Pretrained deep models outperform GBDTs in Learning-To-Rank under label scarcity

Anonymous authors
Paper under double-blind review

Abstract

On tabular data, a significant body of literature has shown that current deep learning (DL) models perform at best similarly to Gradient Boosted Decision Trees (GBDTs), while significantly underperforming them on outlier data Gorishniy et al. (2021); Rubachev et al. (2022); McElfresh et al. (2023). However, these works often study problem settings which may not fully capture the complexities of real-world scenarios. We identify a natural tabular data setting where DL models can outperform GBDTs: tabular Learning-to-Rank (LTR) under label scarcity. Tabular LTR applications, including search and recommendation, often have an abundance of unlabeled data, and *scarce* labeled data. We show that DL rankers can utilize unsupervised pretraining to exploit this unlabeled data. In extensive experiments over both public and proprietary datasets, we show that pretrained DL rankers consistently outperform GBDT rankers on ranking metrics—sometimes by as much as 38%—both overall and on outliers.

1 Introduction

The learning-to-rank (LTR) problem aims to train a model to rank a set of items according to their relevance or user preference (Liu, 2009). An LTR model is typically trained on a dataset of queries and associated *query groups* (i.e., a set of potentially relevant *documents* or *items* per query), as well as an associated (generally incomplete) ground truth ranking of the items in the query group. The model is trained to output the optimal ranking of documents or items in a query group, given a query. LTR is a core ML technology in many real world applications—most notably in search contexts including Bing web search (Qin & Liu, 2013), Amazon product search (Yang et al., 2022), and Netflix movie recommendations (Lamkhede & Kofler, 2021). More recently, LTR has also found use in Retrieval Augmented Generation (RAG), where a retrieval system is paired with an LLM (large language model) to ground the latter's responses to truth and reduce hallucination Lewis et al. (2020); Glass et al. (2022); Pan et al. (2022).

Traditionally, LTR models took *tabular features*—numerical or categorical features—of queries and documents as input features (Burges et al., 2005; Cao et al., 2007; Chapelle & Chang, 2011; Qin & Liu, 2013; Lucchese et al., 2016). As this continues to be the case for many important applications, *tabular LTR* is a core problem in machine learning (Bower et al., 2021; Yang et al., 2022; Yu, 2020; Kveton et al., 2022; Pan et al., 2022; Xia et al., 2008; Chen et al., 2009). Today, deep models are largely outperformed by gradient boosted decision trees (GBDTs) (Friedman, 2001) in ML problems with tabular features (Jeffares et al., 2023; Qin et al., 2021). In contrast, deep models are state-of-the-art by a significant margin in domains like text (Devlin et al., 2018) and images (He et al., 2016).

Recent breakthroughs in modeling non-tabular data like text and images have been driven by first training a deep neural network to learn from unlabeled data (unsupervised pretraining, or pretraining) (Devlin et al., 2018; Chen et al., 2020), followed by supervised training (finetuning). Models that are pretrained in this way can perform significantly better than models that were only trained on existing labeled data. This remarkable success stems from three factors: (1) limited access to labeled data, (2) large, available sources of unlabeled text and image data, and (3) pretraining methods that can exploit unlabeled data.

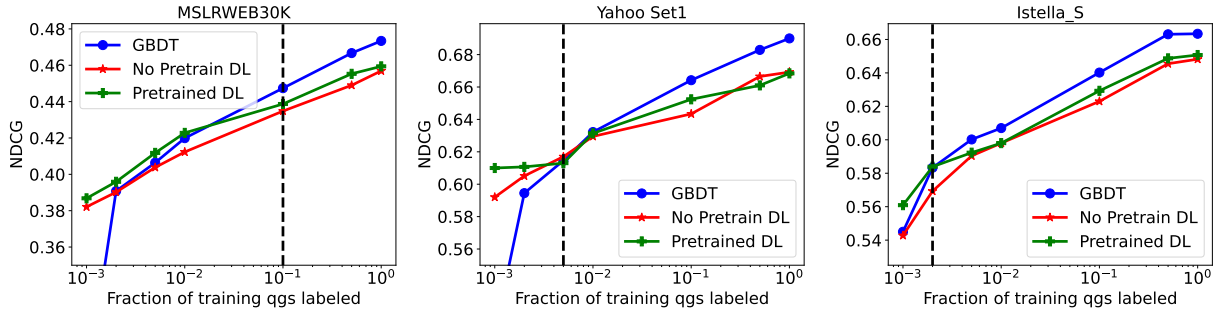

Figure 1: **Main result (Simulated crowdsourcing of labels):** We compare NDCG (↑) as we sweep the percentage of labeled training query groups (QGs); labeled QGs have relevance scores for every element of the QG. For small enough fractions of labeled query groups, pre-trained DL rankers (sometimes significantly) outperform both GBDTs (represents the best of supervised GBDTs and semi-supervised GBDTs) and non-pretrained DL rankers (represents the best of tabular ResNet (Gorishniy et al., 2021), DeepFM (Guo et al., 2017), and DCNv2 (Wang et al., 2021)). To the left of the black dotted line, pretrained rankers perform the best (statistically significant at $p = 0.05$ level). Points are averages over 3 trials.

A natural question is whether deep models can similarly use unsupervised pretraining to outperform GBDTs on the tabular LTR problem. To this end, many papers have studied unsupervised pretraining techniques—both using deep learning and GBDTs—for tabular data Yoon et al. (2020); Ucar et al. (2021); Bahri et al. (2021); Verma et al. (2021); Majmundar et al. (2022); Duh & Kirchhoff (2008); Pseudo-Label (2013). However, none of these methods have shown convincingly that deep learning (with or without unsupervised pretraining) consistently outperforms GBDTs (semi-supervised or supervised); at best, deep methods appear to achieve roughly the same performance as GBDTs (Jeffares et al., 2023; Rubachev et al., 2022; Gorishniy et al., 2021; Shwartz-Ziv & Armon, 2022; Grinsztajn et al., 2022). Furthermore, recent work showed that GBDTs are significantly more robust against outliers in the dataset than deep learning models McElfresh et al. (2023). We note that most of these works study idealized problem settings which may fail to capture complexities of real-world scenarios such as label scarcity and noisy data. Prior work in other domains show that pretrained deep models can outperform purely supervised models Chen et al. (2020) and former models are more robust to noise Hendrycks et al. (2019). Therefore, we hypothesize that pretrained deep models can outperform GBDTs in tabular settings that more accurately reflect real-world challenges.

Through extensive empirical evaluation, we confirm this hypothesis and show that deep models which use unsupervised pretraining consistently outperform GBDTs—sometimes by a large margin—**when labels are scarce**. By scarce, we mean that only a small number of query groups contain labels. This setting is more standard in most real-world deployments of LTR, rather than the idealized settings studied in prior work. For instance, scarcity can arise when organizations: (a) crowdsource labels for a limited number of query groups due to cost constraints, and/or (b) use rare user feedback (e.g., clicks, purchases) as implicit labels. Our experiments consider both sources of scarcity. They are run on the three standard public datasets in the ranking literature: MSLRWEB30K (Qin & Liu, 2013), Yahoo (Chapelle & Chang, 2011), Istella_S (Lucchese et al., 2016), as well as an industry-scale proprietary ranking dataset.

Contributions: *(1) We demonstrate for the first time that unsupervised pretraining can produce deep models that outperform GBDTs in ranking.* Specifically, pretrained deep rankers can achieve up to 38% higher NDCG (normalized discounted cumulative gain) score (Burges, 2010) than GBDTs when labels are scarce (Figure 1). For instance, Figure 1 shows on three datasets that as the labels get scarcer, GBDTs start to substantially underperform pretrained deep rankers (experimental setup in Section 4.1.2).

(2) We prescribe empirically-justified LTR-specific pretraining strategies, including a new ranking-specific pretraining loss, SimCLR-Rank, which is a modification of the widely used SimCLR (Chen et al., 2020) loss. In experiments, SimCLR-Rank achieves similar or better performance than most existing pretraining methods at an order-of-magnitude speedup. This allows SimCLR-Rank to scale to large real-world datasets such as our proprietary industry-scale ranking dataset.

(3) We demonstrate that when labels are scarce, pretrained deep rankers can perform significantly better than GBDTs on ***outlier*** *query groups, boosting NDCG by as much as 80% (Figure 8*, Appendix A.3.4). This is perhaps surprising, given prior results in the label-rich regime showing that GBDTs are significantly more robust to data with irregularities than deep rankers (McElfresh et al., 2023).

The main novelty of this work lies in the identification of label-scarce LTR as a practical real-world tabular learning problem and the results which challenge the prior understanding of the optimal modeling choices for tabular learning. We demonstrate that pretraining strongly help deep models to consistently outperform GBDTs in this label-scarce problem. Of note, these improvements over GBDTs happen even when the labeled data is plentiful in absolute numbers; up to 25% labeled query groups in public datasets and 3 million+ labeled query groups in the large industry-scale ranking dataset. Prior to our work, the ML and ranking literature had not yet identified a widely applicable setting where (pretrained) deep models could consistently outperform GBDTs (Qin et al., 2021; McElfresh et al., 2023). Due to the prevalence of label scarcity in practical settings, our results suggest a positive change in how practitioners may approach real-world tabular learning problems, especially in search and recommendation systems.

Related work: We focus on the traditional LTR setting where the features are all numeric (tabular data). In this setting, gradient boosted decision trees (GBDTs) (Friedman, 2001) have historically been the de-facto models, though there is currently great interest in applying deep neural networks to tabular data. Borisov et al. (2022) categorize existing techniques for using deep neural networks over tabular data into four types: (1) **Encoding-based methods** such as VIME (Yoon et al., 2020), SCARF (Bahri et al., 2021), IGTD (Zhu et al., 2021), and SuperTML (Sun et al., 2019); (2) **Novel hybrid architectures** such as DeepFM (Guo et al., 2017), xDeepFM (Lian et al., 2018), and many others Cheng et al. (2016); Frosst & Hinton (2017); Ke et al. (2018; 2019); Popov et al. (2019); Luo et al. (2020); Liu et al. (2020); Ivanov & Prokhorenkova (2021); Luo et al. (2021); (3) **Transformer-based architectures** including SAINT (Somepalli et al., 2021), TabNet (Arik & Pfister, 2021), TabTransformer (Huang et al., 2020), and ARM-Net Cai et al. (2021)); (4) **Regularized DNNs** (Shavitt & Segal, 2018; Kadra et al., 2021).

Within these categories, proposed architectures include factorization machines (Rendle, 2010), wide & deep architectures (Cheng et al., 2016), and similar architectures designed for recommendation systems (Guo et al., 2017; Wang et al., 2021; Lian et al., 2018; Naumov et al., 2019; Qu et al., 2016). While our paper is architecture agnostic, we perform comparisons of pretrained models against non-pretrained DeepFM and DCNv2 models to show that pretraining is needed to outperform GBDTs consistently. Self-supervised learning (SSL) or unsupervised pretraining has improved overall performance and robustness to noise (Hendrycks et al., 2019) in settings where there is a significant source of unlabeled data like text (Devlin et al., 2018; Howard & Ruder, 2018; Kitaev et al., 2018; Liu et al., 2017; Song & Roth, 2015) and images (Chen et al., 2020; Chen & He, 2021; Grill et al., 2020; Ge et al., 2021; Xie et al., 2020).

Inspired by the success of SSL in images and text, many unsupervised pretraining tasks have been proposed for tabular data. We describe some of the most well-known methods. SubTab (Ucar et al., 2021) proposes to train an autoencoder to reconstruct a table using a subset of the columns. VIME-self (Yoon et al., 2020) is a similar method. SCARF and DACL+ (Bahri et al., 2021; Verma et al., 2021) propose augmentations for the SimCLR pretraining task (Chen et al., 2020) that work for the tabular setting. RegCLR (Wang et al., 2022a) is a method for extracting tables from images, which is not directly applicable to our setting.

A smaller class of approaches has studied regularized DNNs, finding that they require limited changes to perform well on tabular data Shavitt & Segal (2018); Kadra et al. (2021). However, the most competitive of these methods require extensive hyperparameter optimization over a 14-dimensional grid Kadra et al. (2021).

While the body of work studying neural network architectures for tabular data is extensive, deep models have yet to outperform GBDTs convincingly (Qin et al., 2021; Joachims, 2006; Ai et al., 2019; Bruch et al., 2019; Ai et al., 2018; Pang et al., 2020; McElfresh et al., 2023). In this work, we consider encoding-based methods, which alter the representations over which a deep model can learn to rank; we do not focus on introducing new algorithms in this work, but rather evaluate mostly existing methods applied to tabular data. We comprehensively evaluate seven representative DL methods: SubTab, SCARF, DACL+, VIME-self, SimCLR-Rank, SAINT, SimSiam (Chen & He, 2021), and self-supervised approaches for GBDTs in Section 3.1.

Our results show that multiple pretrained deep rankers can consistently outperform GBDTs in the label-scarce tabular LTR setting, with SimCLR-Rank and SimSiam (Chen & He, 2021) generally performing the best. We provide a detailed supplemental discussion of related works in the Appendix A.1.

2 Learning-To-Rank problem and its metrics

In the LTR problem, samples are query groups (QGs) consisting of a query (e.g. shopping query) and L potentially relevant items (e.g. products) for this query. Although an LTR dataset contains multiple QGs, for the sake of simplicity, the notation in this section only handles a single QG. The i-th relevant item in this QG is represented by a feature vector $\boldsymbol{x}_i \in \mathbb{R}^d$, which captures information about this query-item pair, subsuming any query related features. If and only if the QG is labeled, such as the ones used in supervised training or testing, we also have an associated scalar relevance label y_i, which could be a binary, ordinal, or real-valued measurement of the relevance of this item for this query (Qin et al., 2021). The objective is to learn a function that ranks these L items such that the items with highest true relevance are ranked at the top.

