Meta-Learning to Rank for Sparsely Supervised Queries

XUYANG WU, Santa Clara University, Santa Clara, CA, USA
AJIT PUTHENPUTHUSSERY, HONGWEI SHANG, and CHANGSUNG KANG, Walmart Global Tech, Sunnyvale, CA, USA
YI FANG, Santa Clara University, Santa Clara, CA, USA

Supervisory signals are a critical resource for training learning to rank models. In many real-world search and retrieval scenarios, these signals may not be readily available or could be costly to obtain for some queries. The examples include domains where labeling requires professional expertise, applications with strong privacy constraints, and user engagement information that are too scarce. We refer to these scenarios as sparsely supervised queries which pose significant challenges to traditional learning to rank models. In this work, we address sparsely supervised queries by proposing a novel meta-learning to rank framework which leverages fast learning and adaption capability of meta-learning. The proposed approach accounts for the fact that different queries have different optimal parameters for their rankers, in contrast to traditional learning to rank models which only learn a global ranking model applied to all the queries. In consequence, the proposed method would yield significant advantages especially when new queries are of different characteristics with the training queries. Moreover, the proposed meta-learning to rank framework is generic and flexible. We conduct a set of comprehensive experiments on both public datasets and a real-world e-commerce dataset. The results demonstrate that the proposed meta-learning approach can significantly enhance the performance of learning to rank models with sparsely labeled queries.

CCS Concepts: • **Information systems** → **Learning to rank**;

Additional Key Words and Phrases: Meta Learning, Learning to Rank, Sparsely Supervised Queries

ACM Reference format:
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 (November 2024), 29 pages.
https://doi.org/10.1145/3698876

1 Introduction

Learning to Rank (LTR), which refers to machine learning techniques on automatically constructing a model from data for ranking in search, has been widely used in modern search engines [44]. Typically, LTR involves creating a single ranking function that applies universally to all queries to order items based on their relevance. The global ranking model is generally efficient

This work was completed as part of a summer internship program at Walmart Global Tech.
Authors' Contact Information: Xuyang Wu, Santa Clara University, Santa Clara, CA, USA; e-mail: xwu5@scu.edu; Ajit Puthenputhussery, Walmart Global Tech, Sunnyvale, CA, USA; e-mail: ajit.puthenputhussery@walmart.com; Hongwei Shang, Walmart Global Tech, Sunnyvale, CA, USA; e-mail: hongwei.shang@walmart.com; Changsung Kang, Walmart Global Tech, Sunnyvale, CA, USA; e-mail: changsung.kang@walmart.com; Yi Fang (corresponding author), Santa Clara University, Santa Clara, CA, USA; e-mail: yfang@scu.edu.

ACM 1558-2868/2024/11-ART14
https://doi.org/10.1145/3698876

and scalable since it can be reused without requiring separate training or tuning for each query. Such an approach often delivers robust average performance and is easier to maintain in practice, making it widely adopted in LTR. However, the global ranking approach may be sub-optimal for individual queries as it tends to overlook query specificity and user intent. This is particularly problematic given that relevant documents for different queries can have varying distributions in the feature space, which a global ranking function might not adequately capture [2]. For instance, considering two ranking features such as word matching and freshness, in queries like "running shoes for flat feet," emphasis may be placed on word matching over freshness, whereas queries like "latest video games" would prioritize freshness. This variation necessitates the development of query-specific rankers, as global models may not able to generalize across diverse queries. Different queries prioritize different features, leading to domain shifts that can undermine the effectiveness of models trained on different types of data. Query-specific models are desired in certain search scenarios where the characteristics of queries and user intents may lead to distinct distributions of relevant documents in the feature space, and offer the advantage of tailoring model parameters to optimize retrieval for individual queries. The prior works in the literature [2, 11, 27] have also advocated for constructing ranking functions on a per-query basis, recognizing the limitations of a global ranking function.

Moreover, learning an effective ranking function often relies on the availability of a large amount of labeled examples. It may be difficult to obtain sufficient labeled examples for many queries in the real world such as domains where labeling requires professional expertise (e.g., biomedical and legal search) and applications with strong privacy constraints [68] (e.g., personal and enterprise search). User engagement data such as clicks/add-to-cart/purchase on e-commerce platforms are indicative of individual users' relevance judgments and are relatively easy to collect with low cost, but queries with sparse user interaction are still frequently encountered on these platforms such as queries for new products and tail queries. Additionally, due to certain biases in data collection and the limited availability of labeled data, user interactions labels may not necessarily align with actual user preferences [63]. We refer to the above scenarios where queries have limited supervisory signals for LTR as sparsely supervised queries.

Sparsely supervised queries pose significant challenges to LTR models, especially when learning query-specific rankers. First of all, traditional LTR methods typically require a large amount of supervised data to optimize different ranking objectives, but this design is not intended to learn "fast" from limited data. Although some recent works [34, 80] have attempted to dynamically adjust the ranker's optimization direction using online LTR with historical data and current real-time data, these approaches often suffer from insufficient optimization efficiency, unmeasurable performance, or performance that is inferior to offline approaches [55]. Moreover, even if an LTR model is trained with a large amount of supervisory signals, when it encounters sparsely supervised queries at runtime, it may not be able to generalize well. The scarcity or limited number of examples can have a significant impact on inductive bias [7]. The characteristics of sparsely supervised queries could be quite different from those of the training queries, which may lead to the domain shift problem from training to prediction/inference. In addition, sparsely supervised queries usually result in a high imbalance between positive labels and negative labels since irrelevant documents can often be sampled from the dataset while relevant documents have to be labeled. There exist some works in the literature that attempted to address the above respective challenges by generating synthetic data or duplicating existing data to provide more informative training sets. For example, data augmentation [60, 78], resampling methods [6, 15], and ensemble methods [23] were utilized to alleviate data sparsity, balance relevance labels, and attempt to learn an unbiased model in training. These methods have limited improvement in terms of model generalization due to insufficient data and domain shift issues. And the small sample size and uneven distribution of labels can result in

bias or difficulties in transferring knowledge, as well as slow adaptation to new queries, ultimately limiting generalization. On the other hand, some works aim to mitigate the impact of noise and bias through unbiased modeling perspectives and model adjustments [1, 17, 40, 41, 51, 53, 69, 79]. Generally, these methods model position bias by requiring extensive click logs. For instance, to optimize position bias in the data, the concept of counterfactual **Inverse Propensity Scoring (IPS)** was introduced in [1]. In our research scenario, there is a lack of positive sample data, which increases the difficulty of modeling bias. Additionally, due to insufficient training samples, minor propensities, and a large number of noisy clicks, counterfactual LTR systems frequently suffer from excessive variance. Oosterhuis [51] proposed the DR estimator, which provides enormous decreases in variance. These models have achieved remarkable success in the unbiased LTR field by using more efficient estimators to correct the bias problem. Last but not least, the presence of sparsely supervised queries complicates the development of query-specific rankers, as the straightforward approach of training individual models for each query would only exacerbate data sparsity and render the process infeasible.

Given these challenges, we turn to meta-learning [25], which has demonstrated its great success in the setting of few-shot learning where a model can quickly adapt to a new task using only a few data points and training iterations, as shown in a wide range of machine learning applications including image classification, dialog generation [57], text classification [38], and recommendation systems [18, 36, 43]. LTR for sparsely supervised queries shares similar characteristics with meta-learning in a few-shot setting because it focuses on ranking items for a query which only has a small number of labeled documents or supervisory signals. Inspired by the capabilities of meta-learning in fast learning and improving model generalization, we propose a novel **Meta-Learning to Rank (MLTR)** approach to address sparsely supervised queries. In scenarios where labeled data are scarce and the distribution of labels is imbalanced, meta-learning can effectively utilize its efficient learning and adaptability capabilities. Moreover, meta-learning can mitigate the impact of domain shift by allowing models to quickly adapt to different data distributions through task-specific training during the learning process and fine-tuning during inference.

