Prior-Data Fitted Networks as Tabular Foundation Models for Ranking in Low-Data Settings

David Vos
University of Amsterdam
Amsterdam
The Netherlands
d.j.a.vos@uva.nl

Samarth Bhargav
Cohere
Amsterdam
The Netherlands
samarth.bhargav92@gmail.com

Maarten de Rijke
University of Amsterdam
Amsterdam
The Netherlands
m.derijke@uva.nl

Harrie Oosterhuis
University of Amsterdam
Amsterdam
The Netherlands
h.oosterhuis@uva.nl

Abstract

Learning to rank (LTR) traditionally requires large-scale training data to generalize effectively. In low-data domains where expert annotation is scarce, the performance of LTR methods degrades sharply. Foundation models have alleviated similar data dependencies in other domains via in-context learning, but a foundation model for ranking with tabular features has not been explored yet.

We propose prior-data fitted networks (PFNs) as a strong method for ranking in low-data settings. First, we demonstrate that PFNs, which are originally trained for classification, successfully outperform classification baselines on ranking data. Next, we evaluate PFNs as rankers, showing that they surpass state-of-the-art tuned baselines in low-data regimes. We introduce a novel sampling and inference scheme to obtain pairwise predictions from PFNs' native pointwise architecture, analogous to pairwise LTR. To address the limited context window of the transformers underlying PFNs, we propose a dynamic support set selection strategy for queries that scales PFNs beyond random subsampling. Our experimental results show that PFNs are an effective foundation model for ranking that provides significant gains when data is limited.

CCS Concepts

• **Information systems** → **Retrieval models and ranking**.

Keywords

Ranking, Foundational Models, Low-data Settings, Learning to Rank

ACM Reference Format:
David Vos, Samarth Bhargav, Maarten de Rijke, and Harrie Oosterhuis. 2026. Prior-Data Fitted Networks as Tabular Foundation Models for Ranking in Low-Data Settings. In *Proceedings of the 2026 International ACM SIGIR Conference on Innovative Concepts and Theories in Information Retrieval (ICTIR) (ICTIR '26), July 25, 2026, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3805713.3820419

1 Introduction

Ranking models are an essential part of information retrieval (IR) systems [8]. For a given query, ranking models score each document in a set of candidate documents, such that the ordering according to the predicted scores should provide the best ranking [27]. Traditionally, ranking scores are based on tabular features representing relevance signals, page properties, and interaction statistics [5, 29, 36]. Accordingly, many state-of-the-art ranking models have generally been gradient boosted decision trees (GBDTs) [4]. However, recently, neural networks have been proposed as competitive alternatives that can perform on par with GBDTs when applied correctly [38]. The success of transformer architectures has increased attention for text-based ranking [10], nevertheless, ranking based on tabular features remains prominent [5, 29, 36].

1.1 Towards a foundation model for ranking

Learning to rank (LTR) methods traditionally require massive labeled datasets [5, 27, 29, 36], which are prohibitively expensive in many settings. As a result, the performance of ranking models, whether GBDTs or neural networks, degrades sharply when less data is available for training (as confirmed by our experimental results). This poses a significant problem in low-data settings, e.g., when the task is niche, high-expertise, or personalized [34], or when privacy concerns limit data collection [49].

The broader machine learning field has resolved similar data-efficiency issues by transitioning from task-specific models toward general-purpose models, also known as *foundation models*, that reduce the need for large-scale domain- and task-specific training data; e.g., foundation models for weather prediction outperform traditional methods in data-sparse geographical areas [2]. In the IR field, general-purpose text-encoder models like BERT and LLMs like Qwen have drastically improved retrieval and (re-)ranking performance in out-of-domain settings, even with limited training data [11, 46, 54]. Similarly, large language models (LLMs) can tackle label sparsity by generating relevance judgments for text-based ranking [47, 50]; however, their judgment quality can be suboptimal, especially in niche, high-expertise settings [12, 43].

In contrast to text-based retrieval, there appears to be no existing work on foundation models for the ranking task on tabular features. Findings in other domains that rely on tabular numerical and categorical features show that foundation models trained on text have only had limited success [16, 52]. Consequently, as there are no foundation models for this setting, training models from scratch remains state-of-the-art for ranking on tabular data.

In this work, we propose to use *prior-data fitted networks* (PFNs) as a basis for the first foundational model for ranking with tabular features. PFNs are transformer-based models that are trained on large amounts of synthetic data [21, 31]. This allows the model to perform in-context learning (ICL) by simply including training samples in the context window of the transformer, thus without any model fitting and hyperparameter tuning (in contrast with LTR). Given their success in low-data tabular classification, PFNs appear to have great potential for tabular ranking in low-data settings.

ICTIR '26, July 25, 2026, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-2600-2/2026/07
https://doi.org/10.1145/3805713.3820419

1.2 Research questions

Tabular classification is similar to pointwise ranking, as both tasks make independent predictions for data samples. However, it is not self-evident that earlier findings on classification extend to pointwise ranking. Documents in ranking datasets are distributed per query, instead of independently. This means that the features of groups of documents are conditioned on the same query. Furthermore, ranking features are often sparse, with features such as *term overlap* often being 0, while the label distribution is heavily skewed towards non-relevant documents. PFNs are traditionally trained as classifiers. To assess whether PFNs can handle ranking datasets, we start by evaluating their performance as a relevance classifier. We compare a PFN classifier to GBDT and neural network-based classifier baselines on three major ranking datasets by evaluating the classification cross-entropy loss, and ask our first research question:

(RQ1) How do PFNs compare against GBDT and neural network-based classification methods on a relevance classification task on learning to rank datasets in low-data settings?

We find that PFNs outperform classification baselines in low-data settings on ranking datasets. To assess whether this finding translates to good ranking performance, we evaluate them on the ranking task, including a comparison with state-of-the-art ranking models. We ask our second research question:

(RQ2) How do *pointwise* PFNs compare against state-of-the-art ranking methods for ranking in low-data settings?

We find that PFNs outperform point-, pair-, and listwise ranking methods based on GBDTs and neural networks in low-data settings. As our PFN ranker so far treats documents independently, it can be considered a pointwise ranker. We find that all pointwise methods outperform methods with pair- and listwise objectives in low-data regimes [41]. As training data increases, pair- and listwise methods catch up to pointwise methods, including PFNs.

Previous work has found that LLM-based ranking works better with pairwise prompting [37], analogously, extracting pairwise PFN predictions could be fruitful. By concatenating pairs of document feature vectors, we are able to obtain pairwise predictions from a PFN, and subsequently, can create a ranking with a simple aggregation strategy. Hereby, we align PFNs with pairwise ranking approaches and ask our third research question:

(RQ3) How do *pairwise* PFN predictions compare against state-of-the-art ranking methods in low-data settings?

We find that pairwise PFN ranking outperforms state-of-the-art pair- and listwise baselines in low-data settings, performing slightly worse than pointwise methods. To address the transformer's context window bottleneck, we use the PFN's ICL capability by dynamically adjusting the support set of training documents for each query. We introduce a dynamic document selection strategy based on the similarity of queries in the training set, and investigate whether this increases ranking performance over random sub-sampling of training documents.

(RQ4) Can dynamic support set selection improve PFN ranking performance over random support set selection in data-rich settings?

1.3 Contributions

We make the following contributions: (i) We propose to use PFNs for ranking and show that they can outperform state-of-the-art GBDT and neural network baselines in low-data classification and ranking settings across three benchmark datasets. (ii) We introduce a pairwise sampling and inference scheme that adapts the native pointwise architecture of PFNs into a pairwise ranker by using a weighted win-rate aggregation strategy based on calibrated PFN uncertainty estimates. (iii) We propose a dynamic support set selection strategy for data-rich settings that uses k-nearest neighbor (kNN) query similarity. This strategy scales the effectiveness of in-context learning to large training datasets and makes PFNs competitive rankers in data-rich settings. (iv) Finally, to facilitate reproducibility, we release our code through a public repository.[1]

2 Related Work

2.1 Learning to rank

Learning to rank (LTR) concerns machine learning methods for optimizing ranking models, generally in a supervised manner [27]. Let $Q = \{q_1, \ldots, q_m\}$ be a set of queries. Each query q_i is associated with a list of candidate documents $\mathcal{D}_{q_i} = \{d_{i,1}, \ldots, d_{i,n}\}$, a list of corresponding relevance labels $Y_i = \{y_{i,1}, \ldots, y_{i,n}\}$, and a list of feature vectors $X_i = \{x_{i,1}, \ldots, x_{i,n}\}$ representing each document-query pair. The goal is to learn a scoring function $f(x)$ that assigns a ranking score to each query-document pair $s_{i,j} = f(q_i, d_{i,j})$ based on its features. Subsequently, rankings are created by sorting documents according to these scores. The goal of LTR methods is to find the optimal scoring function f such that the resulting rankings maximize a given ranking metric (e.g., NDCG [25]). LTR commonly uses heterogeneous numerical & categorical features that represent relevance signals, page properties, and interaction statistics [5, 29, 36].