Most LTR algorithms formulate the problem as learning a scoring function $f_\theta : \mathbb{R}^d \to \mathbb{R}$ that maps the feature vector associated with each item to a score, and then ranking the items by sorting the scores in descending order. To measure the quality of a ranking induced by our scoring function f_θ on a labeled test QG, a commonly-used (per-QG) metric is NDCG (normalized cummulative discounted gain) (↑):

$$\mathrm{NDCG}(\pi_{f_\theta}, \{y_i\}_{i=1}^L) = \frac{\mathrm{DCG}(\pi_{f_\theta}, \{y_i\}_{i=1}^L)}{\mathrm{DCG}(\pi^*, \{y_i\}_{i=1}^L)} \in [0, 1]\,, \tag{1}$$

where $\pi_{f_\theta} : [L] \to [L]$ ($[L] \triangleq \{1, \ldots, L\}$) is a ranking of the L elements induced by the scoring function f_θ on $\{\boldsymbol{x}_i\}_{i=1}^L$ (i.e. $\pi(i)$ is the rank of the i-th item), while π^* is the ideal ranking induced by the true relevance labels $\{y_i\}_{i=1}^L$, and discounted cumulative gain (DCG) is defined as $\mathrm{DCG}(\pi, \{y_i\}_{i=1}^L) = \sum_{i=1}^L \frac{2^{y_i}-1}{\log_2(1+\pi(i))}$. Here, (↑) indicates that larger metric values are preferred. Note that for summarizing this metric over the full test set, we simply take the average across all the QGs in the set. Typically, a truncated version of NDCG is used that only considers the top-k ranked items, denoted as NDCG@k. In the rest of our paper, we will refer to NDCG@5 as NDCG; this will be our evaluation metric.

We consider a natural label-scarce LTR setting with large amounts of unlabeled data and a limited supply of labels for a subset of this data. The labels that do exist can arise from multiple sources, such as crowdsourced workers paid to label QGs, or through implicit user feedback (e.g., clicks or purchases); we model both in our evaluation. Unless specified otherwise, we use all the query groups for unsupervised pretraining, but we use only the labeled query groups for supervised training.

2.1 Outlier-NDCG for outlier performance evaluation

Motivated by a recent observation that GBDTs outperform DL models in datasets with irregularities McElfresh et al. (2023), we also study their performance on outliers. In interactive ML systems like search, performing well on outlier queries is particularly valuable as it empowers users to search for more outlier queries, which in turn allows the modeler to collect more data and improve the model. To this end, we evaluate the outlier performance by computing the average NDCG on outlier queries groups and refer to it as *Outlier-NDCG* for brevity. In practice, some outlier queries may already be known, and the modeler can define the outlier datasets accordingly, like in our proprietary dataset. For example, since industry data pipelines often have missing data/features one could identify samples with missing features as outliers. When outliers are unknown, detecting them (especially in higher-dimensional data) is challenging. Hence, we create a heuristic algorithm by assuming outliers are rare values that are separated from the bulk of the data. Using it we systematically selected outlier query groups for the public datasets that we use. More details about the outliers are provided in Appendix A.2.2. Separately measuring the performance on the outliers is justified by our later observation that performance on the full dataset and the outliers can significantly differ (Tables 2 and 3).

3 Design: Unsupervised pretraining for LTR

Qualitatively, the ranking relevance of tabular records to a query is typically determined by a combination of fields. For example, if a person searches for a product (say a pair of sandals), we can think of the fields in the LTR problem as representing properties of the results, such as material, color, and price. Qualitatively, we expect that if a pair of records are the same except for small differences between fields (e.g., missing a field), then the two products should generally have similar rankings in the LTR problem. In this section, we discuss how existing methods fail to capture this qualitative intuition.

Semi-supervised learning for GBDTs: While to the best of our knowledge there are no methods to perform unsupervised pretraining in GBDTs, one can use semi-supervised methods like consistency regularization (Jeong & Shin, 2020; Sajjadi et al., 2016; Miyato et al., 2018; Oliver et al., 2018; Berthelot et al., 2019), pseudo-labeling/self-training (Rizve et al., 2021; Pseudo-Label, 2013; Shi et al., 2018; Yarowsky, 1995; McClosky et al., 2006), and PCA (Duh & Kirchhoff, 2008) to address limited size of labeled data. In consistency regularization, one increases the size of the training set by using data augmentations. In pseudo-labeling/self-training, a model trained on the available labeled data is used to label unlabeled data. Then, the final model is trained on the resulting "labeled" dataset. In PCA, one projects the features along directions of maximum variability in the union of labeled and unlabeled datasets. Since prior work has found that consistency regularization decreases the performance of GBDTs (Rubachev et al., 2022), we evaluate only pseudo-labeling and PCA. None of these methods explicitly ensures that semantically similar records have similar representations in the learned models.

Pretraining methods for tabular data: To address this gap, there have been many unsupervised pretraining methods proposed for tabular data, which can be applied to the tabular LTR setting. In this paper, we will evaluate (1) SCARF (Bahri et al., 2021) and (2) DACL+ (Verma et al., 2021) which are contrastive learning methods (Chen et al., 2020), (3) VIME-self (Yoon et al., 2020) and (4) SubTab (Ucar et al., 2021) which are autoencoder based methods. The four most commonly-evaluated baselines in the pretraining literature for tabular data are Ucar et al. (2021); Hajiramezanali et al. (2022); Levin et al. (2022); Majmundar et al. (2022). Our results suggest that many of these methods outperform GBDTs in the label-scarce regime (Section 3.1), in accordance with the main theme of this paper. However, the consistently best-performing methods in our experiments were more basic, domain-agnostic pretraining methods, which can be viewed as more primitive building blocks of some of the above methods. Surprisingly, they use *less* domain knowledge of tabular data than these more sophisticated methods.

Specifically, in the image setting, SimCLR (Chen et al., 2020) and SimSiam (Chen & He, 2021) are two widely-adopted unsupervised pretraining methods. They use the idea that two variations of the same data point should have similar representations. This idea generalizes across domains, including tabular LTR.

SimCLR (Chen et al., 2020): In a batch of B query-groups, the feature vector $\boldsymbol{x}_{q,i}$ of q-th query group and its i-th item is stochastically augmented twice to obtain $\boldsymbol{x}_{q,i}^{(0)}$ and $\boldsymbol{x}_{q,i}^{(1)}$, which are called a *positive pair*. In this paper, we use combinations of the following two augmentation strategies for all domain-agnostic pretraining methods: *(a) Zeroing:* Zero out features selected randomly and i.i.d. with fixed probability. *(b) Gaussian noise:* Add Gaussian noise of a fixed scale to all features. Further details are given in Appendix A.2.3.

Next, a base encoder $h(\cdot)$ and projection head $g(\cdot)$ map $\boldsymbol{x}_{q,i}^{(a)}$ to $\boldsymbol{z}_{q,i}^{(a)} = g(h(\boldsymbol{x}_{q,i}^{(a)}))$ for $a = 0, 1$. Then by minimizing the InfoNCE loss (Oord et al., 2018) in this projected representation space, we push the positive pair, $(\boldsymbol{z}_{q,i}^{(0)}, \boldsymbol{z}_{q,i}^{(1)})$, closer while pushing them farther from all other augmented data points in the batch, which are called their *negatives*. Formally, for $\boldsymbol{x}_{q,i}$, we minimize $\ell_{q,i}^{(0)} + \ell_{q,i}^{(1)}$, where:

$$\ell_{q,i}^{(a)} = -\cos(\boldsymbol{z}_{q,i}^{(a)}, \boldsymbol{z}_{q,i}^{(\bar{a})})/\tau + \log \sum_{q'=1}^{B} \sum_{i'=1}^{L_{q'}} \sum_{a'=0}^{1} \left[\mathbf{1}\{(q', i', a') \neq (q, i, a)\} \cdot \exp(\cos(\boldsymbol{z}_{q,i}^{(a)}, \boldsymbol{z}_{q',i'}^{(a')})/\tau)\right] \tag{2}$$

$\bar{a} = 1 - a$, τ is the temperature parameter, $L_{q'}$ is the number of items in q'-th query group, and $\cos(\boldsymbol{z}, \widetilde{\boldsymbol{z}}) = \langle \boldsymbol{z}, \widetilde{\boldsymbol{z}} \rangle / \|\boldsymbol{z}\| \|\widetilde{\boldsymbol{z}}\|$ denotes the cosine similarity. After pretraining, the encoder h is used in downstream applications. SimCLR achieves superior performance in many domains (Chen & He, 2021; Wang et al., 2022b;a). It is known that contrasting a pair of data samples which are hard to distinguish from each other,

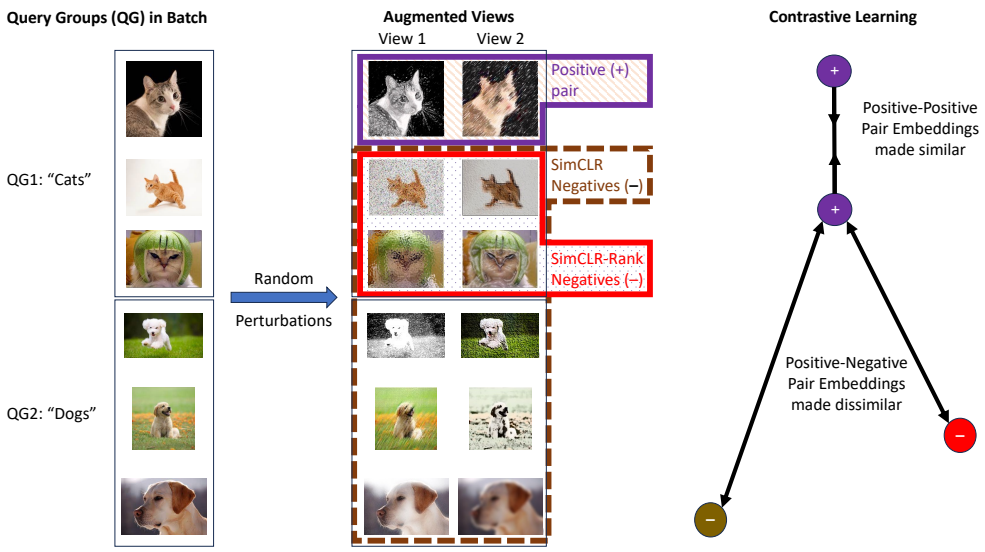

Figure 2: In SimCLR and its variants, *positive pairs*, or augmented versions of the same data sample, are trained to have similar embeddings. Positive-negative pairs, or augmented samples originating from two different data points or classes, are trained to have distant embeddings. In vanilla SimCLR, each positive pair is contrasted with *all other items in the batch*, denoted by the data points contained in the brown dashed line above. In SimCLR-Rank, each positive pair is contrasted with *only the items in the same Query Group (QG)*, denoted by the data points inside the red solid line above.

called *hard negatives* can help an encoder learn good representations (Robinson et al., 2020; Oh Song et al., 2016; Schroff et al., 2015; Harwood et al., 2017; Wu et al., 2017; Ge, 2018; Suh et al., 2019). SimCLR simply contrasts all pairs of data samples in the batch (including from other query groups) with the assumption that a large enough batch is likely to contain a hard negative.

SimSiam (Chen & He, 2021): Similar to SimCLR, for each data point $\boldsymbol{x}_{q,i}$, this method produces stochastically-augmented *positive pairs* and their *projected* representations. However, we pass these representations further through a predictor $\text{pred}(\cdot)$, to get $\boldsymbol{p}_{q,i}^{(a)} = \text{pred}(\boldsymbol{z}_{q,i}^{(a)})$ for $a = 0, 1$. Finally, we maximize the similarity between the projected and predicted representations: $\cos(\boldsymbol{p}_{q,i}^{(0)}, \text{sg}(\boldsymbol{z}_{q,i}^{(1)})) + \cos(\boldsymbol{p}_{q,i}^{(1)}, \text{sg}(\boldsymbol{z}_{q,i}^{(0)}))$, where *sg* is a stop-gradient. Unlike SimCLR, there are no *negatives*, i.e., the loss function does not push the representation of an augmented sample away from that of other samples' augmentations. The asymmetry in the loss due to the stop gradient and the predictor prevents representation collapse to make negatives unnecessary Tian et al. (2021); Zhang et al. (2021); Wang et al. (2022b). If the number of items in each query group is L, then the time/space complexity for SimSiam is only $O(BL)$ per batch, whereas it is $O(B^2L^2)$ for SimCLR. Therefore, SimSiam is more efficient and it can scale to larger batchsizes and data.

Pretraining method for LTR: Motivated by (a) SimCLR's high complexity and (b) the efficacy of contrasting with hard negatives, we additionally propose SimCLR-Rank, an LTR-specific alternative. Recall that the SimCLR loss for an item in equation 2 uses all the other items in the batch as negatives. Good pretraining strategies usually exploit the structure of the data, e.g. text sequences motivates masked token prediction Devlin et al. (2018). Studying the query group structure of LTR data, we notice that high-quality hard negatives are freely available as *the other items from the same query group*. These items are retrieved and deemed potentially similarly relevant for this query by an upstream retrieval model, making these items harder negatives than other items in the batch. So, we propose SimCLR-Rank, which modifies SimCLR to contrast only with the other items in the same query group, as illustrated in Figure 2. Formally, we modify the per-augmentation loss $\ell_{q,i}^{(a)}$ as:

$$\ell_{q,i}^{(a)} = -\cos(\boldsymbol{z}_{q,i}^{(a)}, \boldsymbol{z}_{q,i}^{(\bar{a})})/\tau + \log \sum_{i'=1}^{L_q} \sum_{a'=0}^{1} \left[\mathbf{1}\{(i', a') \neq (i, a)\} \cdot \exp(\cos(\boldsymbol{z}_{q,i}^{(a)}, \boldsymbol{z}_{q,i'}^{(a')})/\tau)\right] \tag{3}$$

Table 1: **Pretraining methods:** A comparison across unsupervised pretraining methods on NDCG, averaged over 3 trials. Here only 0.1% of the query groups have labels for finetuning. **Bold** numbers denote the best in a column, and underlined numbers are within the margin of error of the best. Given that SCARF/DACL+/SimCLR are too slow to scale to large LTR datasets (for example on the industrial-scale proprietary online shopping dataset), we find that SimCLR-Rank and SimSiam are the best pretraining methods for tabular LTR.