In this article, we utilize the optimization-based meta-learning approach [25] to rapidly estimate document relevance for a new query based on only a small number of labeled documents. For each query in the meta-training process, there are two sets: training set and test set. The proposed MLTR model performs local and global updates. During the local update, the algorithm adjusts the parameter of the query-specific ranker on each training set (learning process). During the global update, the algorithm trains the parameter of the meta-ranker to minimize the meta-loss with the adapted parameters on the test sets (learning-to-learn process). Each query-specific ranker only requires few labeled instances for fine-tuning as the meta-ranker is trained across all the queries and the global ranking knowledge is transferred to each query-specific ranker as initial model parameters before fine-tuning. The proposed meta-learning approach is an efficient way to learn from limited data. To estimate document relevance for a new query, the ranker can then be fine-tuned based on the limited amount of labeled documents. Due to the learning-to-learn process, the model is able to quickly adapt to a new query. Query-specific rankers enable the model to capture and adapt to the unique characteristics of each query, while the meta (global)-ranker preserves scalability and efficiency across diverse queries. By leveraging the strengths of both approaches, our method aims to balance scalability with specificity, ensuring robust performance and leading to more precise and relevant results. In consequence, the proposed method leverages the fast learning and adaptation capabilities inherent in the meta-learning framework, yielding significant advantages especially when new queries are of different characteristics with the training queries.

Long-tailed queries can be naturally tackled by the proposed meta-learning approach. A portion of these queries may only appear once, while others could appear multiple times, albeit less than a few. Our proposed meta-learning approach can handle both scenarios through without fine-tuning or with fine-tuning. For queries that appear only once, we may not use data from the same query in training. For queries that appear more frequently, we can apply fine-tuning on unseen queries. The experiments demonstrate performance improvements of the MLTR models over the baselines under both scenarios. It is worth noting that the proposed approach is not limited to long-tailed queries. Even with more frequently occurring queries, such as torso and head queries, the available labels or user engagement data could be quite scarce, especially within a short timeframe since their first appearance. To quickly learn good ranking functions for these queries is crucial for engaging users in real-world search applications. Our work is centered on fast and efficient learning from sparse labels, a focus we believe holds broad applicability across various search scenarios. The main contributions of the article can be summarized as follows:

—We propose a novel meta-learning framework for search and ranking with sparsely supervised queries. To the best of our knowledge, there is no prior work on adopting the optimization-based meta-learning for LTR.
—The proposed MLTR model can leverage its strong generalization ability during training, enabling it to sustain consistently stable performance in ranking tasks involving unseen queries.
—The proposed MLTR model can quickly adapt to a new query with limited supervisory signals and can yield query-specific rankers with optimal model parameters for individual queries.
—The proposed approach is generic and flexible and can be applied to any existing LTR models to improve model generalization.
—We conduct a comprehensive set of experiments on four public LTR benchmarks and one real-world product search dataset. The results demonstrate the effectiveness of the proposed approach over the competitive baselines.

2 Related Work

2.1 LTR

LTR has gained much attention from the **Information Retrieval (IR)** research and industry community, which aims to optimize the perfect search results. A series of LTR models have been developed in the research community, mainly in two divisions. One division is based on Gradient Boosted Decision Trees, which have been used by many production search systems [77]; the other division is based on neural networks. With the intriguing interest in neural LTR models, a lot of papers have been published, and researchers keep proposing state-of-the-art methods [19, 21, 22, 35, 59, 74]. The DSSM model, cited as [35], belongs to the category of representation-based models. It operates by calculating the embeddings of a query and a document, which involves averaging the word embeddings from their respective text fields. On the other hand, the interaction BERT-based model, referenced as [22], employs a different approach. It concatenates the query and document text fields into a single sentence, which is then fed through multiple transformer layers. This approach has proven to yield state-of-the-art results, as cited in [19]. In recent few years, neural LTR models gradually start to be launched on various commercial search engines successfully, such as LinkedIn [29] and Taobao [76]. Han et al. [32] introduced TFR-BERT, a generic document ranking framework that combines the power of LTR models and BERT. Wu et al. [73] proposed a multi-task learning approach for the ranking problem to optimize the different engagement signals simultaneously. Furthermore, some LTR models [4, 42, 46, 75] used effective negative sampling technique to improve the efficiency of model training process and effectiveness of the resulting

model by filtering out noise and reducing the redundancy of the query–document pairs. Lucchese et al. [46] demonstrated the proportion of relevant documents to non-relevant documents highly affects the quality of the learning-to-rank collections. Kanoulas et al. [42] illustrated the relevance grade distribution in the training set is an important factor for the effectiveness of LTR algorithms. While Neu-IR and traditional LTR models have shown promising results on large, well-labeled datasets, their performance decreases when faced with sparsely labeled data. Conversely, the global ranking function in LTR may not be optimal for document retrieval since it ignores differences between feature distributions for each query [2]. Although Can et al. [11] and Dehghani et al. [21] constructed a ranking function specific to each query, this approach is limited by its high cost and low generalization ability due to the almost infinite number of queries and the unpredictable feature distribution of unseen queries. Our proposed MLTR model, however, can effectively improve the generalization of the LTR model on sparse datasets.

While traditional LTR models rely on manually annotated labels for supervision, click-based LTR models harness user interaction logs as a guiding resource [39]. Initially, these models were grounded in an online dueling-bandit framework [64, 80], evolving later to an online pairwise method [52] for unbiased pairwise optimization in LTR. However, that click data are not entirely reliable indicators of user preference due to various influencing factors beyond just preference [40, 61]. Addressing this, Yuan et al. [79] and later Joachims et al. [41] developed the first theoretically unbiased LTR method, based on the premise that a user's likelihood of examining an item is tied to its rank position, with clicks primarily on items that are examined [17, 69]. They employed counterfactual IPS estimation [62] to adjust for the biases inherent in these examination probabilities. IPS-based approaches, combating biases like position bias [40], trust bias [1], and selection bias [53], are prevalent in IR. Oosterhuis and de Rijke [54] enhanced these methods, accounting for potential changes in the logging policy during data collection. Despite their widespread use, IPS methods do grapple with high variance issues [30]. This opens avenues for future research into alternative bias mitigation techniques, such as the doubly robust method [51]. Sample-based approximation [49, 67] is typically employed in LTR scenarios where relevance is clearly established. Notably, Oosterhuis [49] recently introduced the **Plackett-Luce (PL)**-Rank method, an efficient approach for unbiased gradient estimation based on sampled rankings. Click-based LTR models aim to derive unbiased ranking models from inherently biased user behavior data, demonstrating efficiency in unbiased LTR area. However, our research doesn't primarily focus on the bias in data collection or usage. Instead, we concentrate on the application of meta-learning techniques to train robust LTR-based models effectively, particularly in scenarios characterized by sparse datasets.

On the other hand, some works implement online LTR methods. For instance, Yue and Joachims [80] introduced the first online LTR method, which utilizes online evaluation by sampling model variants and comparing them with interleaving to identify better rankers, thereby improving the entire system. Hofmann et al. [34] extended this by guiding exploration through reusing previous interactions. However, these approaches are not always efficient and sometimes their performance at convergence is much worse than offline approaches, especially in online settings where performance cannot be measured, making early stopping unfeasible [64]. Addressing these issues, our MLTR model can ensure more efficient optimization of traditional ranking model and keep stable model generalization with low computational cost in sparse data experimental settings.