LTR algorithms are typically categorized into three families of methods based on their loss function [27]: (i) Pointwise methods have loss functions that consider each document prediction independently. Thereby, they approach ranking as a regression or classification problem, which ignores the dependencies between predicted scores in resulting rankings. (ii) Pairwise optimize loss functions that consider the ordering of a pair of documents at a time. These functions are based on the differences between pairs of scores and aim to match their sign with the corresponding differences in the document labels. (iii) Listwise loss functions consider all document scores at once. They generally use a heuristic approximation of the sorting process, often for a ranking metric proxy.

As a parametrization of the ranking model, recent research has focused on the comparison between GBDTs and neural networks. While GBDT-based methods such as LambdaMART achieve state-of-the-art performance in most settings [4], neural networks have been shown to be competitive in certain settings [38]. To achieve optimal performance, LTR methods often require extensive training and hyperparameter optimization [24].

Our work on PFNs for ranking is in stark contrast with the entirety of the LTR field. Since, instead of fitting model weights to a training set, PFNs optimize for a task through in-context learning. This in-context learning paradigm also eliminates the need for hyperparameter tuning. Furthermore, the LTR field has ubiquitously

[1] https://github.com/davidvos/pfns-for-ranking

considered data-rich settings, to the best of our knowledge, our work is the first with a focus on low-data settings.

2.2 Ranking in low-data settings

There has been limited work on how well state-of-the-art LTR methods perform in low training data regimes [53]. To the best of our knowledge, this work is the first to identify the sharp drop in ranking performance of LTR methods as less data becomes available.

This high data requirement aligns with similar requirements from other tasks involving tabular data, where GBDTs have shown to require dense feature space coverage to determine optimal split points, which can lead to overfitting [56]. Neural network-based methods can perform worse than GBDTs in low-data settings, as they lack the inductive bias necessary to generalize well [18]. Unlike image or text data, tabular ranking features often possess irregular decision boundaries and lack a low-dimensional manifold structure. Neural networks, which are biased towards smooth functions, struggle to approximate these sharp, axis-aligned decision rules without massive datasets, leading to sensitivity to potentially uninformative features [3, 18, 42].

Robust GBDT variants such as YetiRank [19] and frameworks such as CatBoost [35] introduced ordered boosting and permutation-based techniques to overcome sensitivity to uninformative features. Approaches based on meta-learning have been applied to tabular data to adapt to new tasks with few examples [21, 51]. These methods often require computationally expensive optimization and extensive meta-training data, which may not always be available.

In contrast to tabular ranking, text-based retrieval has recently seen great improvements in low-data settings through foundation models. Examples are embedding models that generalize across domains, as well as LLMs that rank documents in zero-shot settings [45, 54]. However, these approaches rely on a semantic understanding of text and do not naturally generalize to the numerical features found in tabular LTR datasets. Recent work that has adapted LLMs to tabular data shows mixed results [20].

Our work on PFNs for ranking contrasts with work on GBDT, neural rankers, and LLMs. Unlike GBDTs and neural rankers, PFNs do not require fitting a model to specific datasets, allowing for adaptation to new data through in-context learning. Unlike LLMs, PFNs are not pre-trained on large amounts of textual data, but rather on synthetic data that represent a prior distribution of tabular data.

3 Background

In this section, we describe *prior-data fitted networks* (PFNs). For a comprehensive account, we refer to foundational work on the PFN architecture [31], the statistical foundations [32], and the performance of PFNs for tabular classification (TabPFN) [21].

3.1 Prior-data fitted networks

Tabular data is characterized by heterogeneous numerical and categorical features, often manually engineered, where the spatial or sequential relationships between columns have no inherent meaning [3]. Prior-data fitted networks (PFNs) can achieve state-of-the-art performance on tabular benchmarks by generalizing directly from a pre-training task without requiring gradient updates on task-specific data. A PFN is a transformer-based model trained on a large number of synthetic datasets that emulate real-world patterns. This pre-training process allows the model to learn a *prior over functions*. Once trained, the PFN performs *in-context learning* (ICL) to produce a posterior predictive distribution for new, unseen datasets. Unlike LLMs, where in-context learning typically involves textual prompts, ICL in the context of PFNs refers to providing the transformer with a set of tabular training samples as tokens, directly within its context window. This enables the model to do training and inference in a single forward pass, without the need for gradient updates on unseen datasets.

3.1.1 Synthetic data as a prior While classical supervised learning methods learn a predictive function from a fixed dataset, PFNs are models that *learn the learning algorithm itself* [31]. In this meta-learning framework, the goal is not to fit a single model to a single dataset, but to train a transformer that can recognize and generalize from the patterns found in any given data distribution. For example, a pattern could be "if feature X and Y are correlated this way, the label is likely Z". A model that learns this relationship between the input features and output label could generalize to all datasets with this pattern, rather than memorizing a pattern specific to a given dataset. PFNs are parameterized by a transformer architecture and trained offline on millions of synthetic datasets generated from *priors*, such as structural causal models (SCMs). During this phase, the transformer learns abstract statistical properties. PFNs can therefore be characterized as foundation models for tabular data, allowing a single model to generalize to a wide variety of tasks and datasets.

3.1.2 In-context inference The primary advantage of this approach is that at inference time, the PFN can adapt to real-world data without any model training because it performs in-context learning instead. The transformer treats a real-world dataset, to which we refer as the *support set* $S = \{(x_1, y_1), \ldots, (x_{n_s}, y_{n_s})\}$, and the features x_t of a test document as a single sequence of tokens:

$$\text{Input} = [\underbrace{x_1, y_1, x_2, y_2, \ldots, x_{n_s}, y_{n_s}}_{\text{Support set (examples)}}, \underbrace{x_t}_{\text{Test sample}}]. \tag{1}$$

By processing the support set consisting of real-world samples through its bi-directional self-attention layers, the model observes the relationships in the data and immediately outputs a prediction for x_t. This allows PFNs to both learn from training data and perform inference on a test sample in milliseconds [21].

PFNs have achieved promising results across a range of tabular tasks. For example, TabPFN has achieved state-of-the-art results on standard tabular classification benchmarks, often outperforming heavily tuned GBDTs and complex AutoML ensembles like AutoGluon [17, 21, 22]. These results extend beyond classification to time-series forecasting, where TabPFN-TS [23] has been shown to outperform specialized models, and to causal inference, where PFNs enable amortized causal discovery [28] and treatment effect estimation [40]. In contrast to these earlier works, we propose PFNs for ranking, where the (ranking) objective and unique data distributions pose new challenges.

3.1.3 Scaling PFNs to data-rich settings The promising results of PFNs in non-ranking settings have been achieved mostly in low-data settings. The primary bottleneck in deploying PFNs in data-rich settings is the quadratic complexity $O(n^2)$ of the transformer's self-attention mechanism with respect to the support set size. In

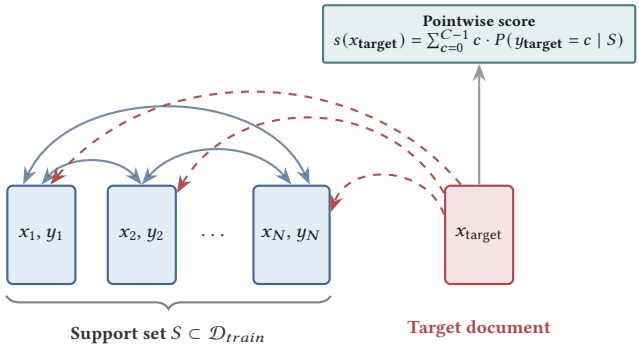

Figure 1: Overview of the pointwise ranking PFN architecture. The support set S (blue) provides context through self-attention (solid lines). The target document features x_{target} (red) queries this context via cross-attention (dashed lines) to calculate the pointwise score.