Method	**MSLR** (↑)	**Yahoo Set1** (↑)	**Istella** (↑)
Supervised GBDT	0.2801 ± 0.0002	0.5083 ± 0.0197	0.5450 ± 0.0000
Semi-supervised GBDT	0.2839 ± 0.0004	0.5061 ± 0.0267	0.4656 ± 0.0310
SimCLR-Rank + GBDT	0.3165 ± 0.0050	0.5504 ± 0.0135	0.5397 ± 0.0098
SimSiam + GBDT	0.3158 ± 0.0030	0.5620 ± 0.0167	0.5297 ± 0.0051
SCARF (Bahri et al., 2021)	0.3807 ± 0.0016	0.5884 ± 0.0129	0.5542 ± 0.0024
DACL+ (Verma et al., 2021)	0.3833 ± 0.0037	0.5887 ± 0.0012	**0.5626 ± 0.0024**
VIME-self (Yoon et al., 2020)	0.3834 ± 0.0011	0.5839 ± 0.0058	0.5514 ± 0.0025
SubTab (Ucar et al., 2021)	0.3748 ± 0.0025	0.5814 ± 0.0062	0.5082 ± 0.0050
SAINT (Somepalli et al., 2021)	0.3355 ± 0.0043	0.5890 ± 0.0075	0.5560 ± 0.0066
SimCLR (Chen et al., 2020)	0.3827 ± 0.0027	0.5837 ± 0.0093	0.5602 ± 0.0055
SimCLR-Sample	0.3498 ± 0.0541	0.5803 ± 0.0070	0.5479 ± 0.0016
SimCLR-Rank	**0.3868 ± 0.0026**	0.5843 ± 0.0062	0.5609 ± 0.0040
SimSiam (Chen & He, 2021)	0.3790 ± 0.0028	**0.6100 ± 0.0072**	0.5189 ± 0.0096

Empirically, SimCLR-Rank better separates embeddings within query groups relative to sampling negatives uniformly from each batch (Appendix A.5). The SimCLR-Rank loss has a time/space complexity of $O(BL^2)$, making it faster than SimCLR's $O(B^2L^2)$ complexity, and enjoying SimSiam's linear complexity in B. Next, we empirically study these methods.

3.1 Evaluation of unsupervised pretraining techniques

In this subsection, we compare the discussed pretraining strategies in the following tabular LTR setting.

Dataset. We use MSLRWEB30K (or MSLR for brevity) (Qin & Liu, 2013), Yahoo Set1 (or Set1 for brevity) (Chapelle & Chang, 2011), and Istella_S (Lucchese et al., 2016) (a smaller version of Istella (Dato et al., 2016)). In the tabular LTR literature, these three datasets are generally considered the standard; most recent papers evaluate (only) on these datasets (Qin et al., 2021; Ai et al., 2019; Pang et al., 2020; Yang et al., 2022; Yu, 2020). To simulate scarcity of labels, we let 0.1% of the training query groups in each dataset keep their labels, while the rest of the query groups have no labels. Dataset statistics are provided in Appendix A.2.1.

Methodology. For the neural pretraining approaches (which include SimCLR-Rank, SimSiam, SimCLR, SubTab, VIME-self, DACL+, SAINT, and SCARF), the base of the model is the tabular ResNet[1] (check the footnote for our explanation on the architecture choice). We also evaluate "SimCLR-Sample", a neural pretraining baseline we create to demonstrate the value of negatives from the same query group. SimCLR-Sample is like SimCLR-Rank, but at the beginning of each epoch the items are first randomly permuted and grouped into fake query groups. Then we apply SimCLR-Rank. Thus SimCLR-Sample has the same time/space complexity as SimCLR-Rank. We pretrain on all the available query groups and finetune/train on only the labeled query groups.

[1]We emphasize that we do not prescribe any specific deep learning architecture choice; the choice for the best DL model may depend on the specific LTR task. Our proposed workflow of pretraining deep models in LTR is agnostic to the specific choice of DL architecture. Prior work has proposed DL architectures designed for click-through rate (CTR)/LTR based on factorization machines (Rendle, 2010), such as DeepFM and DCNv2 (Guo et al., 2017; Wang et al., 2021), which we compare to in Section 4.1.2 and Section 4.1.2 against our pretrained models. Note that it is unclear currently how to do unsupervised pretraining using these CTR/LTR specific architectures at the moment.

For the GBDT-based approaches, we use the implementation in `lightgbm` Ke et al. (2017) and adopt a tuning strategy similar to Qin et al. (2021). We report "Supervised GBDT" as GBDT utilizing only the labeled data, and in "Semi-supervised GBDT" we report the best among pseudo-labeled (Pseudo-Label, 2013) and PCA-enriched (Duh & Kirchhoff, 2008) GBDTs as well as the combination of the two techniques. We also evaluate GBDTs enriched with SimCLR-Rank and SimSiam embeddings as additional features, which we call "SimCLR-Rank + GBDT" and "SimSiam + GBDT". We describe our experimental setup in detail in Appendix A.2.3. This includes hyperparameters, procedures for data augmentation, GBDT pseudo-labeling/PCA details, model training, and finetuning. For comparison to prior baselines on tabular data, our experimental setup is chosen to mimic the original setup in them.

Results. Table 1 charts the test NDCG for each of the pretraining methods we study. We highlight a few observations: (1) The pretraining-based DL methods almost all outperform GBDTs (both supervised and semi-supervised) in this label-scarce setting. Surprisingly, the semi-supervised GBDT baselines tend to achieve lower NDCG than the supervised GBDT baseline without semi-supervised learning. (2) When GBDTs are allowed to use representations learned from our pretraining methods, their performance consistently improves, though they do not outperform pretraining with DL. (3) SimCLR-Rank and SimSiam perform the best or the second best among these methods for most of the datasets. In particular, SimCLR-Rank outperforms SimCLR-Sample, demonstrating the value and efficiency of negatives from the same query group. As expected, SimCLR-Rank is also 7-14x faster than SimCLR/SCARF/DACL+ (Table 6 in Appendix A.3.1). Given these advantages in terms of test NDCG and speed, we select SimCLR-Rank and SimSiam as the best overall pretraining methods for tabular LTR, and use them for our continued evaluation.

4 Empirical evaluation

Next we compare the best DL pretraining methods for tabular LTR (SimCLR-Rank and SimSiam) against GBDT and non-pretrained DL models on (1) the public datasets under different label scarcity patterns to simulate real-world scenarios[2], and (2) a private large-scale ranking dataset which is naturally label-scarce.

4.1 Public datasets results

For the public datasets (MSLRWEB30K, Yahoo Set1, Istella_S; details given in Section 3.1), we simulate label scarcity occuring in two scenarios: (1) *crowdsourcing* of labels for a limited number of query groups, and (2) using scarce *implicit binary user feedback* (e.g. clicks) as labels.

4.1.1 Simulated crowdsourcing of labels

We simulate a setting where labels are crowdsourced for only some query groups due to limited resources.

Dataset. For each of the three public datasets, we vary the fraction of labeled query groups in the training set in $\{0.001, 0.002, 0.005, 0.1, 0.5, 1.0\}$. Note that within each labeled query group all items are labeled.

Methodology. We repeat the methodology from Section 3.1 for the GBDT and the SimCLR-Rank and SimSiam pretraining methods. For each dataset and labeling fraction, we report the test metric of the best GBDT ranker amongst semi-supervised GBDTs and supervised GBDTs under "GBDT". Similarly, under "no pretrain DL" we report the best supervised DL model amongst tabular ResNet (Gorishniy et al., 2021), DeepFM (Guo et al., 2017), and DCNv2 (Wang et al., 2021) under the name "no pretrain DL". Finally, we report the best pretrained ranker amongst SimCLR-Rank and SimSiam under the name "pretrained DL". Pretrained DL rankers use tabular ResNet (see Appendix A.2.3 for more details). During hyperparameter tuning, we tune for NDCG and report the resulting test NDCG and test Outlier-NDCG.

Results. We compare pretrained rankers, non-pretrained DL rankers, and GBDTs on public datasets in Figure 1. The results for Outlier-NDCG are provided in Figure 8 in Appendix A.3.4. We find that pretrained rankers outperform non-pretrained methods (including GBDTs) on NDCG and Outlier-NDCG across all public datasets, up to a dataset-dependent fraction of QGs labeled, as shown in Figure 1 (for NDCG) and Figure 8 (for Outlier-NDCG). We also note that GBDTs have better NDCG than non-pretrained DL models

[2]Code is provided at `https://anonymous.4open.science/r/ltr-pretrain-3C01/README.md`.

Table 2: **Main result (Simulated implicit binary user feedback):** When $X\%$ (MSLRWEB30K=8.9%, Yahoo Set1=5.3%, and in Istella_S=25.2%) of query groups are assigned a few binary labels, pretrained DL method outperform both GBDT and non-pretrained DL models. We generate labels with $\tau_{\text{target}} = 4.5$ (Section 4.1.2). ♣ means that pretrained models are significantly better in NDCG than non-pretrained models, measured by a t-test with significance $p < 0.05$.

Method	MSLRWEB30K (↑)	Yahoo Set1 (↑)	Istella (↑)
	NDCG (↑)		
GBDT	0.346 ± 0.001	0.617 ± 0.000	0.602 ± 0.000
No Pretrain DL	0.333 ± 0.001	0.621 ± 0.000	0.608 ± 0.002
Pretrained DL	**0.356 ± 0.004**♣	**0.622 ± 0.001**	**0.613 ± 0.002**♣
	Outlier-NDCG (↑)		
GBDT	0.288 ± 0.000	0.540 ± 0.000	0.678 ± 0.000
No Pretrain DL	0.238 ± 0.026	0.536 ± 0.003	0.687 ± 0.015
Pretrained DL	**0.289 ± 0.012**	**0.543 ± 0.002**♣	**0.699 ± 0.003**

in most of these regimes. This shows that there exist scenarios with a limited supply of labeled query groups, where self-supervised pretraining is the factor that allows deep learning models to outperform GBDTs. We provide a detailed comparison between semi-supervised and supervised GBDT models in Appendix A.4

4.1.2 Simulated implicit binary user feedback

Here we simulate a scenario when users of an LTR system provide binary implicit feedback for the items, like the clicking of an advertisement on a webpage. This kind of feedback is usually infrequent in an LTR system, as it is the result of a user choosing to spend more time or resources on an item in a presented list. Therefore, most query groups will not have any labels, and even labeled query groups will only have a few positively labeled items. On the other hand, such labels are often cheaper to collect than crowdsourced labels, so they are used in many industry use cases.

Dataset. To simulate this scenario, we follow the methodology from Yang et al. (2022), to generate independent stochastic binary labels for each item from its true relevance label for training and validation sets of each of the public datasets. Note that we still use the true (latent) relevance labels in the test set for evaluation. This models a scenario where we observe binary labels that are noisy observations of a true latent label, but the task is to use these noisy labels to learn to rank according to the true labels. The details of this binarizing transformation are provided in Appendix A.2.4. In this transformation, a parameter, τ_{target}, implicitly controls how sparse the binary labels are (larger is more sparse). We select $\tau_{\text{target}} = 4.5$, to obtain 8.9%, 5.3%, and 25.2% labeled query groups which contain at least one positive label in MSLR, Yahoo Set1, Istella_S datasets, respectively. We present detailed data statistics in Table 11 of Appendix.

Methodology. We reuse the methodology of Section 4.1.1, except we use a linear scoring head for pretrained models to improve stability and performance (see Appendix A.2.3). Similar to Section 4.1.1, we report the best of multiple models for each method and dataset (see Section 4.1.1 for details).

Results. In Table 2, we present the test NDCG and Outlier-NDCG metrics of the models under this setting. We see that pretrained rankers significantly outperform GBDTs and non-pretrained DL rankers in this simulated binary user feedback setting with up to, even with 25.2% labeled query groups (Istella_S). For MSLRWEB30K and Yahoo Set1 datasets, the second best model is the GBDT, and the non-pretrained DL ranker has a much worse Outlier-NDCG. This aligns with recent observations that GBDTs perform better than DL models in supervised training with datasets containing irregularities McElfresh et al. (2023). However, pretraining helps DL rankers close this gap, and at times beat GBDTs in outlier performance. We also present results for $\tau_{\text{target}} \in \{4.25, 5.1\}$ in Appendix A.3.5. There, we find that making the labels sparser ($\tau_{\text{target}} = 5.1$) increases the relative improvement of pretrained DL rankers, while in a less sparse

Table 3: **Main result (Real-world dataset):** % improvement of rankers pretrained by SimSiam and SimCLR-Rank over GBDTs on a large-scale proprietary online shopping dataset. Unsupervised pretraining improves performance (1) overall (full dataset) (internally, 2% is a big improvement), and (2) on outliers.

Method	**NDCG** (↑)	**Outlier-NDCG** (↑)
	Δ% change w.r.t. GBDT	
GBDT	+0.00%	+0.00%
No Pretrain DL	+3.82% ± 0.14%	-6.97% ± 0.43%
SimCLR-Rank	+4.00% ± 0.14%	-4.85% ± 0.44%
SimSiam	**+5.64% ± 0.14%**	**+2.75% ± 0.43%**

setting ($\tau_{\text{target}} = 4.25$) GBDTs dominate. This suggests that pretrained DL models outperform GBDTs when learning from rare implicit binary user feedback, as is common in search and recommendation systems.

4.2 Large-scale real-world dataset results

In this section we test whether the results from the simulated label scarce settings above, especially Section 4.1.2, translate to a large-scale real-world LTR problem.

Dataset. Our proprietary dataset is derived from the online shopping logs of a large online retailer, numbering in the tens of millions of query groups. In this dataset, query groups consist of items presented to a shopper if they enter a search query. We assign each item a purchase label: label is 1 if the shopper purchased the item, or 0 if the shopper did not purchase the item. Hence, this is a real-world instance of the implicit binary user feedback setting simulated in Section 4.1.2. Further, only about 10% of the query groups have items with non-zero labels. See Appendix A.2.1 for more dataset details.

Methodology. We compare the performance of pretrained DL models–SimCLR-Rank and SimSiam–with GBDT and non-pretrained DL models. SimSiam and SimCLR-Rank are first pretrained on all the query groups before being finetuned on the labeled query groups with some non-zero purchase labels. The GBDT and the non-pretrained models are only trained on the labeled data. More details about these models and training strategies is given in Appendix A.2.5.