2.2 Sparsely Supervised Learning

Supervised learning algorithms have faced a challenge in handling sparse and imbalanced labeled data. Previous techniques have typically addressed the negative impact of sparse data and imbalanced data distribution by optimizing the model or through data augmentation methods. From a model perspective, Zhou et al. [82] proposed that existing methods for dealing with the challenge

of few labeled examples often rely on semi-supervised learning techniques that exploit both labeled and unlabeled data. Moreover, Sun and Hardoon [65] introduced an active learning strategy for identifying informative examples that require manual labeling, which is particularly beneficial when manual labeling resources are limited. Nonetheless, it is crucial to note that the inductive bias of a model can be significantly impacted by having a limited number of examples, commonly referred to as sparse data, as noted in [7]. From data augmentation perspective, resampling is a typical technique for handling data imbalance in machine learning [28]. Data oversampling was introduced by Chawla [14], who sampled the minor classes from the available data and included them in the training process to mitigate the imbalance between major and minor classes. One of the popular oversampling techniques is SMOTE [15], which has various adaptations such as those proposed by [31, 33], and others. However, the learned supervised model has limit improvement with duplicated data without new information and high risk of over-fitting. In the same way, Liu et al. [45] employed data under-sampling as a technique to achieve a comparable amount of training data in various classes by reducing the number of data in the major classes. The article referenced as [75] reports on the use of a two-tower neural model that was trained utilizing a mixed negative sampling technique alongside batch random negatives. However, this method may lead to a loss of information during the reduction of training data through sampling. Data generation models for informative data augmentation in LTR are proposed by Yu and Lam [78] and Qiu et al. [60], as they believe that generating informative data is more beneficial than using resampling techniques. Those models generated informative synthetic data based on Adversarial Auto-Encoder [48] and **Gaussian Mixture Variational Auto-Encoder (GMVAE)** [24], respectively. Given the strong text generation capabilities of **Large Language Models (LLMs)**, many researchers [8, 20, 56] propose using LLM-driven methods to generate pseudo-queries or relevance labels from existing collections. Both of them could generate new data given different query types and different relevance levels. Resampling methods and data augmentation techniques have the potential to mitigate the effects of imbalanced data in the training set; however, they have limited improvement on overall model generalization.

2.3 Meta-Learning for IR

Meta-learning is also known as learning to learn, which aims to learn better algorithms, including better parameter initialization, optimization strategy [3], network architecture [83], and distance metrics [26]. Finn et al. [25] proposed a **Model-Agnostic Meta-Learning (MAML)** algorithm, which trains a model on a variety of tasks, such that the model can be easily generalized to a new task with a small number of gradient steps from a small number of data from that task. Also, a lot of existing works have implemented the meta-learning approach in other research areas. Lee et al. [43] proposed **Meta-Learned User Preference Estimator (MeLU)**, which utilizes meta-learning approach to deal with the cold start problem in the recommendation system. Cui et al. [18] proposed a novel approach to address the challenge of data sparsity in next Point-of-Interest recommendation, called Meta-SKR, which leverages a meta-learning approach to generate user-conditioned parameters for a sequential-knowledge-aware embedding module. Bansal et al. [5] proposed a MAML-based meta-learning method LEOPARD for domain adaptation tasks in Natural Language Processing. In addition, there are some works on IR. Carvalho et al. [13] proposed a meta-learning algorithm to suppress the undesirable outlier effects of the pairs of documents using the pairwise ranking function. Zabashta et al. [81] presented a meta-learning model for selecting rank aggregation algorithms based on a specific optimality criterion. Wu et al. [72] introduced a novel **Bayesian Online Meta-Learning Model (BOML)** tailored for personalized product search. BOML harnesses meta-knowledge acquired from inferences made about other users' preferences, enabling accurate predictions even in situations where historical data are limited. By addressing

the challenge of data sparsity, BOML can significantly enhance the accuracy of recommendations in personalized product search. Wang et al. [71] proposed Meta-learning based Fair Ranking, which alleviates the data bias and achieves better fairness metrics in the ranking model through an automatically weighted loss. Sun et al. [66] proposed the MetaAdaptRank, which is a domain adaptive learning method for few-shot Neu-IR based on meta-reweighted weak supervision data selection during the different periods of the training process. However, to the best of our knowledge, there have been no work using meta-learning on ranking models with the sparsely labeled queries.

3 The Framework

Our proposed MLTR framework is presented in this section. First, we will explain the traditional LTR model and the meta-based LTR model, which sets the problem context. Then, we will detail the MLTR's training and testing processes, which enables fast adaptation and improve model generalization.

3.1 LTR

Let $Q = \{q_1, q_2, ..., q_N\}$ denote the collection of N queries, $D = \{d_1, d_2, ..., d_M\}$ denote the collection of M documents, and $L = \{1, 2, ..., l\}$ denote the collection of l labels. There is an order of labels $l > l-1 > ... > 1$, where $>$ denote the sequence of the label order.

For every query q_i, there is a corresponding related document collection $D_i = \{d_{i,1}, d_{i,2}, ..., d_{i,J}\}$ and the corresponding label collection $y_i = \{y_{i,1}, y_{i,2}, ..., y_{i,J}\}$. Above all, the original data $\mathcal{S}$ can be denoted as $\mathcal{S} = \{(q_i, D_i), y_i\}_{i=1}^{N}$. The object is to train the ranking model of a given query q_i and corresponding related document collection D_i with the ranking label y_i, mathematically as $\hat{y}_i = f(\mathbf{x}_i; \theta)$ where $f(\cdot)$ is a ranking function, θ represent all the learnable parameters in $f(\cdot)$, and $\mathbf{x}_i$ denote the concatenated feature vector generated from the query and documents (q_i, D_i). $\mathbf{x}_i = concat(\phi(q_i), \psi(D_i), \mathbf{r}_i)$, where $\phi(\cdot)$ and $\psi(\cdot)$ denote the query encoder and the document encoder, respectively; $\mathbf{r}_i$ denotes the numeric ranking features for each query and corresponding related document collection (q_i, D_i). Generally, we learn the optimized θ^* by $min_\theta \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\hat{y}_i, y_i)$ and $\mathcal{L}$ could be used as any ranking loss functions.

3.2 Problem Formulation

Our work is inspired by optimization-based meta-learning, specifically MAML [25], which optimizes globally shared parameters over several tasks, so as to rapidly adapt to a new task with just one or a few gradient steps based on a small number of examples.

In the search and ranking setting, we define each task as ranking items for a given query. Our MLTR framework trains a model with a good generalization which can quickly adapt to a new query based on the query's sparse engagement information. We divide the raw data into $\mathcal{S}$ and $\mathcal{T}$. We limit each query task (including the query and all its corresponding items) within only one set, such that there is no query overlap between $\mathcal{S}$ and $\mathcal{T}$.

For each task query q_i ($\in Q$) in $\mathcal{S}$, its corresponding items are randomly divided into a training set $\mathcal{S}_{train,i}$ and a test set $\mathcal{S}_{test,i}$ to optimize the model during various stages. For each task query q_i ($\in Q$) in $\mathcal{T}$, its corresponding items are randomly split into a fine-tuning set $\mathcal{S}_{tuning,i}$ and an evaluation set $\mathcal{S}_{eval,i}$ to fine-tune the model and assess its performance, respectively. For further information regarding the notation employed in this article, please refer to Table 1.

3.3 MLTR

The MLTR framework's key concept is to create robust model parameters through many query-based ranking tasks in meta-training, then quickly adapt these parameters for new tasks in meta-testing with a few gradient steps. In meta-training, it performs local and global updates. The local update

Table 1. Summary of Notation

Notation	Definition
q, d	Query, document
$\mathcal{S}$	Set of datasets with queries for meta-training
$\mathcal{S}_{train}$	Set of sampled training data for meta-training $\mathcal{S}$ for query-specific ranker
$\mathcal{S}_{test}$	Set of sampled test data from meta-training $\mathcal{S}$ for meta-ranker
$\mathcal{S}_{train:p\cdot n\cdot}$	Set of fixed positively and negatively labeled items assigned to each query in the training dataset of $\mathcal{S}_{train}$
$\mathcal{S}_{test:p\cdot n\cdot}$	Set of fixed positively and negatively labeled items assigned to each query in the test dataset of $\mathcal{S}_{test}$
$\mathcal{S}_{train,i}$	Training data for query i in $\mathcal{S}_{train}$
$\mathcal{S}_{test,i}$	Test data for query i in $\mathcal{S}_{test}$
$\mathcal{T}$	Set of datasets with unseen queries for fine-tuning and evaluation
$\mathcal{T}_{tuning}$	Set of sampled data from $\mathcal{T}$ for fine-tuning
$\mathcal{T}_{eval}$	Set of sampled data from $\mathcal{T}$ for evaluation
$\mathcal{T}_{tuning:p\cdot n\cdot}$	Set of fixed positively and negatively labeled items assigned to each query in the test data $\mathcal{T}$ for fine-tuning
$\mathcal{T}_{eval:rest}$	Remaining test dataset $\mathcal{T}$ for evaluation
$\mathcal{T}_{tuning,i}$	Fine-tuning data for query i in $\mathcal{T}_{tuning}$
$\mathcal{T}_{eval,i}$	Evaluation data for query i in $\mathcal{T}_{eval}$
$g(\cdot,\theta)$	Meta-ranker with global parameter θ
$f(\cdot,\theta_i)$	Query-specific ranker with parameter θ_i for query i
$\mathcal{L}_{query}(\theta_i)$	Loss function of query-specific ranker
$\mathcal{L}_{meta}(\theta)$	Loss function of meta-ranker

adjusts the parameter of the query-specific ranker on each training set (learning process). The global update trains the parameter of the global ranker to minimize the meta-losses with the adapted parameters on the test sets (learning-to-learn process). The proposed meta-learning approach to ranking considers that individual queries may have distinct optimal parameters for their rankers, which is unlike traditional LTR models that learn a global ranking model applicable to all queries. Figure 1 illustrates the architecture of the proposed meta-training process. To estimate document relevance for a new query in meta-testing, the ranker can then be fine-tuned based on the limited amount of labeled documents. The following subsections provide the details of the proposed approach.