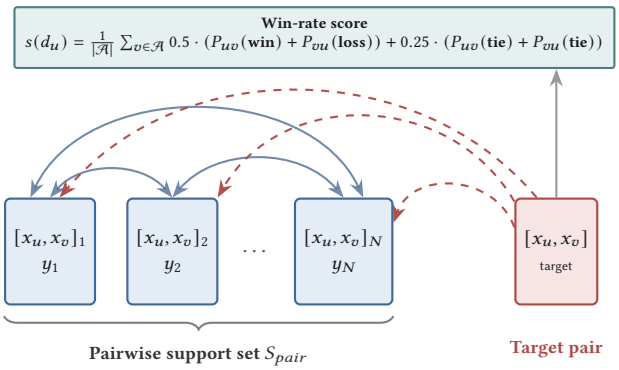

Figure 2: Overview of the pairwise ranking PFN architecture. The support set S (blue) provides context through self-attention (solid lines). The target pair features $[x_u, x_v]$ (red) attends to this context via cross-attention (dashed lines) to calculate the probability of a win, tie, or loss. The document score $s(d_u)$ is computed by aggregating probabilities over comparisons over anchor documents.

practice, in data-rich settings, this limits the support set size, requiring subsampling of training datasets, which may lead to performance limitations. Recent research has focused on several scaling strategies. (i) Methods such as LoCalPFN [48] bypass the global context limit by selecting a local subset of k-nearest neighbors (kNN) for each test sample. (ii) Rather than using raw samples, methods such as TuneTables [14] and ICD [30] optimize the context tokens themselves. (iii) New architectural developments, such as TabICL [39], propose embedding tabular feature vectors before processing them. We build on these methods by introducing a query-level dynamic sampling strategy for the ranking task, which selects the most relevant support set queries for each test query.

4 Methodology

We propose to use PFNs for ranking. Next, we describe how to employ PFNs as classifiers on ranking datasets and how they can be adapted to pointwise rankers. We introduce a novel approach to adapting PFNs for *pairwise* ranking inference. Finally, we propose a query-level dynamic support set selection approach for ranking in more data-rich settings, where the PFN's transformer context window can be a bottleneck.

4.1 Classification & pointwise ranking with PFNs

We start by approaching the ranking problem as a classification task, since this is the most straightforward way to apply PFNs, which are trained as classifiers to the ranking setting. Let $\mathcal{D}_{train}$ be the set of available training data containing query-document feature vectors (denoted by x_i) and their corresponding, graded relevance labels (denoted by y_i). To predict the score for a feature vector x_{target} of target document d_t, associated with a test query, we construct a support set $S \subseteq \mathcal{D}_{train}$. In low-data settings where $|\mathcal{D}_{train}|$ fits within the model's context window, we set $S = \mathcal{D}_{train}$. For larger datasets, we subsample $\mathcal{D}_{train}$ to obtain S. We discuss our strategy for the sampling procedure in Section 4.3.

To do classification on ranking datasets, we treat each query-document pair as an independent data point. The input to the PFN is a sequence of tokens representing support set document features, support set labels, and the target features:

$$\text{Input} = [x_1, y_1, x_2, y_2, \ldots, x_{n_s}, y_{n_s}, x_{\text{target}}], \tag{2}$$

where $(x_i, y_i) \in S$ are samples from the training set, and $n_s = |S|$ denotes the number of examples in the support set. The PFN applies in-context learning on S to predict a probability distribution over class labels for y_{target} based on the features x_{target}. In the LTR setting labels are usually graded relevance judgments, i.e., $y \in \{0, 1, 2, 3, 4\}$.

Applied as a classifier, a PFN can be used as a pointwise ranker, but for this, we need a transformation from the predicted label distribution to a scalar ranking score $s(x_{\text{target}})$. We choose the most straightforward approach by using the expected relevance value, which is computed as a weighted average over the possible label values weighted by the associated predicted probabilities:

$$s(x_{\text{target}}) = \sum_{c=0}^{C-1} c \cdot P(y = c \mid x_{\text{target}}, S), \tag{3}$$

where C is the total number of varying relevance labels (in LTR $C = 5$ is standard [5, 29, 36]). The predicted ranking for a query q is obtained by sorting all candidate documents $d \in D_q$ in descending order of their scores $s(x_d)$. We call this the *pointwise ranking PFN* approach and visualize it in Figure 1.

4.2 Pairwise ranking with PFNs

There is a seeming consensus in the LTR field that pointwise methods generally perform worse than their pairwise and listwise counterparts [4, 27, 33]. Accordingly, it appears to make sense that our pointwise ranking PFN approach could be outperformed by a pairwise or listwise equivalent. This section will introduce our *pairwise ranking PFN approach*, which uses the PFN to predict the ordering of pairs, these predictions are then aggregated into a ranking. Our pairwise approach is also visualized in Figure 2. We will not explore a listwise ranking approach for PFNs as this brings additional combinatorial challenges that are out of our scope.

4.2.1 Pairwise sampling strategy To obtain a support set S that aligns with the pairwise ranking task, S_{pair}, we first construct a pairwise dataset $\mathcal{D}_{\text{pair}}$. For a given query q, we sample pairs of documents $(d_u, d_v) \in (\mathcal{D}_q)^2, d_u \neq d_v$. We then assign a label y_{uv} to each pair, falling into one of three classes: $y_u > y_v$ (win), $y_u < y_v$

(loss), or $y_u = y_v$ (tie). This way, we predict whether a document is more relevant than the other, less relevant, or equally relevant. We find that explicitly modeling ties is important for PFNs, since not explicitly providing them leads to unstable performance.

We define a feature transformation function that combines the feature vectors of the two documents into a single representation suitable for the PFN. We use a concatenation-based encoding: $\phi(d_u, d_v) = \phi_{uv} = [x_u, x_v]$ where x_u and x_v are the feature vectors of the two documents. This ensures the model observes the full context of both documents. In preliminary experiments, we observed that concatenation performs better than subtracting or multiplying document features.

To ensure a diverse and balanced support set of pairs, we introduce a stratified sampling strategy; we categorize pairs into three groups: (i) *High-impact pairs* with a large relevance gap ($|y_u - y_v| > 1$). These are crucial for distinguishing very relevant documents from non-relevant ones. (ii) *Boundary pairs* with a minimal relevance difference ($|y_u - y_v| = 1$, labels are presumed to be integers). These fine-grained comparisons are important for resolving local ordering. (iii) *Tied pairs* with the same relevance labels for both documents ($y_u = y_v$). These help the model learn to predict ties or output equivalent scores for similar documents, regularizing the decision boundary. This sampling approach ensures the PFN learns to distinguish both subtle and large differences in relevance while maintaining stability for equally relevant documents.

4.2.2 Inference and aggregation During inference, the support set S_{pair} consists of document pair features and their corresponding labels. For a test query retrieving n documents, assessing all $n(n-1)$ pairs is computationally expensive and can exceed the context window of the transformer architecture underlying PFNs. To resolve this, we estimate the relevance score $s(d_u)$ of a document d_u by aggregating predicted probabilities from comparisons against a set of anchor documents, $\mathcal{A} \subset D_q$. We select $\mathcal{A}$ based on the documents with the highest BM25 scores, reducing the complexity from quadratic to linear relative to the total number of documents.

We employ a tie-aware win-rate aggregation scheme that leverages the capability of PFNs to produce well-calibrated probabilities for wins, losses, and ties. This approach explicitly accounts for the ordinal nature of relevance labels by treating ties as half-wins, providing a smoother ranking score in dense relevance spaces. The aggregated score $s(d_u)$ for a document d_u is defined as the expected win rate against all other documents in the candidate set $\mathcal{A}$:

$$s(d_u) = \frac{1}{|\mathcal{A} \setminus \{d_u\}|} \sum_{d_v \in \mathcal{A} \setminus \{d_u\}} \Big(0.5 \cdot P(y_{uv} = \text{win} \mid \phi_{uv}, S_{\text{pair}}) + 0.5 \cdot P(y_{vu} = \text{loss} \mid \phi_{vu}, S_{\text{pair}}) + 0.25 \cdot P(y_{uv} = \text{tie} \mid \phi_{uv}, S_{\text{pair}}) + 0.25 \cdot P(y_{vu} = \text{tie} \mid \phi_{vu}, S_{\text{pair}}) \Big) \quad (4)$$

where $P(y_{uv} = \text{win})$ is the predicted probability that d_u is more relevant than d_v, and $P(y_{uv} = \text{tie})$ is the predicted probability that they are equally relevant. To ensure robustness, we enforce symmetric consistency by averaging the predictions from both (d_u, d_v) and (d_v, d_u) directions, as the order of presentation could affect the predictions of PFN. This aggregation method sets the ranking scores to the (predicted) frequency that d_u would win from a document randomly sampled from $\mathcal{A}$, while making full use of the entire 3-class predicted probability distribution. Similarly, a combination of aggregation and the calibrated predictions of PFNs ensures that document pairs where the model expresses high uncertainty (probabilities near 0.5) contribute less to the final score than pairs where the model is certain. Overall, our pairwise strategy aims to bridge the gap between the PFN's pre-trained classification objective and the goal of a ranking task.