Results. Our results are summarized in Table 3 and all numbers are given as relative percentage improvements (Δ%) over the GBDT ranker. First, note that all the DL models outperform GBDT models in terms of NDCG. Amongst them, the pretrained model using SimSiam performs the best with a gain of over 5.5%. This is substantial, given that in large-scale industry datasets, even a 2% improvement in NDCG is considered significant. In terms of Outlier-NDCG, the story is very different. Here, the non-pretrained DL model and SimCLR-Rank performs up to -7% worse than the GBDT. However, SimSiam pretraining allows the DL ranker to overcome this shortcoming and beat the outlier performance of the GBDT. This verifies our hypothesis from Section 4.1.2 that pretraining can help DL models outperform GBDTs on large-scale real-world LTR problems, especially with implicit user feedback. Lastly, we see a significant performance difference between SimSiam and SimCLR-Rank. We investigate this in an ablation with public datasets in Section 4.3.

4.3 Ablations: Choices for pretrained models in LTR

Our experiments suggest that the best pretraining and finetuning strategies for the LTR problem are different from the known best practices for image or text tasks. We make a few observations. (1) SimSiam and SimCLR-Rank perform significantly differently depending on the dataset. (2) Combining models pretrained by SimCLR-Rank and SimSiam can produce a model that is competitive against SimCLR-Rank and SimSiam across all datasets, producing a unified approach that may be performant across many settings. (3) Full finetuning of the model performs much better than *linear probing*, which only finetunes a linear layer over frozen pretrained embeddings. In the remainder of this section, we elaborate on these findings in more detail.

(1) SimCLR-Rank and SimSiam perform substantially differently depending on the dataset. SimSiam and SimCLR-Rank perform significantly differently depending on the dataset, which is different

from the observations of Chen & He (2021), who proposed SimSiam and found that it performs similarly to SimCLR derivatives. We next compare SimCLR-Rank with SimSiam at different levels of label scarcity.

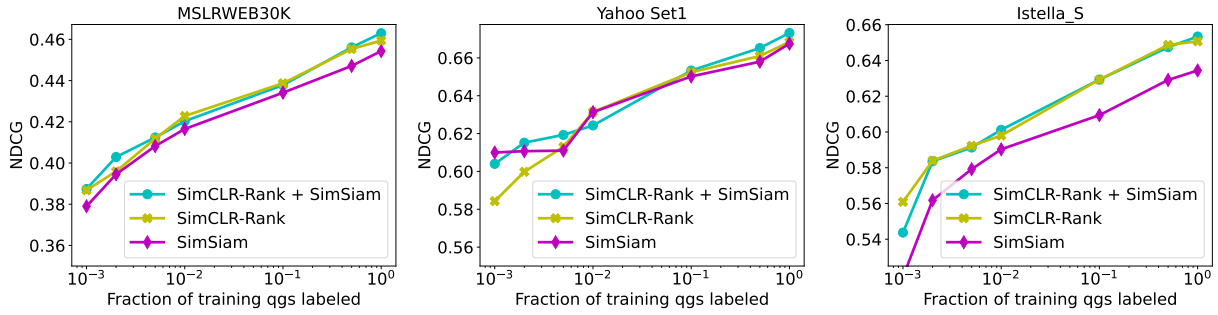

Figure 3: Comparison of SimCLR-Rank and SimSiam pretraining strategies on three public datasets with varying fraction of labeled query groups. Neither method consistently dominate the other in all the datasets. However, by combining SimCLR-Rank and SimSiam encoders, we can produce models that are competitive across all datasets. Data points are averages over 3 trials.

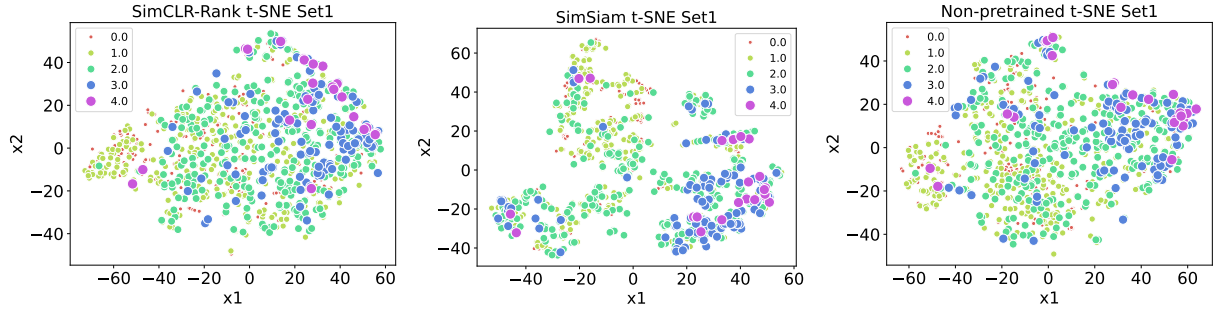

Figure 4: t-SNE plots of SimCLR-Rank, SimSiam, and a non-pretrained model trained on 0.1% of the data for the Yahoo Set1 dataset. Both SimCLR-Rank and SimSiam use the zeros augmentation ($p = 0.1$) to remove augmentation choice as a confounding variable. Relevance label is indicated by marker size/color. We see that SimCLR-Rank and SimSiam embeddings are qualitatively very different. Qualitatively, SimCLR-Rank and SimSiam embeddings separate different relevance labels better than the non-pretrained model. For this setting, SimSiam has the best downstream performance, and also gives the most clustered embeddings.

Methodology. The dataset and methodology follows that of Section 4.1.1, except we let (1) SimCLR-Rank use the Gaussian augmentation with scale 1, and (2) SimSiam use the zeroing augmentation with probability 0.1. These augmentations generally perform well for their respective methods. Experiments illustrating the effect of each augmentation method are included in Appendix A.3.6.

Results. In Figure 3, SimCLR-Rank performs better on MSLRWEB30K and Istella_S, while SimSiam performs better on Yahoo Set1 when labels are scarce. Figure 4 shows the t-SNE projections of sub-sampled item embeddings learned by SimCLR-Rank, SimSiam, and pre-final layer of a fully supervised model for Yahoo Set1. SimSiam exhibit qualitatively more clustered embeddings than SimCLR-Rank, particularly for high relevance labels. Similar t-SNE plots for other datasets are in Appendix A.3.3

(2) Combining models pretrained by SimCLR-Rank and SimSiam produces a model that has the strengths of both. To make steps towards a single recommendation for unsupervised pretraining in LTR, we unify SimCLR-Rank and SimSiam into a single method "SimCLR-Rank + SimSiam".

Methodology. To produce a "SimCLR-Rank + SimSiam" model, we finetune a linear layer over the embeddings from a finetuned SimCLR-Rank concatenated with the embeddings from a finetuned SimCLR-Rank model. More details about how we combine the models is given in Appendix A.3.7.

Table 4: **Finetuning**: Comparison of fine tuning strategies on MSLRWEB30K dataset (LP = Linear Probing, MP = Multilayer Probing, FF = Full Finetuning). Full finetuning consistently performs best or near-best, while linear probing *and* multilayer probing perform poorly.

Method	**SimSiam**	**SimCLR-Rank**
	NDCG (↑)	
LP	0.2679 ± 0.0007	0.3219 ± 0.0224
MP	0.2764 ± 0.0001	0.3890 ± 0.0011
FF	**0.3935 ± 0.0034**	**0.3959 ± 0.0022**
	Outlier-NDCG (↑)	
LP	0.1803 ± 0.0033	0.2304 ± 0.0332
MP	0.1749 ± 0.0023	**0.2969 ± 0.0009**
FF	**0.3149 ± 0.0119**	0.2892 ± 0.0025

Results. In Figure 3 we find that "SimCLR-Rank + SimSiam" performs equal to or better than SimCLR-Rank and SimSiam across many different percentages of labeled query groups. While the combined model does not uniformly dominate across all labeled query group percentages, it is a promising first step towards a single unified method for unsupervised pretraining in ranking.

(3) Full finetuning outperforms linear probing. Linear probing is a popular finetuning strategy in text and image domains, where it produces good results (Chen & He, 2021; Chen et al., 2020; Peters et al., 2019; Kumar et al., 2022). We next compare it with the full finetuning strategy in tabular LTR.

Methodology. We use the methodology from Section 3.1 to compare 3 finetuning strategies: linear probing (LP), full-finetuning (FF), and multilayer probing (MP) for SimSiam and SimCLR-Rank. MP tunes a 3-layer MLP head on top of the frozen pretrained embeddings. Experimental details are in Appendix A.2.3.

Results. Table 4 provides results on MSLRWEB30K dataset. We find that in terms of NDCG of both SimCLR-Rank and SimSiam, LP is the worst and FF is the best. Interestingly, MP comes as a close second in SimCLR-Rank, but it performs poorly on SimSiam. Our results on other datasets are similar (Appendix A.3.2). We thus recommend full fine-tuning as a stable and performant strategy. To explain the above observations, we we can look again at the t-SNE projections of the embeddings generated by the SimSiam and SimCLR-Rank encoders (pretrained using the zeros augmentation with $p = 0.1$ to remove augmentation choice as a confounding factor, finetuned on 0.1% of the data) and the pre-final layer of a model trained only on 0.1% of the data on a randomly sampled 1000 samples from the full training set (Figure 4).

These plots offer the following explanations for the two phenomena. (1) The embeddings produced by the SimSiam/SimCLR-Rank are not sorted by relevance score in the projection space, unlike the fully supervised encoder. This suggests that a linear ranker cannot use the pretrained embeddings to predict the true relevance labels. (2) MP works better for SimCLR-Rank than SimSiam because SimCLR-Rank's embeddings are more evenly spread out (less collapsed) than those of SimSiam. This allows an MLP trained on the SimCLR-Rank embeddings to distinguish the items with different relevance labels.

5 Conclusion

We study the learning-to-rank (LTR) problem with tabular data under a scarcity of labeled data—a common scenario in real-world practical LTR systems. Prior works on supervised learning with tabular data have shown that GBDTs outperform deep learning (DL) models, especially on datasets with outliers. In this paper, we find that in the label-scarce setting, DL pretraining methods can exploit available unlabeled data to obtain new state-of-the art performance. Through experiments with public LTR datasets and a real-world large-scale online shopping dataset, we show that pretraining allows DL rankers to outperform GBDT rankers, especially on outlier data. Finally, we provide guidelines for pretraining on LTR datasets. There remain several open questions around pretraining for tabular and LTR data. It is still unknown whether we can achieve knowledge transfer across different tabular datasets, as is commonly done in image and text domains. It will also be useful to investigate LTR-specific pretraining methods which can further improve the ranking performance.

References

Qingyao Ai, Keping Bi, Jiafeng Guo, and W Bruce Croft. Learning a deep listwise context model for ranking refinement. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pp. 135–144, 2018.

Qingyao Ai, Xuanhui Wang, Sebastian Bruch, Nadav Golbandi, Michael Bendersky, and Marc Najork. Learning groupwise multivariate scoring functions using deep neural networks. In *Proceedings of the 2019 ACM SIGIR international conference on theory of information retrieval*, pp. 85–92, 2019.

Sercan Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. 2021.

Dara Bahri, Heinrich Jiang, Yi Tay, and Donald Metzler. Scarf: Self-supervised contrastive learning using random feature corruption. *arXiv preprint arXiv:2106.15147*, 2021.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. *Advances in neural information processing systems*, 32, 2019.

Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE transactions on neural networks and learning systems*, 2022.

Amanda Bower, Hamid Eftekhari, Mikhail Yurochkin, and Yuekai Sun. Individually fair ranking. *arXiv preprint arXiv:2103.11023*, 2021.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Sebastian Bruch, Masrour Zoghi, Michael Bendersky, and Marc Najork. Revisiting approximate metric optimization in the age of deep neural networks. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, pp. 1241–1244, 2019.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pp. 89–96, 2005.

Christopher JC Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81, 2010.

Shaofeng Cai, Kaiping Zheng, Gang Chen, HV Jagadish, Beng Chin Ooi, and Meihui Zhang. Arm-net: Adaptive relation modeling network for structured data. In *Proceedings of the 2021 International Conference on Management of Data*, pp. 207–220, 2021.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pp. 129–136, 2007.

Olivier Chapelle and Yi Chang. Yahoo! learning to rank challenge overview. In *Proceedings of the learning to rank challenge*, pp. 1–24. PMLR, 2011.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Wei Chen, Tie-Yan Liu, Yanyan Lan, Zhi-Ming Ma, and Hang Li. Ranking measures and loss functions in learning to rank. *Advances in Neural Information Processing Systems*, 22, 2009.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15750–15758, 2021.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pp. 7–10, 2016.

Sajad Darabi, Shayan Fazeli, Ali Pazoki, Sriram Sankararaman, and Majid Sarrafzadeh. Contrastive mixup: Self-and semi-supervised learning for tabular domain. *arXiv preprint arXiv:2108.12296*, 2021.

Domenico Dato, Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Nicola Tonellotto, and Rossano Venturini. Fast ranking with additive ensembles of oblivious and non-oblivious regression trees. *ACM Transactions on Information Systems (TOIS)*, 35(2):1–31, 2016.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Kevin Duh and Katrin Kirchhoff. Learning to rank with partially-labeled data. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 251–258, 2008.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pp. 1189–1232, 2001.

Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017.

Songwei Ge, Shlok Mishra, Chun-Liang Li, Haohan Wang, and David Jacobs. Robust contrastive learning using negative samples with diminished semantics. *Advances in Neural Information Processing Systems*, 34:27356–27368, 2021.

Weifeng Ge. Deep metric learning with hierarchical triplet loss. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 269–285, 2018.

Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, Ankita Rajaram Naik, Pengshan Cai, and Alfio Gliozzo. Re2g: Retrieve, rerank, generate. *arXiv preprint arXiv:2207.06300*, 2022.

Gregory Goren, Oren Kurland, Moshe Tennenholtz, and Fiana Raiber. Ranking robustness under adversarial document manipulations. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 395–404, 2018.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34:18932–18943, 2021.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in Neural Information Processing Systems*, 35:507–520, 2022.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.

Paul Hager, Martin J. Menten, and Daniel Rueckert. Best of both worlds: Multimodal contrastive learning with tabular and imaging data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23924–23935, June 2023.