3.3.1 Meta-Training. We define the meta-ranker and query-specific ranker in our model as well, similar as the MAML [25] setting. The ranker model can be defined with any model structure based on your tasks, such as the basic **Multi-Layer Perceptron (MLP)**. Query-specific ranker $f(\cdot;\theta_i)$ is initialized by the meta-ranker and learns the task-specific parameters θ_i to optimize a specific task at a time. Meta-ranker $g(\cdot;\theta)$ learns across multiple tasks based on the query-specific ranker and can improve the model generalization performance. Although these two rankers share the same network structure and parameters θ_i, θ, respectively, their loss function objectives are different. Thus, the meta-learning for sparsely supervised search could be defined as, for the meta-training dataset $\mathcal{S} = \mathcal{S}_{train} \cup \mathcal{S}_{test}$, the meta-train process aims to train the query-specific ranker $f(\cdot;\theta_i)$ to learn task-specific parameters θ_i on $\mathcal{S}_{train}$, and to train the meta-ranker $g(\cdot;\theta)$ cross multiple tasks on $\mathcal{S}_{test}$ to extend the model generalization. Note that training and test sets are split at the

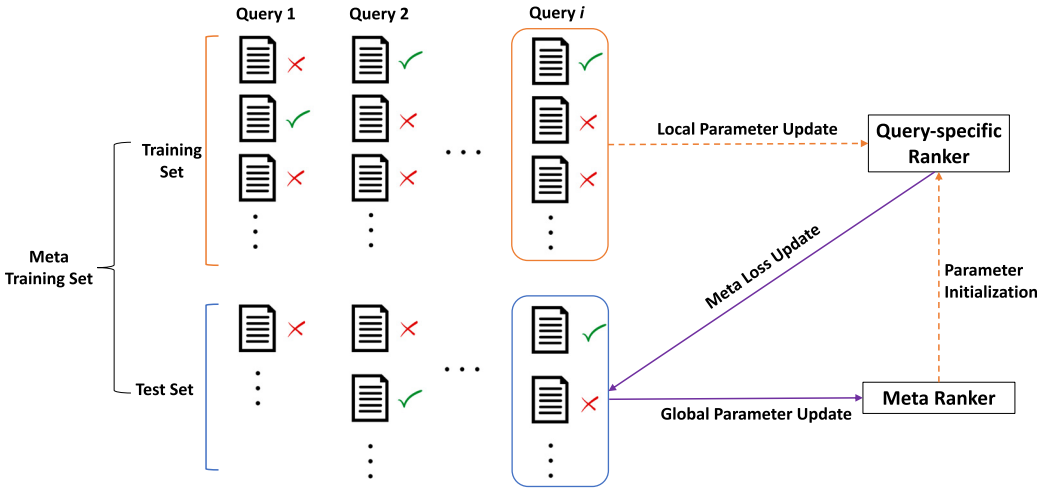


Fig. 1. The architecture of the proposed MLTR in the meta-training process.

query level within meta-training process. As introduced earlier in this section, each query q_i in the meta-train data has a corresponding training set $\mathcal{S}_{train,i}$ ($\subset \mathcal{S}_{train}$) and $\mathcal{S}_{test,i}$ ($\subset \mathcal{S}_{test}$).

The model training consists of a basic specific-task learning process and cross-task meta-adaptation process, trained on training set $\mathcal{S}_{train}$ and test set $\mathcal{S}_{test}$, respectively. Note that there is no intersection between $\mathcal{S}_{train}$ and $\mathcal{S}_{test}$. For the basic specific-task learning process (local parameter updates with training set), the query-specific ranker focuses on the quick acquisition of knowledge to learn task-specific parameters through the LTR loss. LTR loss indicates how well the model is performing on the specific task (query). For the meta-cross-task adaptation process (global updates with test set), the model further learns generalized parameters cross-tasks and updates the meta-ranker through the meta-loss. Meta-loss indicates how well the model is performing across multiple tasks. In attempting to learn a meta-ranker this way, it could solve the generalization issue, especially in the sparsely labeled dataset.

Algorithm 1 shows the detailed steps of the meta-training process. First, we define two different learning rates α and β for query-specific ranker parameter updates and meta-ranker parameter updates, respectively. The model starts with initializing the meta-ranker parameters. Then it updates the parameters based on each batch, until convergence. For each batch, meta-training process could be summarized as following steps: First, initialize the query-specific ranker $f(\cdot;\theta_i)$ with the meta-ranker $g(\cdot;\theta)$ parameters $\theta_i = \theta$. Second, sample a batch of queries $\mathcal{S}_B$ from the $\mathcal{S}$, B denotes the batch size. Then, we can rewrite the loss function of query-specific ranker $\mathcal{L}_{query}$ for each query as the following:

$$\mathcal{L}_{query}(\theta_i) = \mathcal{L}_{\mathcal{S}_{train,i}}(\hat{y}_i, y_i) = \mathcal{L}_{\mathcal{S}_{train,i}}(f(\mathbf{x}_i;\theta_i), y_i), \tag{1}$$

where $\mathcal{S}_{train,i}$ represents the training set of query $q_i \in \mathcal{S}_B$, and $\hat{y}_i = f(\mathbf{x}_i;\theta_i)$ represents the model output of query q_i. $\mathcal{L}$ denotes the different ranking loss. This query-specific ranker aims to find optimal parameters θ_i for query q_i. It will be updated sequentially multiple times (denoted by T) through an inner loop. For each step in the inner training, we sampled K items from q_i's document collection D_i as the training set for this step.

Algorithm 1: Meta Learning to Rank (MLTR)

Require: $p(\mathcal{S})$: distribution over query-level tasks
Require: α, β: step size hyperparameters, K: sampled items number, T: inner loop number
Randomly initialize θ for meta ranker $g(\cdot)$
while not done **do**
  Sample a batch of queries $\mathcal{S}_B$ from $p(\mathcal{S})$
  for each query $q_i \in \mathcal{S}_B$ **do**
    Initialize query-specific ranker parameters $\theta_i = \theta$
    for inner loop $t = 1, \ldots, T$ **do**
      Sample K items $\mathcal{S}_{train:K,i}$ from D_i based on a sample strategy
      Evaluate $\nabla \mathcal{L}_{query}(\theta_i)$ using $\mathcal{S}_{train:K,i}$ and $\mathcal{L}_{query}(\theta_i)$ in Equation (1)
      Compute query-specific ranker parameters θ_i with gradient descent in Equation (2)
    end for
    Sample K items $\mathcal{S}_{test:K,i}$ from D_i based on a sample strategy
    Add $\mathcal{S}_{test:K,i}$ to $\mathcal{S}_{B:test}$
  end for
  Evaluate $\nabla \mathcal{L}_{meta}(\theta)$ using $\mathcal{S}_{B:test}$ and $\mathcal{L}_{meta}(\theta)$ in Equation (3)
  Update meta ranker θ in Equation (4)
end while

Next, the query-specific ranker parameters θ_i are updated by gradient descent of the query-specific loss $\mathcal{L}_{query}$ as the following:

$$\theta_i = \theta_i - \alpha \nabla \mathcal{L}_{query}(\theta_i), \tag{2}$$

where α denotes the learning rate of query-specific ranker $f(\cdot; \theta_i)$.