4.3 Dynamic support set selection

The pointwise and pairwise PFN ranking approaches described above are easily applicable in low-data settings. However, their application encounters an obstacle as the training dataset size increases, namely the context-window size. Since transformers suffer from quadratic memory scaling relative to the context length, it becomes infeasible to encode the entire training set. Consequently, we are forced to apply a subsampling strategy to maintain within memory and latency limits. A naive uniform random subsampling strategy would lead to the model having to learn from a very unbalanced subset of the training dataset, since in general, LTR datasets mostly consist of irrelevant documents ($y_d = 0$). Thus, a uniformly random subsample would be very uninformative as training data, leading to little gains from ICL and poor PFN predictions.

Luckily, PFNs can easily change their support set dynamically for each individual test query. In contrast with GBDT and neural networks, this also means they can subsample their training data to better match given inference features. We exploit this capability by introducing a dynamic query-sampling strategy. While recent work has explored dynamic support set selection for classification by retrieving relevant samples for each test point, we extend this methodology to the ranking task by shifting the selection granularity to the query level [48].

Rather than relying on a static global context shared across all queries, we provide the PFN with a local context tailored to the specific neighborhood of the target query. For each document or pair to be ranked, we construct the support set S by retrieving the nearest neighbor (kNN) queries from the training set. To represent queries, we compute embeddings by averaging the features of the top 20 documents of a query based on their BM25 scores. By then selecting all documents of the retrieved queries, we aim to provide the PFN with the most relevant features to adapt its prior to the current inference query for which a prediction is made.

5 Experimental Setup

In this section, we introduce the datasets, baselines, and experiments that we use to answer our research questions posed in subsection 1.2.

5.1 Datasets

The three LTR datasets we use in our experiments are MSLR-WEB30k [36], the Yahoo Webscope task set 1 [5], and the full Istella LTR [9] dataset. All three datasets come from the domain of web search and consist of queries with corresponding document sets. Documents have a graded relevance $y \in \{0, 1, 2, 3, 4\}$, which are annotations by human experts. Furthermore, documents have numerical features that can be query-dependent, such as BM25, or independent, such as a PageRank score. Statistics on these datasets are presented in Table 1.

Table 1: Statistics of the MSLR30k, Yahoo Set 1, and Istella LETOR datasets. % Pos. indicates the percentage of documents with a relevance label $y > 0$.

Dataset	Feat.	Doc/Q	% Pos.	Split	Queries	Docs
MSLR30k	136	120	48.5%	Train	18,919	2,270,296
				Val	6,306	747,218
				Test	6,306	753,611
Yahoo	519	24	73.9%	Train	19,944	473,134
				Val	2,994	71,083
				Test	6,983	165,660
Istella	220	316	3.7%	Train	18,576	5,862,053
				Val	4,643	1,463,572
				Test	9,799	3,129,004

5.1.1 Data pre-processing We pre-process all datasets by first removing features that are highly correlated with each other (Spearman correlation > 0.95). Preliminary experiments show that this increases performance across methods in low-data settings by reducing redundancy. Then, we apply a *log1p* transformation on the input features: $x = \log_e(1 + |x|) \odot \text{sign}(x)$. This is known to benefit neural rankers on the MSLR30k and Istella datasets [38]. The impact of normalization on Yahoo is limited, as it is already pre-processed.

5.1.2 Low-data settings To simulate low-data ranking settings, we subsample training and validation documents. We always use the entire test set to compute the reported metrics. As low-data settings in production environments are mostly caused by the limited number of labeled documents, we treat this as the limiting factor and define our training-data budget in terms of available documents per experiment. Given a budget (e.g., $N = 1{,}000$ documents), we first sample $\frac{N}{k}$ random queries, followed by sampling k documents per query until the budget is filled. These k documents are sampled based on the highest BM25 score per query, as would happen in a real-world setting, i.e., they are analogous to top-K retrieval candidates for a query [7, 13]. Since Istella does not have publicly available feature descriptions, we treat the feature with the highest label correlation in the training set as a proxy for BM25. We vary the labeling budget for low-data settings by $N \in [100, 500, 1000, 5000, 10{,}000, 25{,}000, 50{,}000]$, with 80% of queries for training and 20% for validation. We increase k as the labeling budget increases: $k \in [5, 5, 5, 10, 20, 20, 50]$, respectively, sorted by labeling budgets. These increasing values for k provide all methods with increasing query depth as more data becomes available, while ensuring query coverage in the lowest-data settings.

5.2 Experiments on ranking datasets

We use TabPFN [21] as our base model, due to its state-of-the-art performance across diverse tabular tasks [21, 23, 40], specifically utilizing TabPFN 2.6 [17], which features an Apache 2.0 architecture and non-commercial weights.[2] We prioritize TabPFN over alternatives like TabICL [**?**] for its superior classification accuracy, while both models can be used for full reproducibility for researchers. We ensured the integrity of our evaluation by verifying that TabPFN 2.6's real-data training corpus strictly excludes our ranking datasets, completely ruling out data leakage.

[2] https://huggingface.co/Prior-Labs/tabpfn_2_6

TabPFN has a context window limitation of 50,000 tokens. This means that for classification and pointwise ranking methods, we use all available data when $N \leq 50{,}000$. For pairwise ranking with PFNs, the number of training pairs very quickly exceeds the context window limitation of TabPFN, requiring us to subsample the number of training pairs. We construct the support set using the stratified sampling strategy defined in subsection 4.2, with sampling ratios of 40% high-impact pairs, 30% boundary pairs, and 30% ties. We set the number of sampled training pairs equal to the labeling budget (e.g., 5,000 pairs for a $N = 5{,}000$ budget), but enforce a minimum of 512 pairs for the smallest datasets (100 and 500 documents) to ensure training stability for all methods. During inference, we aggregate pairwise scores using 16 anchor documents, selected based on the highest BM25 scores, to balance performance and efficiency.

To answer RQ1, we perform a series of classification experiments. As classification baselines, we use XGBoost with a classification training objective, as well as an MLP with three hidden layers of sizes [256, 128, 64] with ReLU activations, batch normalization, and dropout ($p = 0.1$) [6]. The hyperparameters of all baselines were tuned with Optuna for 100 trials on the validation set based on a cross-entropy loss objective [1]. We define the Optuna hyperparameter search space separately for different amounts of available training data, as baselines in low-data settings can quickly overfit when not adapted properly. For exact tuning ranges, we refer to our public repository.[1] We employ early stopping to prevent overfitting. As we aim to compare the accuracy of PFNs as a classifier on LTR data, we report cross-entropy loss on the test set, since this is the exact objective that our baselines are optimized for.

To answer RQ2 and RQ3, we perform a series of ranking experiments. As ranking baselines, we use XGBoost with a classification objective, XGBoost with a regression objective, an XGBoost ranker with a pairwise objective, the LightGBM implementation of LambdaMART and the CatBoost implementation of YetiRank[4, 6, 19, 26, 35]. As a neural baseline, we use the DASALC algorithm as introduced by Qin et al. [38]. We also include a baseline based solely on the BM25 feature, or a BM25 proxy for Istella. In line with the pointwise setting, we tune baseline hyperparameters for 100 trials using Optuna on validation data based on the NDCG@10 objective, and define separate hyperparameter ranges based on data availability.

We report NDCG@10 on the full test set. All reported results are means over five trials repeated with different random seeds, where we also report standard deviation and perform significance testing with the Student's t-test [44].

5.3 Dynamic support set selection

To answer RQ4, we compare the performance of a pointwise PFN ranker with dynamic query-level support set selection against one with random support set selection. We use the full training sets for these experiments, representing a data-rich setting where the dataset size far exceeds the PFN's context window.

For the dynamic strategy, we retrieve all documents of the 32 nearest neighbors of the test query from the training set. We use the centroid-based representation described in section 4, using cosine similarity for retrieving the most similar queries. For the random strategy, we uniformly sample 32 queries from the training data. We report the mean NDCG@10 on the test sets of all three datasets.