Ehsan Hajiramezanali, Nathaniel Lee Diamant, Gabriele Scalia, and Max W Shen. Stab: Self-supervised learning for tabular data. In *NeurIPS 2022 First Table Representation Workshop*, 2022.

Shuguang Han, Xuanhui Wang, Mike Bendersky, and Marc Najork. Learning-to-rank with bert in tf-ranking. *arXiv preprint arXiv:2004.08476*, 2020.

Ben Harwood, Vijay Kumar BG, Gustavo Carneiro, Ian Reid, and Tom Drummond. Smart mining for deep metric learning. In *Proceedings of the IEEE international conference on computer vision*, pp. 2821–2829, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. Tabllm: Few-shot classification of tabular data with large language models. In *International Conference on Artificial Intelligence and Statistics*, pp. 5549–5581. PMLR, 2023.

Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. *Advances in neural information processing systems*, 32, 2019.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020.

Sergei Ivanov and Liudmila Prokhorenkova. Boost then convolve: Gradient boosting meets graph neural networks. *arXiv preprint arXiv:2101.08543*, 2021.

Alan Jeffares, Tennison Liu, Jonathan Crabbé, Fergus Imrie, and Mihaela van der Schaar. Tangos: Regularizing tabular neural networks through gradient orthogonalization and specialization. *arXiv preprint arXiv:2303.05506*, 2023.

Jongheon Jeong and Jinwoo Shin. Consistency regularization for certified robustness of smoothed classifiers. *Advances in Neural Information Processing Systems*, 33:10558–10570, 2020.

Thorsten Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 217–226, 2006.

Arlind Kadra, Marius Lindauer, Frank Hutter, and Josif Grabocka. Well-tuned simple nets excel on tabular datasets. *Advances in neural information processing systems*, 34:23928–23941, 2021.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

Guolin Ke, Jia Zhang, Zhenhui Xu, Jiang Bian, and Tie-Yan Liu. Tabnn: A universal neural network solution for tabular data. 2018.

Guolin Ke, Zhenhui Xu, Jia Zhang, Jiang Bian, and Tie-Yan Liu. Deepgbm: A deep learning framework distilled by gbdt for online prediction tasks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 384–394, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Nikita Kitaev, Steven Cao, and Dan Klein. Multilingual constituency parsing with self-attention and pre-training. *arXiv preprint arXiv:1812.11760*, 2018.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*, 2022.

Branislav Kveton, Ofer Meshi, Masrour Zoghi, and Zhen Qin. On the value of prior in online learning to rank. In *International Conference on Artificial Intelligence and Statistics*, pp. 6880–6892. PMLR, 2022.

Sudarshan Dnyaneshwar Lamkhede and Christoph Kofler. Recommendations and results organization in netflix search. In *Proceedings of the 15th ACM Conference on Recommender Systems*, pp. 577–579, 2021.

Kyungmin Lee and Jinwoo Shin. Rényicl: Contrastive representation learning with skew rényi divergence. *Advances in Neural Information Processing Systems*, 35:6463–6477, 2022.

Roman Levin, Valeriia Cherepanova, Avi Schwarzschild, Arpit Bansal, C Bayan Bruss, Tom Goldstein, Andrew Gordon Wilson, and Micah Goldblum. Transfer learning with deep tabular models. *arXiv preprint arXiv:2206.15306*, 2022.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1754–1763, 2018.

Jianghao Lin, Yanru Qu, Wei Guo, Xinyi Dai, Ruiming Tang, Yong Yu, and Weinan Zhang. Map: A model-agnostic pretraining framework for click-through rate prediction. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1384–1395, 2023.

Guang Liu, Jie Yang, and Ledell Wu. Ptab: Using the pre-trained language model for modeling tabular data. *arXiv preprint arXiv:2209.08060*, 2022.

Hao Liu, Yixin Chen, Bradley Fritz, and Christopher King. Tabcontrast: A local-global level method for tabular contrastive learning. In *NeurIPS 2023 Second Table Representation Learning Workshop*, 2023.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Adversarial multi-task learning for text classification. *arXiv preprint arXiv:1704.05742*, 2017.

Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.

Zhaocheng Liu, Qiang Liu, Haoli Zhang, and Yuntian Chen. Dnn2lr: Interpretation-inspired feature crossing for real-world tabular data. *arXiv preprint arXiv:2008.09775*, 2020.

Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, and Salvatore Trani. Post-learning optimization of tree ensembles for efficient ranking. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 949–952, 2016.

Haoran Luo, Fan Cheng, Heng Yu, and Yuqi Yi. Sdtr: Soft decision tree regressor for tabular data. *IEEE Access*, 9:55999–56011, 2021.

Yuanfei Luo, Hao Zhou, Wei-Wei Tu, Yuqiang Chen, Wenyuan Dai, and Qiang Yang. Network on network for tabular data classification in real-world applications. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2317–2326, 2020.

Kushal Majmundar, Sachin Goyal, Praneeth Netrapalli, and Prateek Jain. Met: Masked encoding for tabular data. *arXiv preprint arXiv:2206.08564*, 2022.

David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pp. 152–159, 2006.

Duncan McElfresh, Sujay Khandagale, Jonathan Valverde, Ganesh Ramakrishnan, Micah Goldblum, Colin White, et al. When do neural nets outperform boosted trees on tabular data? *arXiv preprint arXiv:2305.02997*, 2023.

Bhaskar Mitra, Nick Craswell, et al. An introduction to neural information retrieval. *Foundations and Trends® in Information Retrieval*, 13(1):1–126, 2018.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018.

Jaehyun Nam, Woomin Song, Seong Hyeon Park, Jihoon Tack, Sukmin Yun, Jaehyung Kim, and Jinwoo Shin. Semi-supervised tabular classification via in-context learning of large language models. In *Workshop on Efficient Systems for Foundation Models@ ICML2023*, 2023a.

Jaehyun Nam, Jihoon Tack, Kyungmin Lee, Hankook Lee, and Jinwoo Shin. Stunt: Few-shot tabular learning with self-generated tasks from unlabeled tables. *arXiv preprint arXiv:2303.00918*, 2023b.

Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G Azzolini, et al. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091*, 2019.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. Multi-stage document ranking with bert. *arXiv preprint arXiv:1910.14424*, 2019.

Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4004–4012, 2016.

Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. *Advances in neural information processing systems*, 31, 2018.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Feifei Pan, Mustafa Canim, Michael Glass, Alfio Gliozzo, and James Hendler. End-to-end table question answering via retrieval-augmented generation. *arXiv preprint arXiv:2203.16714*, 2022.

Liang Pang, Jun Xu, Qingyao Ai, Yanyan Lan, Xueqi Cheng, and Jirong Wen. Setrank: Learning a permutation-invariant ranking model for information retrieval. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pp. 499–508, 2020.

Gustavo Penha, Arthur Câmara, and Claudia Hauff. Evaluating the robustness of retrieval pipelines with query variation generators. In *European conference on information retrieval*, pp. 397–412. Springer, 2022.

Matthew E Peters, Sebastian Ruder, and Noah A Smith. To tune or not to tune? adapting pretrained representations to diverse tasks. *arXiv preprint arXiv:1903.05987*, 2019.

Sergei Popov, Stanislav Morozov, and Artem Babenko. Neural oblivious decision ensembles for deep learning on tabular data. *arXiv preprint arXiv:1909.06312*, 2019.

Dong-Hyun Lee Pseudo-Label. The simple and efficient semi-supervised learning method for deep neural networks. In *ICML 2013 Workshop: Challenges in Representation Learning*, pp. 1–6, 2013.

Tao Qin and Tie-Yan Liu. Introducing LETOR 4.0 datasets. *CoRR*, abs/1306.2597, 2013. URL `http://arxiv.org/abs/1306.2597`.

Zhen Qin, Le Yan, Honglei Zhuang, Yi Tay, Rama Kumar Pasumarthi, Xuanhui Wang, Mike Bendersky, and Marc Najork. Are neural rankers still outperformed by gradient boosted decision trees? 2021.

Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. Product-based neural networks for user response prediction. In *2016 IEEE 16th international conference on data mining (ICDM)*, pp. 1149–1154. IEEE, 2016.

Steffen Rendle. Factorization machines. In *2010 IEEE International conference on data mining*, pp. 995–1000. IEEE, 2010.

Mamshad Nayeem Rizve, Kevin Duarte, Yogesh S Rawat, and Mubarak Shah. In defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning. *arXiv preprint arXiv:2101.06329*, 2021.

Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples. *arXiv preprint arXiv:2010.04592*, 2020.

Ivan Rubachev, Artem Alekberov, Yury Gorishniy, and Artem Babenko. Revisiting pretraining objectives for tabular deep learning. *arXiv preprint arXiv:2207.03208*, 2022.

Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. *Advances in neural information processing systems*, 29, 2016.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

Ira Shavitt and Eran Segal. Regularization learning networks: deep learning for tabular datasets. *Advances in Neural Information Processing Systems*, 31, 2018.

Weiwei Shi, Yihong Gong, Chris Ding, Zhiheng MaXiaoyu Tao, and Nanning Zheng. Transductive semi-supervised deep learning using min-max features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 299–315, 2018.

Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C Bayan Bruss, and Tom Goldstein. Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. *arXiv preprint arXiv:2106.01342*, 2021.

Yangqiu Song and Dan Roth. Unsupervised sparse vector densification for short text similarity. In *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 1275–1280, 2015.

Yumin Suh, Bohyung Han, Wonsik Kim, and Kyoung Mu Lee. Stochastic class-based hard example mining for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7251–7259, 2019.

Yi Sui, Tongzi Wu, Jesse Cresswell, Ga Wu, George Stein, Xiaoshi Huang, Xiaochen Zhang, Maksims Volkovs, et al. Self-supervised representation learning from random data projectors. *arXiv preprint arXiv:2310.07756*, 2023.

Baohua Sun, Lin Yang, Wenhan Zhang, Michael Lin, Patrick Dong, Charles Young, and Jason Dong. Supertml: Two-dimensional word embedding for the precognition on structured tabular data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 0–0, 2019.

Robin Swezey, Aditya Grover, Bruno Charron, and Stefano Ermon. Pirank: Scalable learning to rank via differentiable sorting. *Advances in Neural Information Processing Systems*, 34:21644–21654, 2021.

Tahir Syed and Behroz Mirza. Self-supervision for tabular data by learning to predict additive homoskedastic gaussian noise as pretext. *ACM Transactions on Knowledge Discovery from Data*, 2023.

Yuandong Tian, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning dynamics without contrastive pairs. In *International Conference on Machine Learning*, pp. 10268–10278. PMLR, 2021.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Talip Ucar, Ehsan Hajiramezanali, and Lindsay Edwards. Subtab: Subsetting features of tabular data for self-supervised representation learning. *Advances in Neural Information Processing Systems*, 34:18853–18865, 2021.

Vikas Verma, Thang Luong, Kenji Kawaguchi, Hieu Pham, and Quoc Le. Towards domain-agnostic contrastive learning. In *International Conference on Machine Learning*, pp. 10530–10541. PMLR, 2021.

Ellen M Voorhees. The trec robust retrieval track. In *ACM SIGIR Forum*, volume 39, pp. 11–20. ACM New York, NY, USA, 2005.

Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of the web conference 2021*, pp. 1785–1797, 2021.

Weiyao Wang, Byung-Hak Kim, and Varun Ganapathi. Regclr: A self-supervised framework for tabular representation learning in the wild. *arXiv preprint arXiv:2211.01165*, 2022a.

Xiao Wang, Haoqi Fan, Yuandong Tian, Daisuke Kihara, and Xinlei Chen. On the importance of asymmetry for siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16570–16579, 2022b.

Zifeng Wang and Jimeng Sun. Transtab: Learning transferable tabular transformers across tables. *Advances in Neural Information Processing Systems*, 35:2902–2915, 2022.

Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *Proceedings of the IEEE international conference on computer vision*, pp. 2840–2848, 2017.

Chen Wu, Ruqing Zhang, Jiafeng Guo, Wei Chen, Yixing Fan, Maarten de Rijke, and Xueqi Cheng. Certified robustness to word substitution ranking attack for neural ranking models. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 2128–2137, 2022a.

Chen Wu, Ruqing Zhang, Jiafeng Guo, Yixing Fan, and Xueqi Cheng. Are neural ranking models robust? *ACM Transactions on Information Systems*, 41(2):1–36, 2022b.

Chen Wu, Ruqing Zhang, Jiafeng Guo, Maarten De Rijke, Yixing Fan, and Xueqi Cheng. Prada: practical black-box adversarial attacks against neural ranking models. *ACM Transactions on Information Systems*, 41(4):1–27, 2023.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pp. 1192–1199, 2008.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10687–10698, 2020.

Shuo Yang, Sujay Sanghavi, Holakou Rahmanian, Jan Bakus, and Vishwanathan SVN. Toward understanding privileged features distillation in learning-to-rank. *Advances in Neural Information Processing Systems*, 35: 26658–26670, 2022.

David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pp. 189–196, 1995.

Andrew Yates, Rodrigo Nogueira, and Jimmy Lin. Pretrained transformers for text ranking: Bert and beyond. In *Proceedings of the 14th ACM International Conference on web search and data mining*, pp. 1154–1156, 2021.

Chao Ye, Guoshan Lu, Haobo Wang, Liyao Li, Sai Wu, Gang Chen, and Junbo Zhao. Ct-bert: learning better tabular representations through cross-table pre-training. *arXiv preprint arXiv:2307.04308*, 2023.

Jinsung Yoon, Yao Zhang, James Jordon, and Mihaela van der Schaar. Vime: Extending the success of self-and semi-supervised learning to tabular domain. *Advances in Neural Information Processing Systems*, 33:11033–11043, 2020.

Hai-Tao Yu. Pt-ranking: A benchmarking platform for neural learning-to-rank, 2020.

Chaoning Zhang, Kang Zhang, Chenshuang Zhang, Trung X Pham, Chang D Yoo, and In So Kweon. How does simsiam avoid collapse without negative samples? a unified understanding with self-supervised contrastive learning. In *International Conference on Learning Representations*, 2021.