After updating the query-specific ranker $f(\cdot; \theta_i)$ for the task associated with query q_i, we sample K items from D_i to form the test set. This sampling excludes items from the training set used in the last inner loop T. It is important to note that an item may be present in the test set of the last inner loop T and also in the training sets of earlier inner loops $1, \ldots, T-1$. However, this overlap does not lead to data leakage issues, as both rankers operate within the scope of the meta-training process. Next step, we need to calculate the meta-loss and update the meta-ranker's parameters for optimizing all query-based tasks within batch $\mathcal{S}_B$. The meta-loss $\mathcal{L}_{meta}$ will be defined as

$$\mathcal{L}_{meta}(\theta) = \mathcal{L}_{\mathcal{S}_{B:test}}(\hat{y}_i, y_i) = \frac{1}{B} \sum_{i=1}^{B} \mathcal{L}_{\mathcal{S}_{test,i}}(g(\mathbf{x}_i; \theta_i), y_i), \tag{3}$$

where $\mathcal{S}_{B:test}$ represents the test set of query batch S_B, and $\hat{y}_i = g(\mathbf{x}_i; \theta_i)$ represents the model output of each query q_i respect to the meta-ranker $g(\cdot, \theta)$ and the updated parameters θ_i from the query-specific ranker, based on the training set. We let $\mathcal{L}$ denote the different ranking loss. We sum up the loss from each query of batch $\mathcal{S}_B$ and compute the average loss as the meta-loss from this batch. This meta-ranker aims to optimize parameters θ through the batch query $\mathcal{S}_B$, learns across multiple query-level tasks based on the query-specific ranker, and can improve the model generalization performance. We sampled K items from q_i's document collection D_i as the test set for this step.

Meta-ranker updates θ by gradient descent of the meta-loss $\mathcal{L}_{meta}$ as the following:

$$\theta = \theta - \beta \nabla \mathcal{L}_{meta}(\theta), \tag{4}$$

where β denotes the learning rate of meta-ranker $g(\cdot; \theta)$.

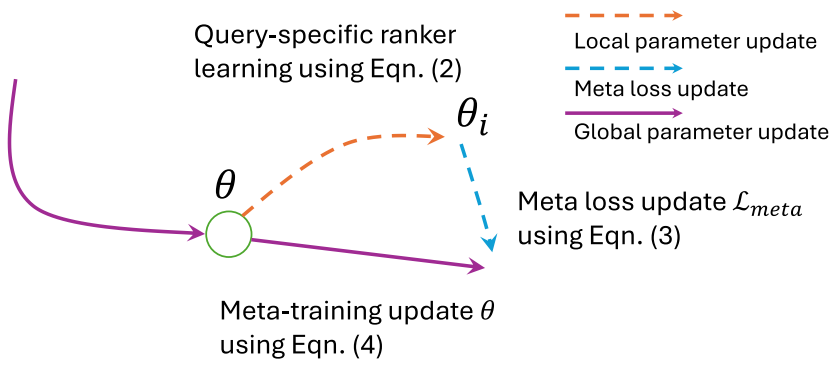


Fig. 2. Illustration of MLTR in meta-training which optimizes for a representation θ that can quickly adapt to new queries. The orange dashed line represents the query-specific ranker initialized from the meta-ranker and locally updated based on the training set data in meta-training. The blue dashed line represents the direction of meta-loss updates based on the updated query-specific ranker on test data in meta-training. The purple solid line represents the global updates of the meta-ranker based on the meta-loss.

Repeating the above batch level meta-training process, the query-specific ranker will continuously train on the training set $\mathcal{S}_{train}$, the meta-ranker will adapt and update meta-parameters θ on the test set $\mathcal{S}_{test}$ until the model parameter converges. Figure 2 shows an illustration of MLTR in meta-training which optimizes for a representation θ that can quickly adapt to new queries.

3.3.2 Meta-Testing. During the meta-testing phase, the meta-trained model (meta-ranker $g(\cdot)$) is used to make predictions on the meta-test queries/tasks. Different from the usual supervised learning model, the meta-trained model has a further fine-tuning process for additional gradient steps with few epochs on the fine-tuning set $\mathcal{T}_{tuning}$ before running the inference. This additional fine-tuning step enables the model parameters to fast adapt to new queries based on Equation (2), due to the learning-to-learn process. This meta-testing process accounts the fact that different queries have different optimal parameters for their rankers and thus reduce the impact of domain shift on the model. In consequence, the MLTR model has a significant advantage, particularly when facing new queries with distinct characteristics compared to the queries used in training. Additionally, we can also disable the fine-tuning mechanism and evaluate the model's performance using the evaluation dataset. The experiments in Section 5 demonstrate that MLTR with or without fine-tuning both improves model generalization for sparsely supervised queries.

4 Experimental Setup

4.1 Datasets

We evaluate the performance of our MLTR framework in the setting of sparsely supervised queries using four different datasets. The datasets include MQ2007, MQ2008, MSLR-10K,[1] and Istella-S LETOR,[2] which are public datasets widely used as benchmarks for LTR models [58]. These datasets consist of queries, retrieved documents, and labels provided by human experts. Furthermore, we used a real-world e-commerce dataset collected from a 1-month user log on Walmart.com. The focus of the dataset is on non-frequent tail queries, meaning the label distribution is extremely sparse. This dataset includes user search queries and the corresponding products in the search results, with labels (rating scores) ranging from 1 to 15 based on the level of user engagement. Query-product

[1] https://www.microsoft.com/en-us/research/project/letor-learning-rank-information-retrieval
[2] http://quickrank.isti.cnr.it/istella-dataset

Table 2. Basic Statistics of the Datasets

	Queries	Items	Query–Item Pairs	Positives	Features	Range of Ratings
MQ2007	1,692	65,323	69,623	25.84%	Sparse features (46)	0~2
MQ2008	784	14,384	15,211	19.28%	Sparse features (46)	0~2
MSLR-10K	10,000	1,200,192	1,200,192	47.99%	Sparse features (136)	0~4
Istella-S	33,018	3,408,630	3,408,630	11.39%	Sparse features (220)	0~4
Walmart Dataset	151,770	12,372,081	38,837,815	2.19%	Re-ranking feature (63), text feature	1~15

pairs that have been purchased receive the highest scores, whereas products that have received only clicks are assigned scores ranging in the middle. Products that have only received impressions are assigned scores lower than that of the click-only products. Negative items are assigned a score of 1. Scores for ordered products are calculated using a smoothed estimation of their order rate ($rate = \frac{order+\alpha}{impressions+\alpha}$), where α is the smoothing factor.

For all the above datasets, we first divide the raw data into meta-train, meta-validation, and meta-test sets, with a ratio of 80%, 10%, and 10%, respectively. Each query–document (item) pair is associated with a relevant rating label, which has different ranges for each dataset. Table 2 provides more details about the data statistics.

4.2 Sparsely Labeled Data

To simulate the sparse labeled queries, we further process the train, validation, and test datasets. In our experiments, we primarily control the number of positively labeled documents since it is usually limited in the real world and the negative documents can often be sampled from the dataset in a relatively large quantity. We perform a quantitative comparison on the simulated imbalanced datasets during training and testing.