Table 2: Classification performance of XGBoost, MLP, and TabPFN in terms of cross-entropy loss on the test sets of the MSLR, Yahoo, and Istella datasets for a varying number of subsampled training documents. * denotes a statistically significant improvement over the second-highest performing method/baseline ($p < 0.05$).

		Training documents						
Dataset	**Method**	**100**	**500**	**1k**	**5k**	**10k**	**25k**	**50k**
MSLR	XGBoost	1.3870 (0.0756)	1.2261 (0.0316)	1.1701 (0.0078)	1.0697 (0.0063)	1.0468 (0.0060)	1.0235 (0.0019)	1.0072 (0.0025)
	MLP	1.6275 (0.0672)	1.4337 (0.4663)	1.2248 (0.4960)	1.0515 (0.0169)	1.0298 (0.0057)	**1.0156** (0.0083)	**0.9955*** (0.0012)
	PFN	**1.2051*** (0.0743)	**1.0879*** (0.0219)	**1.0558*** (0.0098)	**1.0364*** (0.0055)	**1.0280** (0.0024)	1.0193 (0.0035)	1.0069 (0.0036)
Yahoo	XGBoost	1.5273 (0.0213)	1.3984 (0.0385)	1.3038 (0.0182)	1.1899 (0.0050)	1.1585 (0.0038)	1.1208 (0.0028)	1.0988 (0.0020)
	MLP	2.0509 (0.5128)	1.3428 (0.2948)	1.2799 (0.0194)	1.2602 (0.0320)	1.2246 (0.0243)	1.1526 (0.0189)	1.1212 (0.0068)
	PFN	**1.3537*** (0.0661)	**1.2209*** (0.0245)	**1.1723*** (0.0140)	**1.1164*** (0.0054)	**1.1029*** (0.0017)	**1.0833*** (0.0021)	**1.0647*** (0.0025)
Istella	XGBoost	0.4277 (0.0967)	0.2334 (0.0263)	0.1936 (0.0146)	0.1551 (0.0035)	0.1471 (0.0018)	0.1372 (0.0006)	0.1338 (0.0004)
	MLP	1.3189 (0.2711)	0.6721 (0.1363)	0.3028 (0.1949)	0.1726 (0.0172)	0.1655 (0.0766)	0.1400 (0.0062)	0.1342 (0.0039)
	PFN	**0.3314*** (0.0537)	**0.1926*** (0.0253)	**0.1654*** (0.0092)	**0.1375*** (0.0022)	**0.1311*** (0.0007)	**0.1243*** (0.0008)	**0.1218*** (0.0007)

6 Results

6.1 Classification with PFNs in low-data settings

We start by considering RQ1: *how do PFNs compare against baselines on a classification task on LTR datasets in low-data settings?*

Table 2 displays the cross-entropy loss for XGBoost, MLP, and PFN as the number of available training documents varies from 100 to 50,000. We observe that across the three datasets, PFN achieves the lowest loss of all methods up to 10k documents on MSLR30k, and up to 50k documents on the other datasets. In the highest-data settings (25k and 50k documents), PFN still obtains the lowest loss on Yahoo and Istella, but loses to the MLP on the MSLR30k dataset. On the Yahoo and Istella datasets, PFN performs significantly better than XGBoost and MLP across all numbers of training documents.

Given these results, we answer RQ1 positively: *In low-data settings, PFNs outperform GBDT and neural network-based classification baselines, only being overtaken by the MLP on MSLR30k at the 25,000 and 50,000 document marks.* These findings clearly signal the promising potential of PFNs on ranking datasets.

6.2 Pointwise ranking in low-data settings

We continue with RQ2 and assess *how pointwise PFNs compare against state-of-the-art ranking methods in low-data settings.*

In Figure 3 and Table 3, we see the ranking performance in NDCG@10 for state-of-the-art ranking baselines and our point- and pairwise PFN rankers as the number of available training documents ranges from 100 to 50,000. For RQ2, our focus is on comparing pointwise PFNs against strong point-, pair-, and listwise baselines, including XGBoost, LambdaMART, and YetiRank; thus, we ignore pairwise PFN's performance for now.

For MSLR30k and Yahoo, in settings with only 100 training documents, all methods struggle to outperform the BM25 baseline. From 500 documents on, pointwise PFN achieves the highest ranking performance in every low-data setting, with the strongest learned baseline consistently being the pointwise XGBoost classifier or regressor. We find that the XGBoost regressor has a slight edge over the classifier in the lowest-data settings on Yahoo. As data availability increases across all datasets, the classification baseline generally overtakes the regression baseline. These results align with earlier findings that pointwise objectives can sometimes generalize better than pair- or listwise objectives when data is scarce [41].

The magnitude of the pointwise PFN's advantage varies across the datasets. On Istella, the performance gains over the baselines are the most pronounced, with traditional methods struggling to compete across all settings up to 50k documents. On Yahoo and MSLR30k, the improvements are more gradual but remain consistent up to the 50k mark. This difference likely reflects the skewed label distribution of Istella, which makes it difficult for traditional baselines to generalize when trained from scratch on limited data. Notably, the pointwise PFN maintains its lead even as pair- and listwise baselines begin to close the performance gap as data increases.

Overall, these results demonstrate that pointwise PFNs provide a strong inductive bias for ranking with tabular features when labeled data is limited, outperforming carefully tuned GBDT and neural baselines across a wide range of realistic low-data scenarios. We therefore answer RQ2 by concluding that *pointwise PFNs outperform state-of-the-art ranking methods in low-data settings.*

6.3 Pairwise ranking in low-data settings

We move on to RQ3, which concerns *how pairwise PFN predictions compare against state-of-the-art ranking methods in low-data settings*, by analyzing the results in Figure 3 and Table 3 once more. This time, we focus on comparing the performance of our pairwise ranking PFNs, which adapt the PFN classifier to provide pairwise predictions of the relative ordering of document pairs and then aggregate these predictions to construct their predicted rankings.

When only 100 training documents are available, the pairwise PFN outperforms all learned baselines on MSLR30k and Istella, and even surpasses its pointwise counterpart on MSLR30k. However, in this extremely low-data regime, the pairwise model still struggles to surpass the unsupervised BM25 baseline on MSLR30k, and it also lags behind both the pointwise PFN and the pointwise XGBoost baselines on Yahoo.

As the amount of labeled data increases, the performance of the pairwise PFN consistently falls behind its pointwise counterpart across all three datasets. This relative degradation is the largest on Istella as the number of documents scales. As shown in Table 1, Istella is characterized by extreme label sparsity, with only 3.7%

Table 3: Mean and standard deviation of NDCG@10 over 5 trials of ranking models across three datasets for a varying number of training documents. * denotes statistically significant improvements over the second-highest means ($p < 0.05$). The ‡ for the Istella BM25 baseline clarifies that we use a proxy for the BM25 feature, as described in subsection 5.1.