Peng Zhang, Dawei Song, Jun Wang, and Yuexian Hou. Bias-variance decomposition of ir evaluation. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pp. 1021–1024, 2013.

Bingzhao Zhu, Xingjian Shi, Nick Erickson, Mu Li, George Karypis, and Mahsa Shoaran. Xtab: Cross-table pretraining for tabular transformers. *arXiv preprint arXiv:2305.06090*, 2023.

Yitan Zhu, Thomas Brettin, Fangfang Xia, Alexander Partin, Maulik Shukla, Hyunseung Yoo, Yvonne A Evrard, James H Doroshow, and Rick L Stevens. Converting tabular data into images for deep learning with convolutional neural networks. *Scientific reports*, 11(1):11325, 2021.

Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. Rankt5: Fine-tuning t5 for text ranking with ranking losses. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2308–2313, 2023.

A Appendix

A.1 Detailed discussion of related work

Learning-To-Rank. In our paper, we focus on the traditional LTR setting where the features are all numeric (tabular data). However, there is a line of work in LTR where raw text is also an input. In this case, one can leverage large language models in the ranking setting (Zhuang et al., 2023; Han et al., 2020; Yates et al., 2021; Nogueira et al., 2019; Mitra et al., 2018).

In tabular LTR problems, the dominant models currently used are gradient boosted decision trees (GBDTs) (Friedman, 2001), which are not deep learning models. GBDT models, which perform well on tabular data, are adapted to the LTR setting via losses that are surrogates for ranking metrics like NDCG. Surrogate losses (including LambdaRank/RankNet (Burges, 2010) and PiRank (Swezey et al., 2021)) are needed because many important ranking metrics (like NDCG) are non-differentiable. The combination of tree-based learners and ranking losses has become the de-facto standard in ranking problems, and deep models have yet to outperform them convincingly (Qin et al., 2021; Joachims, 2006; Ai et al., 2019; Bruch et al., 2019; Ai et al., 2018; Pang et al., 2020).

Deep tabular models. Given the success of neural methods in many other domains, there have been many attempts to adapt deep models to the tabular domain. Borisov et al. (2022) categorize existing techniques for using deep neural networks over tabular data into four types: (1) **Encoding-based methods**

such as VIME (Yoon et al., 2020), SCARF (Bahri et al., 2021), IGTD (Zhu et al., 2021), and SuperTML (Sun et al., 2019); (2) **Novel hybrid architectures** such as DeepFM (Guo et al., 2017), xDeepFM (Lian et al., 2018), and many others Cheng et al. (2016); Frosst & Hinton (2017); Ke et al. (2018; 2019); Popov et al. (2019); Luo et al. (2020); Liu et al. (2020); Ivanov & Prokhorenkova (2021); Luo et al. (2021); (3) **Transformer-based architectures** including SAINT (Somepalli et al., 2021), TabNet (Arik & Pfister, 2021), FT-Transformer Gorishniy et al. (2021), TabTransformer (Huang et al., 2020), and ARM-Net Cai et al. (2021)); (4) **Regularized DNNs** (Shavitt & Segal, 2018; Kadra et al., 2021). For instance, TANGOS introduced special tabular-specific regularization to try to improve deep models' performance (Jeffares et al., 2023).

Self-supervised learning. Self-supervised learning (SSL) or unsupervised pretraining has improved performance in settings where there is a significant source of unlabeled data like text (Devlin et al., 2018) and images (Chen et al., 2020). In SSL, deep models are first pretrained on perturbed unlabeled data using self-supervised tasks to learn useful representations for the data. Then these models are finetuned for a downstream task with labeled data. Finetuning often takes one of two forms: (1) linear probing (a popular finetuning strategy in text and images (Chen & He, 2021; Chen et al., 2020; Peters et al., 2019)),where we freeze the pretrained model and only update the linear head during supervised finetuning, and (2) full finetuning, where we update the whole model during supervised finetuning (Devlin et al., 2018). Sometimes a mix of the two is used (Kumar et al., 2022). The core idea behind prominent SSL approaches like SimSiam and SimCLR is to carefully perturb input training samples, and train a representation that is consistent for perturbations of the same sample. This provides robustness to natural perturbations and noise in data (Hendrycks et al., 2019).

Inspired by the success of pretraining and self-supervised learning in images and text, several works show how to apply SSL to unlabeled tabular data. One strategy is to corrupt tabular data and train a deep model to reconstruct it (Yoon et al., 2020; Majmundar et al., 2022; Ucar et al., 2021; Nam et al., 2023b; Lin et al., 2023; Syed & Mirza, 2023; Hajiramezanali et al., 2022). Another approach is to use contrastive losses, which have been highly successful in the image domain (Chen et al., 2020). These methods are also applicable to our case because tabular data, like image data, is often composed of fixed-dimensional real vectors (Verma et al., 2021; Bahri et al., 2021; Lee & Shin, 2022; Hager et al., 2023; Liu et al., 2023; Darabi et al., 2021). Rubachev et al. (2022) evaluate a variety of different pretraining methods for tabular learning across many different datasets, finding that there is not a clear state of the art.

Robustness in LTR. There has been prior work on studying worst-case behavior (robustness) of rankers (Voorhees, 2005; Zhang et al., 2013; Goren et al., 2018; Wu et al., 2022b;a; Penha et al., 2022). Some previous metrics measure a model's robustness against adversarial attack (Goren et al., 2018; Wu et al., 2022a; 2023). Others measure the model's per-query performance variance on a dataset (Voorhees, 2005; Zhang et al., 2013; Wu et al., 2022b). Our outlier metric, Outlier-NDCG, is a departure from previous work because it is not directly a measure of robustness and it is possible for a model to perform *better* on outlier data.

Transfer learning. We summarize some related work on transfer learning in tabular domain. One direction revolves around pretraining models on common columns across many datasets (Zhu et al., 2023; Wang & Sun, 2022; Sui et al., 2023; Ye et al., 2023). Another direction leverages LLMs (large language models) to do few-shot tabular learning (Hegselmann et al., 2023; Liu et al., 2022; Nam et al., 2023a). Pretrained model features are often useful across datasets. For example, text features are transferable between text datasets (Devlin et al., 2018), which has contributed to the recent success of foundation models (Devlin et al., 2018; Touvron et al., 2023; Kojima et al., 2022; Brown et al., 2020). Unfortunately, it is still unclear how to achieve transferability in tabular LTR, where previous work has leveraged common columns (or at least column documentation) to achieve transfer learning (Levin et al., 2022; Hegselmann et al., 2023). However most LTR datasets do not have common columns or publicly-documented column meanings (Qin & Liu, 2013; Chapelle & Chang, 2011; Lucchese et al., 2016). Developing a method to transfer features in tabular LTR is an important open problem.

A.2 Experimental Details

A.2.1 Dataset statistics

In table Table 5, we provide the details of our three public datasets and the proprietary real-world dataset. Note that for the public dataset experiments we sub-sample the pre-training dataset in different ways (as described in Sections 3, 4.1.1 and 4.1.2) to create the training dataset and simulate scarcity of labeled samples. However, we use the same validation dataset without and with the labels for the self-supervised pretraining and supervised learning stages, respectively. Note that in all these datasets, the higher the label value the higher the relevance of the item.

Table 5: Dataset statistics on the MSLRWEB30K, Yahoo Set1, Istella_S, and proprietary datasets.

	Number of query groups					**Number**	**List**
	Pretraining		***Supervised learning***			**of**	**of**
Dataset	*Pretrain*	*SSL Val*	*Train*	*Val*	*Test*	**features**	**labels**
MSLRWEB30K	18151	6072	Subset Pretrain	SSL Val	6072	136	{0,1,2,3,4}
Yahoo Set1	14477	2147	Subset Pretrain	SSL Val	5089	700	{0,1,2,3,4}
Istella_S	19200	7202	Subset Pretrain	SSL Val	6523	220	{0,1,2,3,4}
Proprietary dataset	35.6M	1.62M	3.68M	171k	1.08M	63	{0,1}

A.2.2 Outliers query group details

Here we detail how we select the outlier query groups for the Outlier-NDCG metrics (Section 2.1). For our proprietary online shopping dataset, there are known outlier features that cause low quality predictions. These outliers arise from noise introduced in various stages in the data pipeline. We have mitigation strategies in place against these feature outliers, but they are often imperfect. Because we already know some of the outliers features, we deem any query group with items with outlier features as an outlier. These outliers comprise roughly 10% of the test set, which is still a large number of query groups (much larger than the number of query groups in the outlier datasets of the public datasets).

For our public datasets, we systematically select outliers as follows: we generate a histogram with 100 bins for each feature across the *validation dataset*. For example, Istella (Lamkhede & Kofler, 2021) has 220 features, so we have 220 different histograms. For each histogram, we scan from left to right on the bins until we have encountered at least G empty bins in a row, and if there is less than 1% of the validation set above this bin, then all the feature values above this bin are considered outliers. We also repeat this process right to left. Any test query group containing items with outlier feature values is labeled an outlier query group, and placed in the test outlier dataset. In Figure 5, we present an example histogram of a feature in Istella_S dataset.

Because different datasets have differently-sized typical gaps, we tune G for each dataset (MSLR, Yahoo, Istella) such that the resulting percentage of outlier queries is as close to 1% of the test set as we can get. MSLRWEB30K has $G = 5$, with 0.65% (40/6072) outlier queries, Yahoo has $G = 20$, with 1.4% (30/2147) outlier queries, and Istella has $G = 32$, with 0.46% (34/7202) outlier queries. 1% is a hyperparameter that can be tuned according to the user's goals.

A.2.3 Model Settings and Training Details

Pretrained DL model.

Pretraining: (1) the encoder used for pretraining is the tabular ResNet (Gorishniy et al., 2021) with three ResNet blocks, with the final linear layer removed, (2) pretraining is done on the entire dataset with learning rate 0.0005 using Adam (Kingma & Ba, 2014) (3) with SimSiam/SimCLR-Rank/SimCLR-Sample, we tuned among four different augmentations for each pretraining method: randomly zeroing out features ("Zeroing") with probabilities 0.1 or 0.7, and Gaussian noise with scale 1.0 or 2.0, (4) for SCARF we tune the corruption

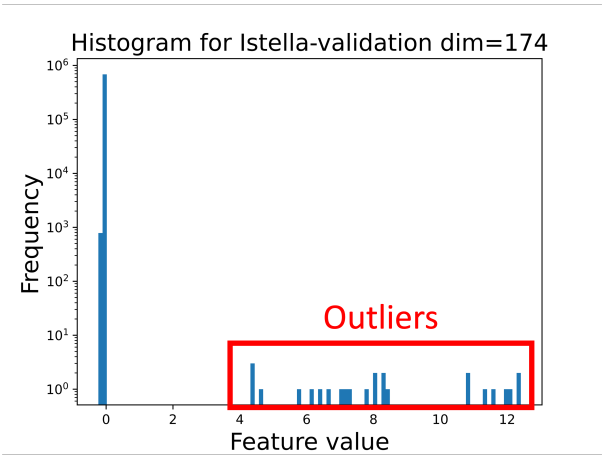

Figure 5: An example of outlier detection in Istella_S (Lucchese et al., 2016) for our Outlier-NDCG metric.

probability in $\{0.3, 0.6, 0.9\}$, (5) for VIME-self we tune the corruption probability in $\{0.3, 0.5, 0.7\}$, (6) for DACL+ we tune the mixup amount in $\{0.3, 0.6, 0.9\}$, (7) for SubTab we divide input features into 4 subsets with 75% overlap, and tune the masking probability in $\{0.3, 0.6, 0.9\}$, (8) we pretrain for 300 epochs for all methods, and (9) we use a batch size of roughly 200000 items for SimCLR-Rank, SimSiam, VIME-self, and SubTab as they are less GPU memory intensive while using a batch size of roughly 2000 items for SimCLR/DACL+/SCARF which are more GPU memory intensive.

We also evaluate SAINT (Somepalli et al., 2021), a transformer-based pretrainining method. For the architecture, we follow (Somepalli et al., 2021), using $L = 4$ layers, dropout of 0.1, attention heads $h = 4$, with a self-attention query/key/value dimension of 16 and a intersample attention query/key/value dimension of 64. For augmentation, we follow (Somepalli et al., 2021) and use a CutMix mask parameter of 0.3 and MixUp parameter of 0.2. Again following (Somepalli et al., 2021) we weight the denoising loss 10 times more than the contrastive loss. For all other minor architectural details, we follow the code of (Somepalli et al., 2021). We pretrain for 5 epochs with a batch size of roughly 100 items due to GPU memory contraints and also the time cost (one epoch of saint took 300x longer than an epoch of SimCLR-Rank in MSLRWEB30K, 6000x longer in Yahoo Set1, and 1000x longer in Istella). We use a learning rate of 5e-5.

Finetuning: (1) finetuning is done on the labeled train set by adding a three-layer MLP to the top of the pretrained model and training only this head for 100 epochs and then fully finetuning for 100 epochs using Adam with a learning rate of 5e-5, (2) we use an average batch size of roughly 1000 items (may vary based on query group size), (3) we use the LambdaRank loss (Burges, 2010), (4) we use the validation set to perform early stopping (i.e. using the checkpoint that performed best on the validation set to evaluate on the test set).

For SAINT we follow the finetuning process of (Somepalli et al., 2021) in using the output of the CLS embedding as the representation for an item, and tune the learning rate in {5e-5, 1e-5, 5e-6} as we found the performance to be sensitive to this setting.

Implicit binary user feedback setting. We reuse the methodology of Section 4.1.1, except we use a scoring head of one linear layer (as opposed to a three layer MLP) for pretrained models. We found that this improved stability and performance in the binary label setting.