We use $\mathcal{S}$ with superscript $p \cdot n \cdot$ to denote the number of sampled positive-/negative-labeled items per query in the training data (e.g., $\mathcal{S}_{train:p1n9}$ means the sampled training data with one positive-labeled items and nine negative-labeled items per query). Thus, $\mathcal{S}_{train}$ is chosen from

$$\{\mathcal{S}_{train:p1n4}, \mathcal{S}_{train:p1n9}, \mathcal{S}_{train:p1n19}, \mathcal{S}_{train:p1n29}, \mathcal{S}_{train:p1n39}\}$$

and $\mathcal{S}_{test}$ is chosen from

$$\{\mathcal{S}_{test:p1n4}, \mathcal{S}_{test:p1n9}, \mathcal{S}_{test:p1n19}, \mathcal{S}_{test:p1n29}, \mathcal{S}_{test:p1n39}\}.$$

Similarly, we use $\mathcal{T}$ with superscript $p \cdot n \cdot$ to denote the number of sampled positive-/negative-labeled items per query for model fine-tuning, $\mathcal{T}_{tuning}$ is chosen from

$$\{\mathcal{T}_{tuning:p1n4}, \mathcal{T}_{tuning:p1n9}, \mathcal{T}_{tuning:p1n19}, \mathcal{T}_{tuning:p1n29}, \mathcal{T}_{tuning:p1n39}\}$$

and the rest of the items of each query to evaluation our model with $\mathcal{T}_{eval:rest}$.

The validation set will be used to find the best model and hyper-parameters during the meta-training process and will be split in the same manner as the meta-test dataset.

4.3 Evaluation Metrics

For the evaluation of the ranking results in MLTR, we apply **Normalized Discounted Cumulative Gain (NDCG)** which is suitable for ranking where users are usually sensitive to the ranked position of the relevant items [37].

4.4 Baseline Methods

To verify the efficiency and compatibility of our proposed model, we refrained from directly using overly complex baseline models. Instead, we conducted experiments on simple models and observed

the resulting improvements to verify the effectiveness of our meta-learning based method for overall performance improvement in LTR models. On the other hand, due to the lack of semantic features for queries and corresponding documents in the public datasets MQ2007, MQ2008, MLSR-10K, and Istella-S, we did not utilize the corresponding text embedding representation features in our tests. Nevertheless, we supplemented the corresponding text embedding representation features using the BERT model with pre-trained weights from distillbert-base-uncased[3] based on the text information of the query and document in the subsequent Walmart.com dataset. We then conducted experiments to verify the effectiveness of these features. We compare MLTR with the following competitive baselines:

- —*LTR*: The LTR baseline is a three-layer MLP with a ReLu activation function. The ranking loss functions are introduced later in this section. To ensure fair comparisons, we perform fine-tuning on the test stage.
- —*LTR+SMOTE* [15]: This method is resampling-based and generates a resampled list using SMOTE, a popular oversampling strategy. We added the resampled data to the original training data and followed the same training and testing protocol as the LTR baseline model.
- —*LTR+GMVAE* [60]: This method is based on data augmentation and utilizes GMVAE to generate additional synthetic items. The GMVAE model is pre-trained with the entire training dataset, and then the augmented lists are produced. We added the synthetic data to the original training data and followed the same training and testing approach used with the LTR baseline model.
- —*LTR+Policy-Gradient* [50]: PL ranking models, a decision theory-based approach to ranking. This model employs Gumbel sampling techniques for efficient sampling of multiple rankings from a PL model. Following this, algorithms PL-Rank-1, PL-Rank-2, and PL-Rank-3 are applied to these samples. This process enables an unbiased approximation of the gradient of a ranking metric in relation to the model. For our experiments, we have adhered to the model parameters and implementation as detailed in the official repository[4] and adapted the data-loader to fit our experimental setup.
- —*LTR+Unbiased* [51]: Unbiased click-based LTR models, tailored specifically to adjust for position bias in click feedback. In our experiments, we deploy three distinct estimators. First, the IPS approach employs counterfactual IPS estimation to mitigate the selection bias linked to examination probabilities. Next, we utilize DM and DR approaches that account for position bias, trust bias, and item-selection bias, offering a more flexible criterion for unbiasedness compared to the widely used IPS method. Our implementation follows the model parameters outlined in the official repository[5] with $N = 10^{-4}$ for comparison, and we have adapted the data-loader to suit our experimental framework.

To ensure a fair comparison, most components in our proposed MLTR model employ the same model structure as the LTR baseline model. Regarding MLTR, we made corresponding adjustments to the model training process and data usage.

In traditional LTR models, three different types of loss functions, namely Pointwise, Pairwise, and Listwise [44], are usually used depending on the task and data. We use the following representative losses for the LTR baseline and MLTR: RankMSE, RankNet, LambdaRank, and ListNet losses.

[3]https://huggingface.co/distilbert-base-uncased
[4]https://github.com/HarrieO/2022-SIGIR-plackett-luce
[5]https://github.com/HarrieO/2022-doubly-robust-LTR

Pointwise Loss. It only takes into account a single document $d_{i,j}$ at a time for a query q_i. RankMSE algorithm [16] is as follows:

$$\mathcal{L}(f; \mathbf{x}_{i,j}, y_{i,j}) = \sum_{j=1}^{M} (f(\mathbf{x}_{i,j}) - y_{i,j})^2. \tag{5}$$

Pairwise Loss. It considers a pair of documents $< d_{i,j}, d_{i,s} >$ at a time for a query q_i if $y_{i,j} > y_{i,s}$ ($d_{i,j}$ should be ranked before $d_{i,s}$) [9]. RankNet algorithm [10] and LambdaRank algorithm [70] with their loss functions shown in Equation (6) are as follows:

$$\mathcal{L}(f; \mathbf{x}_{i,j}, \mathbf{x}_{i,s}) = \sum_{j=1}^{M-1} \sum_{s=1, y_{i,j} > y_{i,s}}^{M} \varphi(f(\mathbf{x}_{i,j}) - f(\mathbf{x}_{i,s})), \tag{6}$$

where φ denotes the Sigmoid function for RankNet loss, $\varphi(u) = \Delta NDCG(j,s) \log_2(1 + e^{-\sigma u})$ for LambdaRank loss, where σ is a hyper-parameter and $\Delta NDCG(j,s)$ is the absolute difference between the NDCG values when two documents $d_{i,j}$ and $d_{i,s}$ are swapped.

Listwise Loss. It directly looks at the entire list of documents D_i and tries to come up with the optimal ordering for each query q_i [12]. For example, the loss function for the ListNet algorithm is as follows:

$$\mathcal{L}(f; \mathbf{x}_i, y_i) = \sum_{i=1}^{N} L(f(\mathbf{x}_i), y_i), \tag{7}$$

where $L(\cdot)$ denote the cross-entropy loss. $f(\mathbf{x}_i)$ is the predict label for query q_i. y_i denotes the true label of each document in query q_i.

4.5 Research Questions (RQs)

An extensive set of experiments were designed to address the following questions of the proposed research:

- RQ1: Can the proposed MLTR framework achieve improved performance on sparsely labeled queries over the baseline methods? (Section 5.1)
- RQ2: How does the training and test mechanism designed for MLTR effectively improve model performance compared to traditional model training processes? (Section 5.2.1)
- RQ3: Without fine-tuning toward a specific query in test data, can MLTR still improve model generalization? (Sections 5.2.2 and 5.2.3)
- RQ4: Can MLTR alleviate the data sparsity issue and domain shift problem? How much NDCG lift can the MLTR models gain over the traditional LTR models? Is the amount of NDCG relative gain correlated with training/test data's sparseness? (Section 5.3)
- RQ5: Can MLTR be effective in real-world applications with limited labeled data and result in improved performance? (Section 5.4)

5 Experimental Results

In this section, we conduct experiments on the datasets introduced in Section 4.1. We compare the proposed MLTR model and the baseline LTR models under different scenarios, taking into account multiple ranking loss functions and multiple simulated data sparsity cases.