Dataset	Method	Training documents						
		100	500	1k	5k	10k	25k	50k
MSLR30k	BM25	**0.2872*** (0.0000)	0.2872 (0.0000)	0.2872 (0.0000)	0.2872 (0.0000)	0.2872 (0.0000)	0.2872 (0.0000)	0.2872 (0.0000)
	XGBoost classification	0.2385 (0.0343)	0.3163 (0.0126)	0.3468 (0.0166)	0.3970 (0.0054)	0.4178 (0.0060)	0.4361 (0.0037)	0.4497 (0.0024)
	XGBoost regression	0.2260 (0.0615)	0.3188 (0.0175)	0.3290 (0.0207)	0.3992 (0.0068)	0.4162 (0.0057)	0.4341 (0.0030)	0.4454 (0.0022)
	XGBoost pairwise	0.2105 (0.0329)	0.2598 (0.0259)	0.2592 (0.0107)	0.3524 (0.0124)	0.3951 (0.0101)	0.4140 (0.0082)	0.4419 (0.0032)
	LambdaMART	0.2048 (0.0299)	0.2655 (0.0369)	0.2803 (0.0141)	0.3488 (0.0235)	0.3777 (0.0097)	0.4124 (0.0062)	0.4312 (0.0073)
	YetiRank	0.1985 (0.0652)	0.2956 (0.0339)	0.2898 (0.0269)	0.3756 (0.0108)	0.4053 (0.0076)	0.4324 (0.0048)	0.4496 (0.0047)
	DASALC	0.2198 (0.0395)	0.2590 (0.0207)	0.2883 (0.0195)	0.3940 (0.0156)	0.3964 (0.0082)	0.4266 (0.0055)	0.4422 (0.0050)
	PFN pointwise	0.2241 (0.0704)	**0.3262** (0.0273)	**0.3571** (0.0242)	**0.4070** (0.0124)	**0.4261*** (0.0043)	**0.4410*** (0.0041)	**0.4528*** (0.0022)
	PFN pairwise	0.2459 (0.0353)	0.3063 (0.0363)	0.3282 (0.0254)	0.3634 (0.0154)	0.4003 (0.0197)	0.4189 (0.0139)	0.4256 (0.0087)
Yahoo	BM25	**0.6984*** (0.0000)	0.6984 (0.0000)	0.6984 (0.0000)	0.6984 (0.0000)	0.6984 (0.0000)	0.6984 (0.0000)	0.6984 (0.0000)
	XGBoost classification	0.6467 (0.0159)	0.6876 (0.0048)	0.7007 (0.0040)	0.7217 (0.0009)	0.7285 (0.0017)	0.7365 (0.0014)	0.7394 (0.0017)
	XGBoost regression	0.6598 (0.0230)	0.6936 (0.0047)	0.7044 (0.0040)	0.7235 (0.0010)	0.7273 (0.0020)	0.7357 (0.0012)	0.7381 (0.0010)
	XGBoost pairwise	0.5947 (0.0391)	0.6472 (0.0137)	0.6719 (0.0160)	0.7007 (0.0058)	0.7097 (0.0024)	0.7246 (0.0017)	0.7337 (0.0014)
	LambdaMART	0.6029 (0.0383)	0.6499 (0.0148)	0.6673 (0.0083)	0.6967 (0.0067)	0.7034 (0.0089)	0.7194 (0.0032)	0.7300 (0.0029)
	YetiRank	0.6233 (0.0359)	0.6680 (0.0134)	0.6797 (0.0113)	0.7053 (0.0054)	0.7205 (0.0046)	0.7334 (0.0018)	0.7390 (0.0008)
	DASALC	0.6201 (0.0239)	0.6321 (0.0093)	0.6702 (0.0111)	0.6818 (0.0120)	0.7062 (0.0044)	0.7147 (0.0026)	0.7209 (0.0023)
	PFN pointwise	0.6660 (0.0274)	**0.7016** (0.0042)	**0.7140*** (0.0026)	**0.7306*** (0.0025)	**0.7376*** (0.0012)	**0.7443*** (0.0010)	**0.7486*** (0.0007)
	PFN pairwise	0.6317 (0.0467)	0.6825 (0.0187)	0.6890 (0.0126)	0.7175 (0.0070)	0.7214 (0.0055)	0.7276 (0.0022)	0.7316 (0.0014)
Istella	BM25‡	0.1781 (0.0000)	0.1781 (0.0000)	0.1781 (0.0000)	0.1781 (0.0000)	0.1781 (0.0000)	0.1781 (0.0000)	0.1781 (0.0000)
	XGBoost classification	0.3053 (0.0540)	0.3978 (0.0394)	0.4579 (0.0455)	0.5514 (0.0191)	0.5870 (0.0049)	0.6133 (0.0046)	0.6172 (0.0011)
	XGBoost regression	0.3087 (0.0572)	0.3902 (0.0446)	0.4222 (0.0466)	0.5429 (0.0210)	0.5631 (0.0075)	0.5974 (0.0032)	0.5891 (0.0037)
	XGBoost pairwise	0.2253 (0.0426)	0.3387 (0.0504)	0.4462 (0.0272)	0.4966 (0.0241)	0.5753 (0.0055)	0.5944 (0.0061)	0.6073 (0.0059)
	LambdaMART	0.2262 (0.0603)	0.3458 (0.0601)	0.4139 (0.0352)	0.5056 (0.0224)	0.5680 (0.0114)	0.5880 (0.0114)	0.6074 (0.0057)
	YetiRank	0.3123 (0.0508)	0.3657 (0.0515)	0.4391 (0.0471)	0.5525 (0.0176)	0.5922 (0.0087)	0.6163 (0.0031)	0.6267 (0.0051)
	DASALC	0.1848 (0.0965)	0.3804 (0.0300)	0.4290 (0.0364)	0.5231 (0.0138)	0.5329 (0.0395)	0.5874 (0.0111)	0.5923 (0.0110)
	PFN pointwise	**0.3272** (0.0742)	**0.4776** (0.0377)	**0.5052** (0.0434)	**0.5845*** (0.0155)	**0.6182*** (0.0063)	**0.6358*** (0.0052)	**0.6463*** (0.0046)
	PFN pairwise	0.3236 (0.0358)	0.4578 (0.0262)	0.4725 (0.0553)	0.5514 (0.0145)	0.5597 (0.0086)	0.5812 (0.0042)	0.5832 (0.0022)

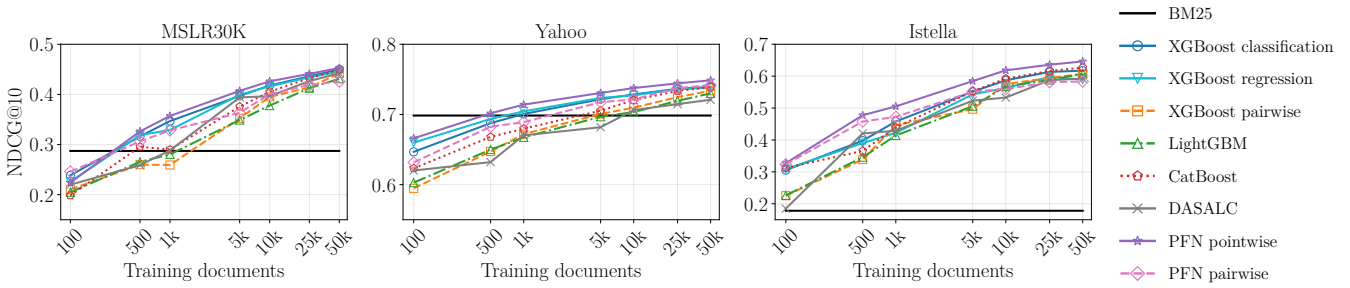

Figure 3: Mean NDCG@10 over 5 trials of ranking models across three datasets over a varying number of training documents.

of documents having a positive relevance label. While our pair sampling strategy is specifically designed to ensure a balanced support set, it appears insufficient to provide the pairwise PFN with a fully representative context.

Despite the scaling challenges, pairwise PFN remains highly competitive with traditional baseline methods in data-constrained scenarios. On Yahoo, we observe that the pairwise PFN outperforms all pair- and listwise baselines up to 10k documents. On MSLR30k and the highly sparse Istella dataset, it maintains this strict advantage over all pair- and listwise baselines up to 1k documents. Beyond these points, however, as data availability increases, strongly tuned baselines such as YetiRank and DASALC quickly close the performance gap and eventually surpass the pairwise PFN.

Table 4: Mean and standard deviation of NDCG@10 over three datasets. * denotes statistical significance improvements over the second-highest means ($p < 0.05$). Pointwise PFN with dynamic sampling using query-level kNN outperforms pointwise PFN with random data subsampling.

Method	MSLR30k	Yahoo	Istella
XGBoost class.	0.5055 (0.0006)	0.7589 (0.0001)	0.7325 (0.0001)
XGBoost pairwise	0.5036 (0.0014)	0.7536 (0.0023)	**0.7352*** (0.0001)
LambdaMART	**0.5069*** (0.0007)	**0.7644*** (0.0015)	0.7220 (0.0047)
YetiRank	0.4873 (0.0002)	0.7541 (0.0001)	0.7040 (0.0003)
DASALC	0.4623 (0.0012)	0.7382 (0.0007)	0.6202 (0.0027)
PFN (random)	0.4041 (0.0090)	0.7071 (0.0080)	0.6065 (0.0025)
PFN (kNN)	0.4676 (0.0263)	0.7488 (0.0157)	0.6747 (0.0075)

Therefore, we answer RQ3 by concluding that *while pairwise PFNs can achieve state-of-the-art results among pair- and listwise methods in low-data settings, their advantage generally diminishes as more labeled data becomes available.*

This conclusion contrasts with the general view of the LTR field that pairwise methods are expected to outperform pointwise methods, and suggests that there is still potential in future PFNs whose architecture and synthetic priors are specialized for pair- or listwise predictions. We hypothesize that in our current approach, the required pair subsampling limits the pairwise PFN's scalability, as its support set covers less of the query-document space, hindering ICL. Furthermore, concatenating feature vectors doubles the input dimensionality, further sparsifying the covered space.

