
40. Lucchese, C., Nardini, F.M., Orlando, S., Perego, R., Veneri, A.: ILMART: interpretable ranking with constrained lambdamart. In: Amigó, E., Castells, P., Gonzalo, J., Carterette, B., Culpepper, J.S., Kazai, G. (eds.) SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022. pp. 2255–2259. ACM (2022). https://doi.org/10.1145/3477495.3531840, https://doi.org/10.1145/3477495.3531840
41. Lundberg, S.M., Lee, S.: A unified approach to interpreting model predictions. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA. pp. 4765–4774 (2017), https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html
42. Martins, A.F.T., Astudillo, R.F.: From softmax to sparsemax: A sparse model of attention and multi-label classification. In: Balcan, M., Weinberger, K.Q. (eds.) Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016. JMLR Workshop and Conference Proceedings, vol. 48, pp. 1614–1623. JMLR.org (2016), http://proceedings.mlr.press/v48/martins16.html
43. Masoomi, A., Wu, C., Zhao, T., Wang, Z., Castaldi, P.J., Dy, J.G.: Instance-wise feature grouping. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual (2020), https://proceedings.neurips.cc/paper/2020/hash/9b10a919ddeb07e103dc05ff523afe38-Abstract.html
44. Molnar, C.: Interpretable machine learning. Lulu. com (2020)
45. Pan, F., Converse, T., Ahn, D., Salvetti, F., Donato, G.: Feature selection for ranking using boosted trees. In: Cheung, D.W., Song, I., Chu, W.W., Hu, X., Lin, J. (eds.) Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009. pp. 2025–2028. ACM (2009). https://doi.org/10.1145/1645953.1646292, https://doi.org/10.1145/1645953.1646292
46. Pang, L., Xu, J., Ai, Q., Lan, Y., Cheng, X., Wen, J.: Setrank: Learning a permutation-invariant ranking model for information retrieval. In: Huang, J.X.,

Chang, Y., Cheng, X., Kamps, J., Murdock, V., Wen, J., Liu, Y. (eds.) Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020. pp. 499–508. ACM (2020). https://doi.org/10.1145/3397271.3401104, https://doi.org/10.1145/3397271.3401104

47. Pobrotyn, P., Bartczak, T., Synowiec, M., Bialobrzeski, R., Bojar, J.: Context-aware learning to rank with self-attention. CoRR **abs/2005.10084** (2020), https://arxiv.org/abs/2005.10084
48. Purpura, A., Buchner, K., Silvello, G., Susto, G.A.: Neural feature selection for learning to rank. In: Hiemstra, D., Moens, M., Mothe, J., Perego, R., Potthast, M., Sebastiani, F. (eds.) Advances in Information Retrieval - 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12657, pp. 342–349. Springer (2021). https://doi.org/10.1007/978-3-030-72240-1_34, https://doi.org/10.1007/978-3-030-72240-1_34
49. Qin, T., Liu, T.: Introducing LETOR 4.0 datasets. CoRR **abs/1306.2597** (2013), http://arxiv.org/abs/1306.2597
50. Qin, T., Liu, T., Xu, J., Li, H.: LETOR: A benchmark collection for research on learning to rank for information retrieval. Inf. Retr. **13**(4), 346–374 (2010). https://doi.org/10.1007/S10791-009-9123-Y, https://doi.org/10.1007/s10791-009-9123-y
51. Qin, Z., Yan, L., Zhuang, H., Tay, Y., Pasumarthi, R.K., Wang, X., Bendersky, M., Najork, M.: Are neural rankers still outperformed by gradient boosted decision trees? In: 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net (2021), https://openreview.net/forum?id=Ut1vF_q_vC
52. Rahangdale, A., Raut, S.A.: Deep neural network regularization for feature selection in learning-to-rank. IEEE Access **7**, 53988–54006 (2019). https://doi.org/10.1109/ACCESS.2019.2902640, https://doi.org/10.1109/ACCESS.2019.2902640
53. Ribeiro, M.T., Singh, S., Guestrin, C.: "why should I trust you?": Explaining the predictions of any classifier. In: Krishnapuram, B., Shah, M., Smola, A.J., Aggarwal, C.C., Shen, D., Rastogi, R. (eds.) Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016. pp. 1135–1144. ACM (2016). https://doi.org/10.1145/2939672.2939778, https://doi.org/10.1145/2939672.2939778
54. Rigutini, L., Papini, T., Maggini, M., Scarselli, F.: Sortnet: Learning to rank by a neural-based sorting algorithm. CoRR **abs/2311.01864** (2023). https://doi.org/10.48550/ARXIV.2311.01864, https://doi.org/10.48550/arXiv.2311.01864
55. Rong, Y., Leemann, T., Nguyen, T., Fiedler, L., Seidel, T., Kasneci, G., Kasneci, E.: Towards human-centered explainable AI: user studies for model explanations. CoRR **abs/2210.11584** (2022). https://doi.org/10.48550/arXiv.2210.11584, https://doi.org/10.48550/arXiv.2210.11584
56. Rudin, C.: Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nat. Mach. Intell. **1**(5), 206–215 (2019). https://doi.org/10.1038/s42256-019-0048-x, https://doi.org/10.1038/s42256-019-0048-x