5.1 Baseline Comparison (RQ1)

We compare the performance of MLTR to traditional LTR models when handling sparsely labeled queries. We tested our models on the four public datasets by simulating sparse data scenarios. The process involved training on $\mathcal{S}_{train:p1n9}$ and $\mathcal{S}_{test:p1n9}$, followed by fine-tuning on $\mathcal{T}_{tuning:p1n9}$

Table 3. Comparative Performance of Baseline Models and the MLTR Framework in Terms of NDCG@1, NDCG@5, and NDCG@10 Metrics on the Evaluation Set $\mathcal{T}_{eval:rest}$

Method	MQ2007 NDCG@1/5/10	MQ2008 NDCG@1/5/10	MSLR-10K NDCG@1/5/10	Istella-S NDCG@1/5/10
		LTR		
RankNet	0.4090/0.4672/0.5166	0.5000/0.5931/0.6461	0.3392/0.3594/0.3886	0.6146/0.6335/0.702
RankMSE	0.4286/0.4501/0.4946	0.4583/0.5407/0.6030	0.3624/0.3723/0.3974	0.6255/0.6412/0.706
ListNet	0.4454/0.4857/0.5354	0.4722/0.5796/0.6309	0.3984/0.3994/0.4224	0.6279/0.6433/0.7088
LambdaRank	0.4314/0.5025/0.5441	0.5972/0.6264/0.6907	0.3731/0.3808/0.4097	0.6146/0.6354/0.7054
		LTR + SMOTE		
RankNet	0.4762/0.5114/0.5603	0.4861/0.5937/0.6557	0.3584/0.3695/0.3978	0.6279/0.6371/0.7041
RankMSE	0.4286/0.4897/0.5351	0.5000/0.5877/0.6524	0.3640/0.3823/0.4085	0.6114/0.6263/0.6912
ListNet	0.4762/0.4959/0.5523	0.4861/0.6095/0.6603	0.3737/0.3789/0.4061	0.6207/0.6297/0.6928
LambdaRank	0.4622/0.5064/0.5469	0.5972/0.6452/0.6861	0.3479/0.3576/0.3871	0.6217/0.6359/0.700
		LTR + GMVAE		
RankNet	0.4930/0.5006/0.5412	0.5417/0.6352/0.6886	0.3570/0.3739/0.3989	0.6056/0.6281/0.6967
RankMSE	0.4454/0.4861/0.5168	0.4861/0.5617/0.6348	0.3582/0.3630/0.3879	0.6036/0.6286/0.6946
ListNet	0.4622/0.4820/0.5274	0.4167/0.5467/0.6327	0.3832/0.3829/0.4043	0.5997/0.6261/0.6943
LambdaRank	0.4762/0.4873/0.5369	0.5556/0.6237/0.6844	0.3627/0.3827/0.4091	0.5927/0.6187/0.6885
		LTR + Policy-Gradient		
PL-Rank-1	0.4416/0.5046/0.5342	0.5611/0.6133/0.6569	0.3489/0.3614/0.3906	0.6051/0.6110/0.6764
PL-Rank-2	0.4400/0.4975/0.5275	0.5485/0.6124/0.6502	0.3448/0.3591/0.3822	0.6129/0.6159/0.6788
PL-Rank-3	0.4458/0.5036/0.5426	0.5563/0.6209/0.6825	0.3455/0.3649/0.3958	0.6108/0.6284/0.6921
		LTR + Unbiased		
IPS	0.3921/0.4866/0.5154	0.6102/0.6514/0.6968	0.3383/0.4008/0.4100	0.6497/0.6683/0.7019
DM	0.3808/0.4715/0.5042	0.5994/0.6491/0.6979	0.3246/0.3918/0.4289	**0.6597**/0.6512/0.7048
DR	0.4226/0.4736/0.5159	0.6108/0.6465/0.6985	0.3613/0.3981/0.4246	0.6535/**0.6817**/0.7002
		MLTR + without Fine-Tuning		
RankNet	0.4874[a]/0.4895[a]/0.5444[a]	0.5694[a]/0.6048[a]/0.6667[a]	0.3592[a]/0.3702[a]/0.3991[a]	0.6159/0.6339/0.7028
RankMSE	0.4454[a]/0.4788[a]/0.5191[a]	0.5833[a]/0.5964[a]/0.6544[a]	0.3867[a]/0.3917[a]/0.4158[a]	0.6275/0.6400/0.7082
ListNet	0.5042[a]/0.5068[a]/0.5519[a]	0.5694[a]/0.6116[a]/0.6669[a]	0.4002/0.3996/0.4230	0.6336[a]/0.6465/0.7141[a]
LambdaRank	0.5014[a]/0.5139[a]/0.5669[a]	0.6250[a]/0.6504[a]/0.6981[a]	0.3662/0.3798/0.4092	0.6072/0.6307/0.7024
		MLTR + with Fine-Tuning		
RankNet	**0.5770**[a]/**0.5460**[a]/0.5913[a]	0.6250[a]/0.6452[a]/0.6949[a]	0.3873[a]/0.3889[a]/0.4122[a]	0.6212[a]/0.6385[a]/0.7071[a]
RankMSE	0.4902[a]/0.5238[a]/0.5646[a]	0.5972[a]/0.6505[a]/0.6985[a]	0.3887[a]/0.3981[a]/0.4237[a]	0.6362[a]/0.6450[a]/0.7112[a]
ListNet	0.5266[a]/0.5254[a]/0.5704[a]	0.6111[a]/0.6473[a]/0.7027[a]	**0.4088**[a]/**0.4100**[a]/**0.4342**[a]	0.6388[a]/0.6490[a]/**0.7172**[a]
LambdaRank	0.5350[a]/0.5409[a]/**0.5914**[a]	**0.6389**[a]/**0.6590**[a]/**0.7130**[a]	0.3739[a]/0.3850[a]/0.4119[a]	0.6196[a]/0.6397[a]/0.7089[a]

This evaluation encompasses four publicly available datasets: MQ2007, MQ2008, MSLR-10K, and Istella-S. The highest scoring results for each task and metric are emphasized.

[a]Indicates a statistically significant improvement of MLTR (with and without fine-tuning) over the corresponding LTR models. This is evidenced by a p-value < 0.01 in a two-tailed t-test.

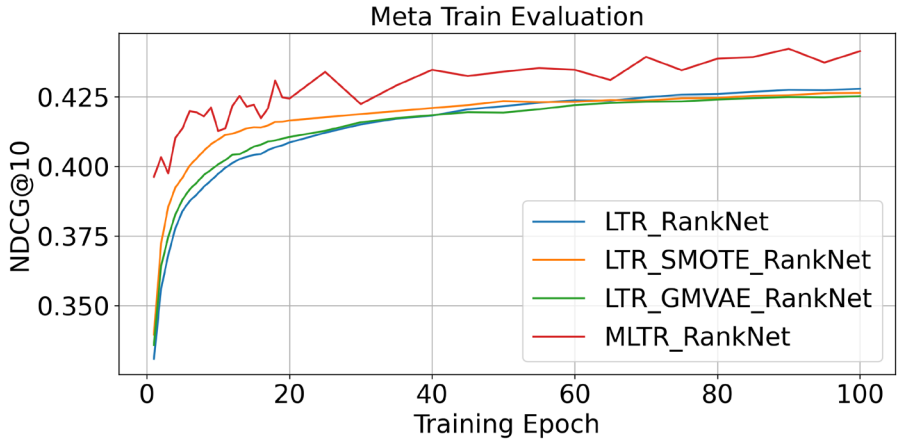
and evaluating the results on $\mathcal{T}_{eval:rest}$. The results shown in Table 3 indicate that our MLTR models, regardless of being in a without fine-tuning or with fine-tuning setting, outperform the baseline models in all metrics (NDCG@1, NDCG@5, and NDCG@10) across all four datasets, with the exception of the unbiased click-based LTR baseline models. This improved performance is maintained across various loss functions. Notably, on the MQ2007 and MQ2008 datasets, where the positive sample distribution is relatively sparse, the MLTR model shows significant improvement across all loss functions, providing further evidence that the meta-learning approach can enhance the

model's predictive ability in sparse data. While the MSLR dataset has a relatively even distribution of positive and negative samples overall, the MLTR model still manages to improve the model's predictive results in most of the loss functions. The results also suggest that the SMOTE resampling technique can alleviate the issue of sparse data and improve performance compared to the traditional LTR model. The GMVAE-based data augmentation method outperforms the SMOTE resampling method in most cases as it can incorporate more informative data. However, data augmentation-based methods do not significantly enhance model generalization compared to our proposed MLTR approach. In comparing the PL-Rank methods with MLTR, we found that the PL ranking methods lack consistent stability, particularly underlined by our experiments that highlight the sparse nature of positive samples during training. In contrast, the test results indicate that MLTR yields more stable outcomes in scenarios characterized by a limited number of training samples or a sparser distribution of positive samples. On the other hand, when examining the results of Unbiased click-based LTR methods, these methods show a notable advantage when the training samples contain rich features in the query and document pairs. For instance, on the Istella-S dataset, DM and DR achieved the best performance in NDCG@1 and NDCG@5. However, in the other three datasets, the MLTR model displayed a more consistent performance advantage. In our proposed MLTR model, we did not strictly address the bias present in the data. The experimental outcomes of unbiased click-based LTR methods indicate that incorporating unbiased methods like IPS into the training process of meta-learning might further enhance the performance of meta-learning-based LTR models. We plan to implement and evaluate this approach in our future work. On the other hand, regardless of whether the MLTR model utilizes the fine-tuning process during meta-testing, it consistently demonstrates competitiveness compared to traditional methods. The results of MLTR without fine tuning still lead in most experiments, surpassing traditional LTR models as well as baseline models with other optimization approaches. Furthermore, if the fine-tuning process in meta-testing is employed, we find that MLTR can adapt more rapidly to changes in queries, thereby further enhancing the model's performance on test evaluation data. When dealing with sparsely labeled queries, our MLTR model can achieve better adaptability with a small proportion of labeled data, leading to improved overall model performance.