6.4 Dynamic support set selection

Finally, we address RQ4: *whether dynamic support set selection can improve PFN ranking performance over random support set selection in data-rich settings.* We scale the training data for the pointwise PFN ranker to the full datasets and compare our random and dynamic query-level support-set subsampling strategies.

Table 4 displays the ranking performance (NDCG@10) for pointwise PFN with random or kNN subsampling strategies over the full dataset, together with the baselines trained on the full datasets (without subsampling). On MSLR30k and Yahoo, the LambdaMART baseline now achieves the highest performance, whereas on Istella, pairwise XGBoost performs best. For PFN with random subsampling, we observe considerably worse performance across datasets than the baselines. We conclude that, as expected, PFN performs very poorly when provided with a small random subset of training data, compared to baselines trained on full datasets.

In comparison, we see that dynamically retrieving documents for each inference query from similar training queries using kNN substantially increases performance over random subsampling. Table 4 reveals that, for the PFN with dynamic sampling, performance increases to competitive levels with DASALC on MSLR30k and Yahoo. On Istella, the kNN-based dynamic sampling PFN achieves the largest gain over random sampling. The gap from the dynamic PFN to the best-performing baseline, XGBoost pairwise, is the largest for Istella. We attribute this to the sparse label nature of Istella, requiring models to see an extensive number of documents and queries to have representative distributions.

In conclusion, we answer RQ4 affirmatively: *Using dynamic support-set selection for pointwise PFN ranking increases the performance over random subsampling.* However, performance is not at the level of state-of-the-art baselines, and future work should focus on scaling PFN-based rankers to large-data settings.

7 Discussion

Prior-data fitted networks learn from training data through in-context learning, requiring zero task-specific parameter optimization or hyperparameter tuning. This makes adaptation to real data extremely fast, requiring only a single forward pass. To illustrate, on MSLR30k with 1,000 training documents, a pointwise PFN produces its first prediction in 447 milliseconds on an H100 GPU. In contrast, tuning a LightGBM listwise ranker on the same hardware for 100 trials requires 2,100 seconds before any prediction can be made. PFN rankers are well-suited for environments where training bottlenecks must be minimized, but strict inference latency is less critical, such as deep research agents or rapid pipeline prototyping by data scientists.

For production environments with strict low-latency requirements, PFNs are not yet suitable. The model averages 80 milliseconds per prediction on the MSLR30k test set (using 1,000 training documents) on an H100 GPU, compared to 0.4 milliseconds for LightGBM. A fair and comprehensive latency comparison involves many interacting factors, such as hardware, measurement methodology (e.g., time-to-first-prediction vs. per-query inference latency), and tuning strategy, which falls beyond the scope of this work [15]. Future work should benchmark PFN architecture optimizations such as KV caching, sub-quadratic attention alternatives [55], and distillation of PFNs into smaller models [17], each offering distinct performance trade-offs.

8 Conclusion

In this paper, we proposed prior-data fitted networks as strong *foundational models for ranking in low-data settings.* While the existing LTR paradigm trains GBDTs or neural networks from scratch on labeled data, PFNs learn a synthetic prior and use in-context learning. Our experiments show that PFNs outperform baselines in classification performance on ranking datasets in low-data settings. We also find that pointwise PFN ranking achieves state-of-the-art performance in low-data settings. In data-rich settings, where GBDT-based methods excel, our dynamic support set selection enables the pointwise PFN ranker to improve performance over random subsampling. Based on these findings, we argue that PFNs for ranking are an important and potentially fruitful new research area for state-of-the-art ranking performance in low-data settings.

Our work focused on benchmarking and adapting the existing PFN formulation for ranking. While we found that our pointwise PFN approach already provides significant improvements in low-data settings, there appears to be a lot of potential left. In particular, our pairwise PFN approach did worse overall than the pointwise PFN, which contradicts the general trend in LTR. We think this means that PFNs could be more effective once an appropriate pair- or listwise approach is found for them. Accordingly, future work could explore (i) aligning the architecture of the transformer underlying PFNs to the ranking task, and (ii) developing ranking-specific

synthetic priors that would enable PFNs to perform even better in low-data settings and scale to data-rich settings.

Resources

At https://github.com/davidvos/pfns-for-ranking, we release the following under an Apache-2.0 license to support reproducibility:

- Implementations of our PFN-based pointwise and pairwise rankers.
- Implementations of the GBDT- and MLP-based baselines.
- Preprocessing scripts and dataloaders for MSLR30k, Yahoo LTRC, and Istella.
- Experiment scripts, Optuna search spaces, and configuration files to reproduce all reported results.

Acknowledgments

This research was (partially) supported by the Dutch Research Council (NWO), under project numbers 024.004.022, NWA.1389.20.-183, KICH3.LTP.20.006 and VI.Veni.222.269, and the European Union under grant agreement No. 101201510 (UNITE). Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of their respective employers, funders and/or granting authorities.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[2] Cristian Bodnar, Wessel P. Bruinsma, Ana Lucic, Megan Stanley, Anna Allen, Johannes Brandstetter, Patrick Garvan, Maik Riechert, Jonathan A Weyn, Haiyu Dong, et al. 2025. A Foundation Model for the Earth System. *Nature* (2025), 1–8.

[3] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. 2022. Deep Learning for Tabular Data: A Survey. *IEEE Transactions on Neural Networks and Learning Systems* 35, 1 (2022), 21–39.

[4] Christopher JC Burges. 2010. From Ranknet to LambdaRank to LambdaMART: An Overview. *Learning* 11, 23-581 (2010), 81.

[5] Olivier Chapelle and Yi Chang. 2011. Yahoo! Learning to Rank Challenge Overview. In *Proceedings of the Learning to Rank Challenge (Proceedings of Machine Learning Research, Vol. 14)*, Olivier Chapelle, Yi Chang, and Tie-Yan Liu (Eds.). PMLR, Haifa, Israel, 1–24. https://proceedings.mlr.press/v14/chapelle11a.html

[6] Tianqi Chen. 2016. XGBoost: A Scalable Tree Boosting System. *Cornell University* (2016).

[7] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, Jimmy Lin, Ellen M. Voorhees, and Ian Soboroff. 2022. Overview of the TREC 2022 Deep Learning Track. In *Proceedings of the Thirty-First Text REtrieval Conference, TREC 2022, online, November 15-19, 2022 (NIST Special Publication, Vol. 500-338)*, Ian Soboroff and Angela Ellis (Eds.). National Institute of Standards and Technology (NIST). https://doi.org/10.6028/NIST.SP.500-338.deep-overview

[8] W. Bruce Croft, Donald Metzler, Trevor Strohman, et al. 2010. *Search Engines: Information Retrieval in Practice*. Vol. 520. Addison-Wesley Reading.

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

[10] Domenico Dato, Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, and Nicola Tonellotto. 2022. The Istella22 Dataset: Bridging Traditional and Neural Learning to Rank Evaluation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Madrid, Spain) *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 3099–3107. https://doi.org/10.1145/3477495.3531740

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*. 4171–4186.

[12] Guglielmo Faggioli, Laura Dietz, Charles L. A. Clarke, Gianluca Demartini, Matthias Hagen, Claudia Hauff, Noriko Kando, Evangelos Kanoulas, Martin Potthast, Benno Stein, and Henning Wachsmuth. 2023. Perspectives on Large Language Models for Relevance Judgment. In *Proceedings of the 2023 ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR '23)*. 39–50. https://doi.org/10.1145/3578337.3605136

[13] Hui Fang, Tao Tao, and ChengXiang Zhai. 2004. A formal study of information retrieval heuristics. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*. 49–56.

[14] Benjamin Feuer, Robin Tibor Schirrmeister, Valeriia Cherepanova, Chinmay Hegde, Frank Hutter, Micah Goldblum, Niv Cohen, and Colin White. 2024. TuneTables: Context Optimization for Scalable Prior-Data Fitted Networks. *arXiv preprint arXiv:2402.11137* (2024).

[15] Ishaan Gangwani and Aayam Bansal. 2025. Light-Weight Benchmarks Reveal the Hidden Hardware Cost of Zero-Shot Tabular Foundation Models. *arXiv preprint arXiv:2512.00888* (2025).