No Pretrained DL model. There are a few rankers we evaluated for the no-pretrain DL baseline: (1) a 3-layer tabular ResNet from (Gorishniy et al., 2021) with a three-layer MLP on top of it, and was trained for 300 epochs on the labeled training set with learning rate tuned in {0.01, 0.001, 0.0001} using Adam, with early stopping using the validation set, (2) a DeepFM (Guo et al., 2017) model following the model architecture in the paper with learning rate tuned in {0.05, 0.005, 0.0005}, and (3) a DCNv2 (Wang et al., 2021) model with learning rate tuned in {0.05, 0.005, 0.0005} and cross layers tuned in {2,3,4}. DeepFM and DCNv2 were designed to use categorical features, and in MSLRWEB30K/Yahoo Set1/Istella_S do not

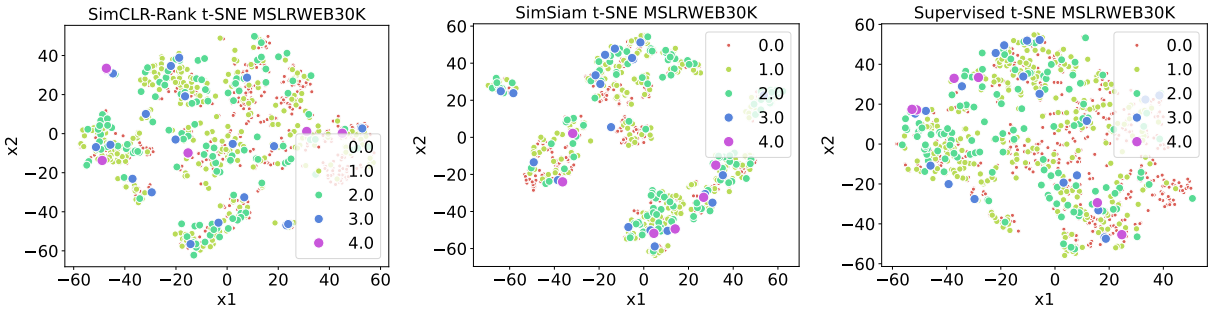

Figure 6: We plot the t-SNE plots of embeddings produced by three different encoders for the MSLRWEB30K dataset: (1) pretrained by SimCLR-Rank, (2) pretrained by SimSiam, (3) trained via supervised training on the entire training set on roughly 1000 samples from MSLR. Marker size and color indicates relevance. We find (1) SimCLR-Rank/SimSiam cluster different relevances effectively but do not order them as well as the supervised encoder. (2) SimCLR-Rank produces more spread-out embeddings with less defined clusters than SimSiam.

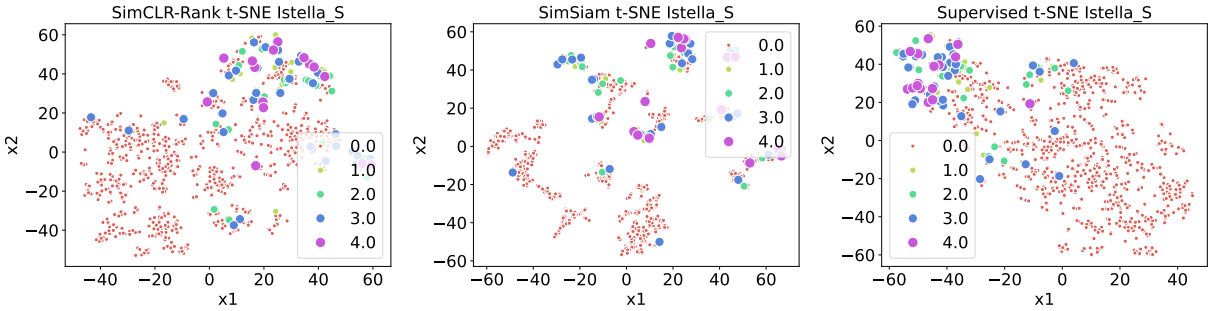

Figure 7: We plot the t-SNE plots of embeddings produced by three encoders: (1) pretrained by SimCLR-Rank, (2) pretrained by SimSiam, (3) trained via supervised training on the entire training set on roughly 1000 samples from Istella_S. Marker size and color indicates relevance. We find (1) SimCLR-Rank/SimSiam cluster different relevances effectively but do not order them as well as the supervised encoder. (2) SimCLR-Rank produces more spread-out embeddings with less defined clusters than SimSiam.

document their categorical features (if any). We performed an analysis of the datasets and found 5, 48, 6 categorical features in MSLRWEB30K/Yahoo Set1/Istella_S respectfully each with cardinality at most 3. We embedded these categorical features into embeddings of dimension 8, following the rule-of-thumb in (Wang et al., 2021).

GBDT model. The GBDT ranker is the one from `lightgbm` (Ke et al., 2017) and we grid search the number of leaves in {31, 96, 200} minimum data in leaf in {20, 60, 200}, and column sample in {0.5, 0.9, 1.0} individually for each data point (for a total of 9 difference choices), while letting the rest of the parameters be the default in `lightgbm` (our tuning strategy is similar to Qin et al. (2021)). We use the LambdaRank loss to train the models.

GBDT + pseudolabeling. We add pseudo-labeled GBDTs as a semi-supervised GBDT baseline. For GBDTs with pseudo-labeling (GBDT-pseudo), we propose the following algorithm as there are no references to our knowledge. First, we train the GBDT on the labeled train set, and then use this GBDT to pseudo-labels for the unlabeled train set. A GBDT ranker outputs real-valued scores. To convert real-valued scores into relevance scores, we take the resulting scores and evenly bin them into 31 buckets. The scores converted to 0 are the lowest bucket, the scores converted to 1 are the second lowest bucket, and so on. We choose 31

because that is the largest number of labels allowed for GBDT ranking in lightgbm (the reason is because the LambdaRank loss requires the calculation of 2^{label} which can get very large).

GBDT + PCA. For GBDT+PCA, we take the entire training set's features and learn a PCA on it with {5, 10, 15} components, similar to (Duh & Kirchhoff, 2008), to produce {5, 10, 15} new features (i.e. we tune among 5, 10, 15). We then train on the labeled train set, with both the original features and the new PCA features together.

GBDT + PCA + pseudolabeling. We also introduce a baseline where we perform PCA on the full training dataset, and then perform pseudolabeling as above. We tune the PCA similarly to the GBDT + PCA baseline.

SimCLR-Rank + GBDT/SimSiam + GBDT. We also explore using pretrained neural embeddings as features in the GBDT. To do this, we used the pretrained SimCLR-Rank (or SimSiam) model to produce embeddings over the labeled data, concatenated the original features with the embeddings, and then trained a GBDT on this new dataset. We use the same tuning grid as for the original GBDT model.

Comparing finetuning strategies. When linear probing (LP), we freeze the pretrained model and update only a linear head on top of it for 200 epochs. In multilayer probing (MP), we freeze the pretrained model and update a 3-layer MLP head on top of it for 200 epochs. In full finetuning (FF), we use the finetuning strategy from Section 4.1.1. We use the validation set for early stopping.

All reported values for Outlier-NDCG are those achieved by the corresponding ranker with best overall NDCG. We report all results as averages over 3 trials, and use single V100 GPUs from a shared cluster to run the experiments.

A.2.4 Details of Binary Label generation used in Section 4.1.2

Following the methodology from Yang et al. (2022), we generate stochastic binary labels y from the relevance labels r as $y = \mathbb{1}\{t \cdot r + G_1 > t \cdot \tau_{\text{target}} + G_0\}$ where t is a temperature parameter, G_1, G_0 are standard Gumbels, and τ_{target} is a parameter controlling how sparse the binary labels are. Higher τ_{target} leads to higher label sparsity. Yang et al. (2022) show that y is 1 with probability $\sigma(t \cdot (r - \tau_{\text{target}}))$ where σ is the sigmoid function. We set $t = 4$ as in Yang et al. (2022). For a given τ_{target}, we produce produce a new dataset for each of MSLRWEB30K, Yahoo Set1, and Istella_S where we convert the relevance labels in the training/validation sets to binary labels and keep the true relevance labels in the test set.

A.2.5 Large industry-scale dataset experimental details

For our DL ranker, first we pass the features (numerical and categorical) through some featurizers. Then we use a simple 4 hidden layer MLP, with and ReLU activation, to predict a single scalar score for each (query, item) pair. We also use dropout and batch normalization in input and hidden layers.

For the SimCLR pretraining stage, we remove the last linear layer of the above ranker to create the main encoder to produce the pretrained embeddings. Projector encoder here is a 2 hidden layer MLP, with ReLU activation and no bias, which outputs an input dimensional embedding. We also do not use any normalization or dropout in this projector.

For SimSiam we use the same embedder and a similar projector. The only difference in the projector being batch normalization in the hidden layers and batch normalization without affine learnable parameters in the output layer. Additionaly, SimSiam also employs a predictor MLP, with a single hidden layer and ReLU activation, and with the same input and output sizes. It uses batch normalization and no bias in the hidden layer and no input and hidden dropouts.

For pretraining, we used an augmentation which randomly zeroed out some features. We also tried randomly swapping some features between items in the same batch or query group, but they didn't perform as well as zeros augmentation. During the fine-tuning stage we discard the projectors and predictor and add back the linear layer to complete the non-pretrained ranker structure. DL rankers are trained/fine-tuned using PiRank loss Swezey et al. (2021), batchsize of 1000, AdamW optimizer with learning rate of 0.00001, weight decay of 0.01 and cosine scheduling with warmup. All training stages and experiments were conducted on machines

with 4 GPUs, where we ran 100k iterations with ddp and picked the best checkpoint based on validation metric.

GBDT rankers were trained with LambdaRank using the `lightgbm` package. They were tuned with Bayesian hyperparameter optimization strategy by tuning number of leaves, learning rate, and minimum data in leaf in a reasonably large range with a maximum of 150 jobs. We used default values of the library for the other hyperparameters.

A.3 Additional results

A.3.1 Runtime comparisons of pre-training strategies

In Table 6, we provide the runtime comparisons between SimSiam, SimCLR, and SimCLR-Rank (new method).

Table 6: Seconds per epoch comparison between pretraining methods. Average over 3 trials. The encoder we use for pretraining is the tabular ResNet (Gorishniy et al., 2021) with the final linear layer taken out.

		Seconds per epoch		
Method	Complexity	MSLRWEB30K	Yahoo Set1	Istella
SimCLR	$O(B^2L^2)$	102.81 ± 1.14	14.45 ± 0.04	69.26 ± 0.16
SimCLR-Rank (new)	$O(BL^2)$	9.4700 ± 0.02	1.970 ± 0.01	5.950 ± 0.04
SimSiam	$O(BL)$	0.8700 ± 0.01	0.130 ± 0.00	0.730 ± 0.00

A.3.2 Additional Results on linear probing vs finetuning

Here we provide more results on the question of linear probing vs full finetuning, as discussed in Section 4.3. In Tables 7 and 8 we compare different probing strategies for all datasets when using SimSiam and SimCLR-Rank, respectively, as the pretraining method.

A.3.3 Additional Results on SimCLR-Rank vs SimSiam

In this section we give additional t-SNE plots of the embeddings learned by SimCLR-Rank and SimSiam encoders, and the pre-final layer of the fully supervised ranker (trained with all labels) on MSLRWEB30K dataset (Figure 6), and Istella_S dataset (Figure 7). For the Yahoo Set1 dataset, SimSiam performs the best, but for the above two datasets SimCLR-Rank is the best.

A.3.4 Additional Results: Simulated crowdsourced label setting

Here we provide the results on label scarcity for Outlier-NDCG in Figure 8. Result interpretation is given in Section 4.1.1.

A.3.5 Additional Results for Simulated Implicit Binary user feedback setting

In this part of the appendix we provide additional results on binary label generation as detailed in Section 4.1.2. In Tables 9 and 10 we give the dataset statistics when using $\tau_{\text{target}} = 4.25$ and $\tau_{\text{target}} = 5.1$, respectively. For these two settings, the GBDT baseline does not contain semi-supervised runs as we found that semi-supervision tends to generally degrade GBDT performance; see Appendix A.4. The "No Pretrain" baseline uses the tabular ResNet architecture.

In Tables 12 and 13 we give the comparison between pretrained and non-pretrained rankers following the methodology in Section 4.1.2 and find that GBDTs perform the best when binary labels are less sparse ($\tau_{\text{target}} = 4.25$) and pretrained DL rankers perform the best when binary labels are sparser ($\tau_{\text{target}} = 5.1$). This strongly suggests that pretrained DL models outperform GBDTs when the labels arise from binary user feedback that is moderately uncommon or rare (as in online shopping, recommendations, or search).

Table 7: SimSiam under different finetuning strategies on NDCG/Outlier-NDCG, averaged over 3 trials (LP = Linear Probing, MP = Multilayer Probing, FF = Full Finetuning). We find that linear probing *and* MLP probing perform extremely poorly (except in Yahoo Set1, where MLP probing performs well).

Method	MSLRWEB30K (↑)	Yahoo Set1 (↑)	Istella (↑)
	NDCG (↑)		
LP	0.2679 ± 0.0007	0.6089 ± 0.0032	0.3805 ± 0.0042
MP	0.2764 ± 0.0001	**0.6137 ± 0.0022**	0.4484 ± 0.0020
FF	**0.3935 ± 0.0034**	0.6107 ± 0.0035	**0.5618 ± 0.0049**
	Outlier-NDCG (↑)		
LP	0.1803 ± 0.0033	0.5088 ± 0.002	0.4407 ± 0.0388
MP	0.1749 ± 0.0023	0.5157 ± 0.0080	0.5324 ± 0.0002
FF	**0.3149 ± 0.0119**	**0.52 ± 0.0133**	**0.6348 ± 0.0164**

Table 8: SimCLR-Rank under different finetuning strategies on NDCG/Outlier-NDCG, averaged over 3 trials (LP = Linear Probing, MP = Multilayer Probing, FF = Full Finetuning). Multilayer probing performs moderately worse than full finetuning, and linear probing performs much worse than all other strategies.

Method	MSLRWEB30K (↑)	Yahoo Set1 (↑)	Istella (↑)
	NDCG (↑)		
LP	0.3219 ± 0.0224	0.5202 ± 0.0093	0.4029 ± 0.0073
MP	0.3890 ± 0.0011	0.5942 ± 0.0016	0.5813 ± 0.0006
FF	**0.3959 ± 0.0022**	**0.6022 ± 0.0013**	**0.5839 ± 0.0013**
	Outlier-NDCG (↑)		
LP	0.2304 ± 0.0332	0.3811 ± 0.0048	0.4717 ± 0.0166
MP	**0.2969 ± 0.0009**	0.4888 ± 0.0046	**0.6369 ± 0.0045**
FF	0.2892 ± 0.0025	**0.5143 ± 0.0055**	0.6352 ± 0.0140

A.3.6 Ablation: Data Augmentation Techniques

In Tables 14 and 15, we show the performance of each augmentation choice on SimCLR-Rank and SimSiam, respectively. We use the methodology described in Section 4.1.1, and have 0.2% of training QGs labeled.