57. Shrikumar, A., Greenside, P., Kundaje, A.: Learning important features through propagating activation differences. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017. Proceedings of Machine Learning Research, vol. 70, pp. 3145–3153. PMLR (2017), http://proceedings.mlr.press/v70/shrikumar17a.html
58. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. In: Bengio, Y., LeCun, Y. (eds.) 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings (2014), http://arxiv.org/abs/1312.6034
59. Sun, Z., Qin, T., Tao, Q., Wang, J.: Robust sparse rank learning for non-smooth ranking measures. In: Allan, J., Aslam, J.A., Sanderson, M., Zhai, C., Zobel, J. (eds.) Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, Boston, MA, USA, July 19-23, 2009. pp. 259–266. ACM (2009). https://doi.org/10.1145/1571941.1571987, https://doi.org/10.1145/1571941.1571987
60. Sundararajan, M., Taly, A., Yan, Q.: Axiomatic attribution for deep networks. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017. Proceedings of Machine Learning Research, vol. 70, pp. 3319–3328. PMLR (2017), http://proceedings.mlr.press/v70/sundararajan17a.html
61. Taylor, M.J., Guiver, J., Robertson, S., Minka, T.: Softrank: optimizing non-smooth rank metrics. In: Najork, M., Broder, A.Z., Chakrabarti, S. (eds.) Proceedings of the International Conference on Web Search and Web Data Mining, WSDM 2008, Palo Alto, California, USA, February 11-12, 2008. pp. 77–86. ACM (2008). https://doi.org/10.1145/1341531.1341544, https://doi.org/10.1145/1341531.1341544
62. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological) **58**(1), 267–288 (1996)
63. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA. pp. 5998–6008 (2017), https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
64. Wang, L., Lin, J., Metzler, D.: A cascade ranking model for efficient ranked retrieval. In: Ma, W., Nie, J., Baeza-Yates, R., Chua, T., Croft, W.B. (eds.) Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011. pp. 105–114. ACM (2011). https://doi.org/10.1145/2009916.2009934, https://doi.org/10.1145/2009916.2009934
65. Wu, Q., Burges, C.J.C., Svore, K.M., Gao, J.: Adapting boosting for information retrieval measures. Inf. Retr. **13**(3), 254–270 (2010). https://doi.org/10.1007/S10791-009-9112-1, https://doi.org/10.1007/s10791-009-9112-1
66. Xia, F., Liu, T., Wang, J., Zhang, W., Li, H.: Listwise approach to learning to rank: theory and algorithm. In: Cohen, W.W., McCallum, A., Roweis, S.T. (eds.) Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008. ACM International Conference Proceeding Series,

vol. 307, pp. 1192–1199. ACM (2008). https://doi.org/10.1145/1390156.1390306, https://doi.org/10.1145/1390156.1390306

67. Xu, J., Li, H.: Adarank: a boosting algorithm for information retrieval. In: Kraaij, W., de Vries, A.P., Clarke, C.L.A., Fuhr, N., Kando, N. (eds.) SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, July 23-27, 2007. pp. 391–398. ACM (2007). https://doi.org/10.1145/1277741.1277809, https://doi.org/10.1145/1277741.1277809
68. Xu, Z.E., Huang, G., Weinberger, K.Q., Zheng, A.X.: Gradient boosted feature selection. In: Macskassy, S.A., Perlich, C., Leskovec, J., Wang, W., Ghani, R. (eds.) The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014. pp. 522–531. ACM (2014). https://doi.org/10.1145/2623330.2623635, https://doi.org/10.1145/2623330.2623635
69. Yoon, J., Jordon, J., van der Schaar, M.: INVASE: instance-wise variable selection using neural networks. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net (2019), https://openreview.net/forum?id=BJg_roAcK7
70. Yue, Y., Finley, T., Radlinski, F., Joachims, T.: A support vector method for optimizing average precision. In: Kraaij, W., de Vries, A.P., Clarke, C.L.A., Fuhr, N., Kando, N. (eds.) SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, July 23-27, 2007. pp. 271–278. ACM (2007). https://doi.org/10.1145/1277741.1277790, https://doi.org/10.1145/1277741.1277790
71. Zhang, Z., Rudra, K., Anand, A.: Explain and predict, and then predict again. In: WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining, Virtual Event, Israel, March 8-12, 2021. pp. 418–426. ACM (2021). https://doi.org/10.1145/3437963.3441758, https://doi.org/10.1145/3437963.3441758
72. Zhang, Z., Setty, V., Anand, A.: Sparcassist: A model risk assessment assistant based on sparse generated counterfactuals. In: Amigó, E., Castells, P., Gonzalo, J., Carterette, B., Culpepper, J.S., Kazai, G. (eds.) SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022. pp. 3219–3223. ACM (2022). https://doi.org/10.1145/3477495.3531677, https://doi.org/10.1145/3477495.3531677
73. Zhuang, H., Wang, X., Bendersky, M., Grushetsky, A., Wu, Y., Mitrichev, P., Sterling, E., Bell, N., Ravina, W., Qian, H.: Interpretable ranking with generalized additive models. In: WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining, Virtual Event, Israel, March 8-12, 2021. pp. 499–507. ACM (2021). https://doi.org/10.1145/3437963.3441796, https://doi.org/10.1145/3437963.3441796