[16] Yanjun Gao, Skatje Myers, Shan Chen, Dmitriy Dligach, Timothy A Miller, Danielle Bitterman, Matthew Churpek, and Majid Afshar. 2024. When Raw Data Prevails: Are Large Language Model Embeddings Effective in Numerical Data Representation for Medical Machine Learning Applications?. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, Miami, Florida, USA, 5414–5428. https://doi.org/10.18653/v1/2024.findings-emnlp.311

[17] Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Benjamin Jäger, Dominik Safaric, Simone Alessi, Adrian Hayler, et al. 2025. TabPFN-2.5: Advancing the State of the Art in Tabular Foundation Models. *arXiv preprint arXiv:2511.08667* (2025).

[18] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. 2022. Why Do Tree-based Models Still Outperform Deep Learning on Typical Tabular Data?. In *Advances in Neural Information Processing Systems*, Vol. 35. 507–520.

[19] Andrey Gulin, Igor Kuralenok, and Dimitry Pavlov. 2011. Winning The Transfer Learning Track of Yahoo!'s Learning To Rank Challenge with YetiRank. In *Proceedings of the Learning to Rank Challenge (Proceedings of Machine Learning Research, Vol. 14)*, Olivier Chapelle, Yi Chang, and Tie-Yan Liu (Eds.). PMLR, Haifa, Israel, 63–76. https://proceedings.mlr.press/v14/gulin11a.html

[20] Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, and David Sontag. 2023. TabLLM: Few-shot Classification of Tabular Data with Large Language Models. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 5549–5581.

[21] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. 2023. TabPFN: A Transformer that Solves Small Tabular Classification Problems in a Second. In *International Conference on Learning Representations (ICLR)*.

[22] Noah Hollmann, Samuel G. Müller, Katharina Eggensperger, and Frank Hutter. 2025. Accurate Predictions on Small Tabular Data with Tabular Foundation Models. *Nature* 637 (2025), 319–326.

[23] Sherrie Hoo, Noah Hollmann, Samuel G. Müller, and Frank Hutter. 2025. From Tables to Time: How TabPFN-v2 Outperforms Specialized Time Series Forecasting Models. *arXiv preprint arXiv:2501.02945* (2025).

[24] Rolf Jagerman, Xuanhui Wang, Honglei Zhuang, Zhen Qin, Michael Bendersky, and Marc Najork. 2022. Rax: Composable Learning-to-Rank Using JAX. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Washington DC, USA) *(KDD '22)*. Association for Computing Machinery, New York, NY, USA, 3051–3060. https://doi.org/10.1145/3534678.3539065

[25] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.* 20, 4 (Oct. 2002), 422–446. https://doi.org/10.1145/582415.582418

[26] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Advances in Neural Information Processing Systems* 30 (2017).

[27] Tie-Yan Liu. 2009. Learning to Rank for Information Retrieval. *Foundations and Trends in Information Retrieval* 3, 3 (June 2009), 225–331.

[28] Lars Lorch, Scott Sussex, Jonas Rothfuss, Andreas Krause, and Bernhard Schölkopf. 2022. Amortized Inference for Causal Structure Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 35.

[29] Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, and Salvatore Trani. 2016. 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* (Pisa, Italy) *(SIGIR '16)*. Association for Computing Machinery, New York, NY, USA, 949–952. https://doi.org/10.1145/2911451.2914763

[30] Junwei Ma, Valentin Thomas, Guangwei Yu, and Anthony Caterini. 2024. In-Context Data Distillation with TabPFN. *arXiv [cs.LG]* (Feb. 2024).

[31] Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. 2022. Transformers can do Bayesian inference. In *International Conference on Learning Representations (ICLR)*.

[32] Thomas Nagler. 2023. Statistical Foundations of Prior-data Fitted Networks. In *International Conference on Machine Learning (ICML)*. PMLR, 25660–25676.

[33] Harrie Oosterhuis. 2021. Computationally Efficient Optimization of Plackett-Luce Ranking Models for Relevance and Fairness. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 1023–1032. https://doi.org/10.1145/3404835.3462830

[34] Harrie Oosterhuis and Maarten de Rijke. 2018. Differentiable Unbiased Online Learning to Rank. In *Proceedings of the 27th ACM International Conference on*

Information and Knowledge Management (CIKM '18). ACM, 1293–1302. https://doi.org/10.1145/3269206.3271686

[35] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2018. CatBoost: Unbiased Boosting with Categorical Features. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 31. 6638–6648.

[36] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 datasets. *arXiv preprint arXiv:1306.2597* (2013).

[37] Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. 2024. Large Language Models are Effective Text Rankers with Pairwise Ranking Prompting. In *Findings of the Association for Computational Linguistics: NAACL 2024*, Kevin Duh, Helena Gomez, and Steven Bethard (Eds.). Association for Computational Linguistics, Mexico City, Mexico, 1504–1518. https://doi.org/10.18653/v1/2024.findings-naacl.97

[38] Zhen Qin, Le Yan, Honglei Zhuang, Yi Tay, Rama Kumar Pasumarthi, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2021. Are Neural Rankers Still Outperformed by Gradient Boosted Decision Trees?. In *International Conference on Learning Representations (ICLR)*.

[39] Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. 2025. TabICL: A Tabular Foundation Model for in-Context Learning on Large Data. *arXiv [cs.LG]* (Feb. 2025).

[40] Jake Robertson, Arik Reuter, Siyuan Guo, Noah Hollmann, Frank Hutter, and Bernhard Schölkopf. 2025. Do-PFN: In-Context Learning for Causal Effect Estimation. In *International Conference on Learning Representations (ICLR)*.

[41] David Sculley. 2010. Combined Regression and Ranking. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 979–988.

[42] Ravid Shwartz-Ziv and Amitai Armon. 2022. Tabular Data: Deep Learning is not All You Need. *Information Fusion* 81 (2022), 84–90.

[43] Ian Soboroff. 2025. Don't Use LLMs to Make Relevance Judgments. *Information Retrieval Research* 1, 1 (2025), 29–46. https://doi.org/10.54195/irrj.19625

[44] Student. 1908. The Probable Error of a Mean. *Biometrika* (1908), 1–25.

[45] Weiwei Sun, Lingyong Yan, Zheng Ma, Pengjie Ren, Zhumin Chen, and Zhaochun Ren. 2023. Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 14914–14937.

[46] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. https://openreview.net/forum?id=wCu6T5xFjeJ

[47] Paul Thomas, Seth Spielman, Nick Craswell, and Bhaskar Mitra. 2024. Large Language Models can Accurately Predict Searcher Preferences. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Washington DC, USA) *(SIGIR '24)*. Association for Computing Machinery, New York, NY, USA, 1930–1940. https://doi.org/10.1145/3626772.3657707

[48] Maksims Volkovs, Yves Raimond, David Steiner, and Et al. 2024. Retrieval & Fine-Tuning for In-Context Tabular Models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[49] Shuyi Wang, Bing Liu, Shengyao Zhuang, and Guido Zuccon. 2021. Effective and Privacy-preserving Federated Online Learning to Rank. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR '21)*. ACM, 3–12. https://doi.org/10.1145/3471158.3472236

[50] Xinru Wang, Hannah Kim, Sajjadur Rahman, Kushan Mitra, and Zhengjie Miao. 2024. Human-LLM Collaborative Annotation through Effective Verification of LLM Labels. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–21.

[51] Zifeng Wang and Jimeng Sun. 2022. TransTab: Learning Transferable Tabular Transformers Across Tables. In *Advances in Neural Information Processing Systems*, Vol. 35. Curran Associates, Inc., 2902–2915. https://proceedings.neurips.cc/paper_files/paper/2022/file/1377f76686d56439a2bd7a91859972f5-Paper-Conference.pdf

[52] Cornelius Wolff and Madelon Hulsebos. 2025. How Well Do LLMs Reason over Tabular Data, Really?. In *The 4th Table Representation Learning Workshop at ACL 2025*.

[53] Xuyang Wu, Ajit Puthenputhussery, Hongwei Shang, Changsung Kang, and Yi Fang. 2024. Meta-Learning to Rank for Sparsely Supervised Queries. *ACM Trans. Inf. Syst.* 43, 1, Article 14 (Nov. 2024). https://doi.org/10.1145/3698876

[54] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388* (2025).

[55] Yuchen Zeng, Tuan Dinh, Wonjun Kang, and Andreas C. Mueller. 2025. Tabflex: Scaling Tabular Learning to Millions with Linear Attention. *arXiv preprint arXiv:2506.05584* (2025).

[56] Zheyu Zhang, Tianping Zhang, and Jian Li. 2023. Unbiased Gradient Boosting Decision Tree with Unbiased Feature Importance. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI)*. 4629–4637. https://doi.org/10.24963/ijcai.2023/515