A.3.7 Combining SimCLR-Rank and SimSiam

Here we describe how we constructively combined the SimCLR-Rank and SimSiam to get the competitive results in Section 4.3. Following the setting of Appendix A.2.3, we train (1) a SimCLR-Rank model with gaussian augmentation and noise of scale 1.0, and (2) a SimSiam model with zeroing augmentation and corruption probability 0.1.

After pretraining the SimCLR-Rank and SimSiam models, we finetune each of them on the labeled data and keep the checkpoints with the best validation NDCG. Then we combine these two models by concatenating the final embedding layers together—if each model's penultimate embedding dimension is h ($h = 136$ for us), then the concatenated embedding is of dimension $2 \times h$. Then we pass this concatenated model through a batchnorm (to normalize the scales of each embedding to prevent scale issues), pass it through dropout layer of probability 0.7 (if the dataset is MSLRWEB30K or Istella_S) or 0.0 (if the dataset is Yahoo Set1), and map this vector (of dimension $2 \times h$) to a score of dimension 1 using a linear layer. We finetune only the final linear layer.

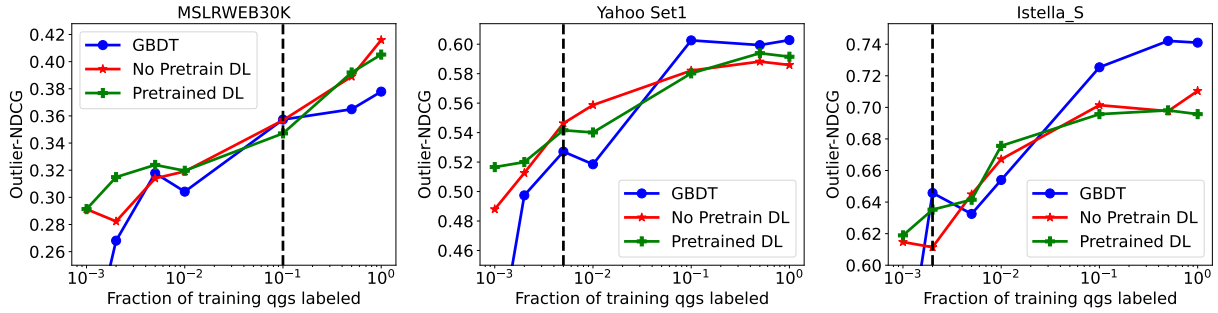

Figure 8: **Simulated crowdsourcing of labels:** We compare Outlier-NDCG (↑) pretrained rankers, non-pretrained DL rankers, and GBDT rankers as we change the percentage of training query groups that are labeled. To the left of the black dotted line, pretrained rankers perform the best. Points are averages over three trials. To the left of the black dotted vertical line, pretrained rankers are (1) significantly better on outliers than GBDTs at the $p = 0.05$ level using a one-sided t-test, and (2) on average better on outliers than all other non-pretrained methods.

Table 9: Train and validation dataset statistics on MSLRWEB30K, Yahoo Set1, and Istella_S after binary label generation with $\tau_{\text{target}} = 4.25$. The test set is left untouched. Labeled QGs are those query groups that have at least one item with $y = 1$.

Dataset	# query groups		# labeled QGs		# items per QG		# positives per QG	
	train	val	train	val	train	val	train	val
MSLRWEB30K	18151	6072	15.9%	16.2%	124.35	122.38	1.4%	1.4%
Yahoo Set1	14477	2147	11.1%	10.8%	30.02	30.32	6.2%	6.0%
Istella_S	19200	7202	49.0%	49.5%	106.36	94.98	1.4%	1.6%

A.4 Semi-supervised GBDT

Here we provide a more detailed comparison between Supervised GBDTs, GBDT + pseudolabeling, and GBDT + PCA + pseudolabeling in Figure 9 for the simulated crowdsourcing of labels setting and in Table 17 for the simulated implicit binary user feedback setting. We follow the experimental methodology in Appendix A.2.3. We find that for the most part, semi-supervised learning does not benefit GBDT rankers.

A.5 Quantitative measures of embedding quality

We provide a quantitative measure of embedding quality in this section. We attempt to verify that the focus on in-query group negatives helps improve embedding quality. To do this, we compare the embeddings produced by a SimCLR-Rank model and a SimCLR-Sample model (SimCLR-Sample is the same as SimCLR-Rank except the negatives are randomly taken from the batch rather than being in-query group), both under zeroes augmentation with $p = 0.1$, and measure how spread out the embeddings are within a query group for each method. In Table 16, we find that SimCLR-Rank embeddings are consistently more spread out within a query group than SimCLR-Sample, showing that in-query group focus leads to a measurable impact on the distinguish-ability of embeddings within a query group. This may help explain why SimCLR-Rank performs better than SimCLR-Sample.

Table 10: Train and validation dataset statistics on MSLRWEB30K, Yahoo Set1, and Istella_S after binary label generation with $\tau_{\text{target}} = 5.1$. The test set is left untouched. Labeled QGs are those query groups that have at least one item with $y = 1$.

Dataset	# query groups		# labeled QGs		# items per QG		# positives per QG	
	train	val	train	val	train	val	train	val
MSLRWEB30K	18151	6072	1.1%	1.3%	124.35	122.38	0.8%	0.7%
Yahoo Set1	14477	2147	0.6%	0.6%	30.02	30.32	5.4%	5.1%
Istella_S	19200	7202	3.0%	2.7%	106.36	94.98	1.1%	1.3%

Table 11: Train and validation dataset statistics on MSLRWEB30K, Yahoo Set1, and Istella_S after binary label generation with $\tau_{\text{target}} = 4.5$. The test set is left untouched. Labeled QGs are those query groups that have at least one item with $y = 1$.

Dataset	# query groups		# labeled QGs		# items per QG		# positives per QG	
	train	val	train	val	train	val	train	val
MSLRWEB30K	18151	6072	8.9%	8.9%	124.35	122.38	1.1%	1.1%
Yahoo Set1	14477	2147	5.3%	5.4%	30.02	30.32	5.9%	5.4%
Istella_S	19200	7202	25.2%	25.2%	106.36	94.98	1.3%	1.3%

Table 12: We compare pretrained models to non-pretrained models in the binary label setting with $\tau_{\text{target}} = 4.25$ (Section 4.1.2) on NDCG averaged over three trials. We follow the methodology in Section 4.1.2.

Method	MSLRWEB30K (↑)	Yahoo Set1 (↑)	Istella (↑)
	NDCG (↑)		
GBDT	0.3616 ± 0.0000	0.6233 ± 0.0000	0.6251 ± 0.0000
No Pretrain DL	0.3552 ± 0.0015	0.6297 ± 0.0005	0.6147 ± 0.0006
Pretrained DL	0.3602 ± 0.0007	0.6272 ± 0.0024	0.6243 ± 0.0007
	Outlier-NDCG (↑)		
GBDT	0.2792 ± 0.0000	0.5352 ± 0.0000	0.7393 ± 0.0000
No Pretrain DL	0.3142 ± 0.0063	0.5389 ± 0.0036	0.6631 ± 0.0102
Pretrained DL	0.2923 ± 0.0079	0.5471 ± 0.0050	0.6945 ± 0.0041

Table 13: We compare pretrained models to non-pretrained models in the binary label setting with $\tau_{\text{target}} = 5.1$ (Section 4.1.2) on NDCG averaged over three trials. We follow the methodology in Section 4.1.2. ♣ indicates metrics on which pretrained rankers outperform non-pretrained rankers significantly via a $p < 0.05$ t-test.

Method	MSLRWEB30K	Yahoo Set1	Istella
	NDCG (↑)		
GBDT	0.2844 ± 0.0000	0.5782 ± 0.0000	0.5638 ± 0.0000
No Pretrain DL	0.3432 ± 0.0065	0.5703 ± 0.0246	0.5722 ± 0.0080
Pretrained DL	**0.3564 ± 0.0054**♣	**0.6031 ± 0.0064** ♣	**0.599 ± 0.0014** ♣
	Outlier-NDCG (↑)		
GBDT	0.2416 ± 0.0000	0.5011 ± 0.0000	0.6640 ± 0.0000
No Pretrain DL	0.2441 ± 0.0089	0.4593 ± 0.0317	0.6334 ± 0.0054
Pretrained DL	**0.2553 ± 0.0161**	**0.54 ± 0.0055**♣	**0.6730 ± 0.0090** ♣

Table 14: We compare different augmentation strategies for SimCLR-Rank. The methodology used is the one in Section 4.1.1, with 0.2% of training QGs labeled.

Augmentation	MSLRWEB30K	Yahoo Set1	Istella
	NDCG (↑)		
Zeroing (p=0.1)	0.3830 ± 0.0007	**0.6022 ± 0.0013**	0.5820 ± 0.0024
Zeroing (p=0.7)	0.3737 ± 0.0025	0.5998 ± 0.0053	0.5646 ± 0.0048
Gaussian noise (scale=1.0)	**0.3959 ± 0.0022**	0.5998 ± 0.0026	**0.5839 ± 0.0013**
Gaussian noise (scale=2.0)	0.3907 ± 0.0025	0.5953 ± 0.0043	0.5809 ± 0.0010
	Outlier-NDCG (↑)		
Zeroing (p=0.1)	0.2782 ± 0.0036	**0.5143 ± 0.0055**	**0.6408 ± 0.0063**
Zeroing (p=0.7)	0.2730 ± 0.0054	0.5062 ± 0.0070	0.6141 ± 0.0064
Gaussian noise (scale=1.0)	**0.2892 ± 0.0025**	0.4963 ± 0.0024	0.6352 ± 0.0140
Gaussian noise (scale=2.0)	0.2886 ± 0.0096	0.4875 ± 0.0054	0.6327 ± 0.0191

Table 15: We compare different augmentation strategies for SimSiam. The methodology used is the one in Section 4.1.1, with 0.2% of training QGs labeled.

Augmentation	MSLRWEB30K	Yahoo Set1	Istella
	NDCG (↑)		
Zeroing (p=0.1)	**0.3935 ± 0.0034**	**0.6107 ± 0.0035**	0.5618 ± 0.0049
Zeroing (p=0.7)	0.3911 ± 0.0003	0.6076 ± 0.0072	**0.5660 ± 0.0047**
Gaussian noise (scale=1.0)	0.3782 ± 0.0093	0.6010 ± 0.0026	0.5612 ± 0.0060
Gaussian noise (scale=2.0)	0.3860 ± 0.0011	0.6100 ± 0.0089	0.5587 ± 0.0047
	Outlier-NDCG (↑)		
Zeroing (p=0.1)	**0.3149 ± 0.0119**	**0.5200 ± 0.0133**	**0.6348 ± 0.0164**
Zeroing (p=0.7)	0.3002 ± 0.0060	0.5081 ± 0.0097	0.6201 ± 0.0104
Gaussian noise (scale=1.0)	0.2585 ± 0.0371	0.4893 ± 0.0084	0.6145 ± 0.0160
Gaussian noise (scale=2.0)	0.2929 ± 0.0021	0.5163 ± 0.0101	0.5905 ± 0.0146

Table 16: The ratio of the in-query group mean absolute deviation over the overall mean absolute deviation of embeddings from SimCLR-Rank and SimCLR-Sample. These embeddings were calculated over the validation set of each dataset. We find that SimCLR-Rank has a higher ratio, meaning the embeddings are more spread out within a query group.

Method	MSLRWEB30K	Yahoo Set1	Istella
SimCLR-Rank	0.9953	0.9904	0.9915
SimCLR-Sample	0.9945	0.9881	0.9797

Table 17: We compare GBDT models in the binary label setting with $\tau_{\text{target}} = 4.5$ (Section 4.1.2) on NDCG averaged over three trials. We follow the methodology in Section 4.1.2.

Method	MSLRWEB30K	Yahoo Set1	Istella
	NDCG (↑)		
Supervised GBDT	0.3335 ± 0.0000	0.6168 ± 0.0000	0.6024 ± 0.0000
GBDT + PCA	0.3105 ± 0.0236	0.6141 ± 0.0022	0.5508 ± 0.0410
GBDT + pseudolabeling	0.3467 ± 0.0011	0.5868 ± 0.0012	0.5178 ± 0.0068
GBDT + PCA + pseudolabeling	0.3241 ± 0.0270	0.5944 ± 0.0012	0.5315 ± 0.0120

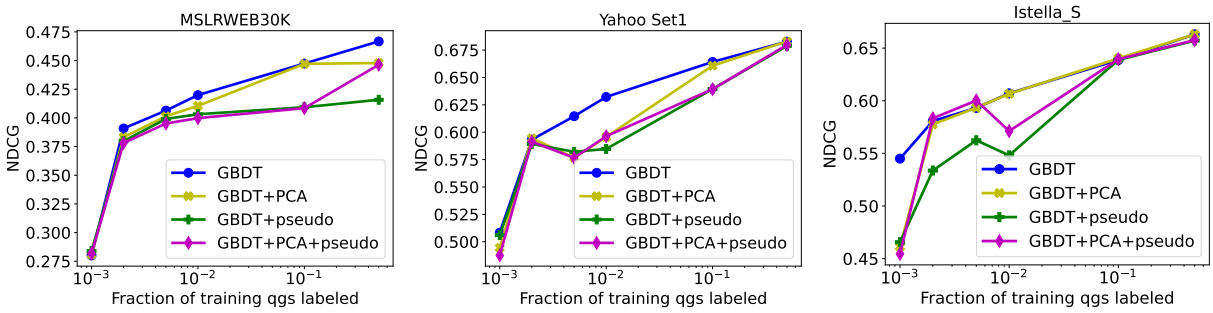

Figure 9: A comparison of Supervised GBDT, GBDT + PCA, GBDT + pseudolabeling, and GBDT + PCA + pseudolabeling. We find that semi-supervised learning may sometimes benefit GBDT rankers, though it is not very consistently beneficial.