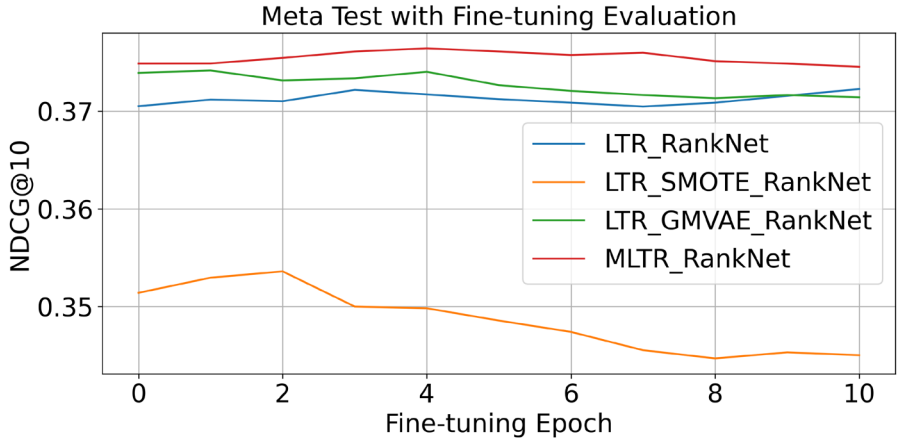
5.2 Ablation Study

5.2.1 The Effect of Meta-Train and Meta-Test (RQ2). Figure 3 illustrates the performance trend of the NDCG@10 metric on the test data during the training process (as shown in Figure 3(a)) and the fine-tuning process (as shown in Figure 3(b)) for both MLTR and baseline models based on the RankNet loss. In Figure 3(a), NDCG@10 of the test data is calculated by fine-tuning the model for 1 epoch on the meta-test tuning data after each training epoch ($1 \leq e \leq 100$) during the training process. The results show that MLTR consistently outperforms the LTR and other baseline models during the training process and can achieve close to its best performance within only a few epochs.

Figure 3(b) demonstrates the performance of MLTR and baseline models on the test data during the fine-tuning process. The model (the best model from meta-training stage) was fine-tuned for 10 epochs on the meta-test test data, with NDCG@10 computed for each epoch. The results show that MLTR consistently performs better than the LTR and other baseline models throughout the fine-tuning process. Our model demonstrates clear and stable performance on unseen datasets through a straightforward fine-tuning process, mitigating the effects of label imbalance and potential domain shifts. Additionally, MLTR still significantly outperforms the baseline models even without any fine-tuning on the meta-test test data, as shown by the NDCG@10 metrics at epoch 0 in Figure 3(b). During fine-tuning, MLTR continues to improve and outperform the corresponding baseline models under the RankNet loss from epoch 0 to epoch 4. On the other hand, the baseline models tend to suffer from over-fitting problems, resulting in a decline in performance.



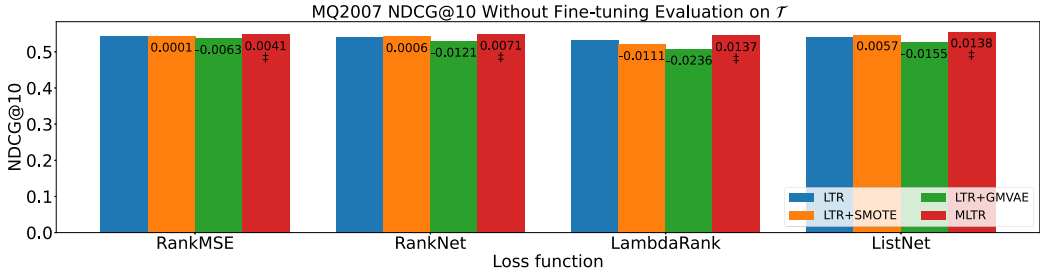
(a) Meta train evaluation based on meta test single fine-tuning



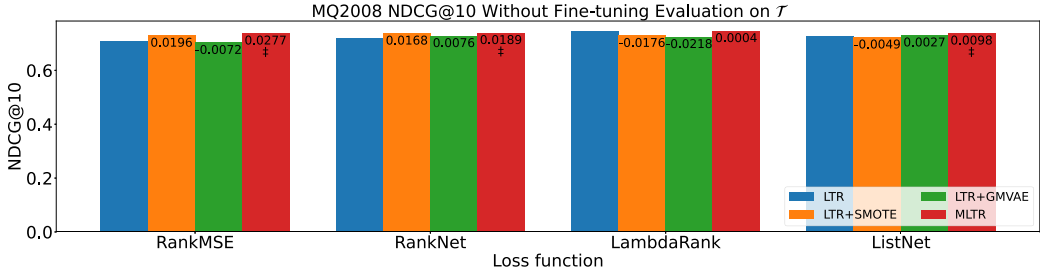
(b) Meta test fine-tuning evaluation with the best training model

Fig. 3. Meta-train/test evaluation on NDCG@10 of MLTR and other baselines with RankNet on the MSLR-10K dataset.

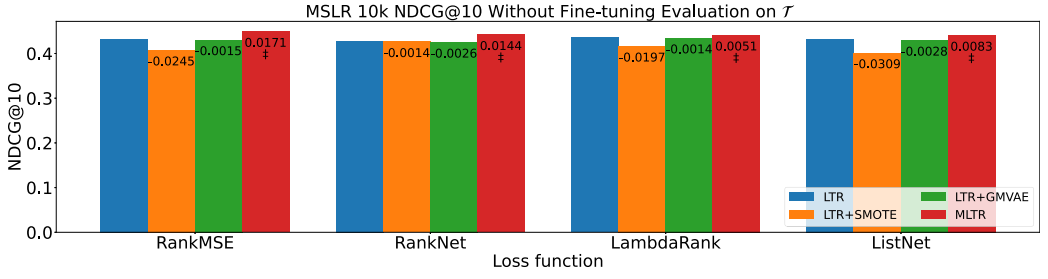
5.2.2 Meta-Test without Fine-Tuning (RQ3). In this section, we delve deeper into the results of our model's ability to maintain competitiveness on unseen datasets without fine-tuning. Figure 4 provides a comparison of the performance of our MLTR model with the baseline models on various datasets using the same entire test dataset $\mathcal{T} = \mathcal{T}_{tuning} \cup \mathcal{T}_{eval}$. As we can see, our model still has stronger prediction ability for unseen data or distribution compared to the other models. We have calculated the absolute growth of the data-augmentation based model and MLTR as compared to the baseline LTR across multiple metrics. When comparing with other models, we can see that MLTR consistently outperforms the LTR model across all 16 experiments, which encompass 4 datasets and 4 different loss functions. It is evident that the red bars, symbolizing MLTR, consistently exhibit an increase in performance in all comparative experiments relative to other methods. While the absolute magnitude of this growth may not appear substantial, the consistent improvement observed across four distinct datasets and four diverse optimization ranking functions underscores the robust reliability of the MLTR approach. Additionally, the significant test results further demonstrate



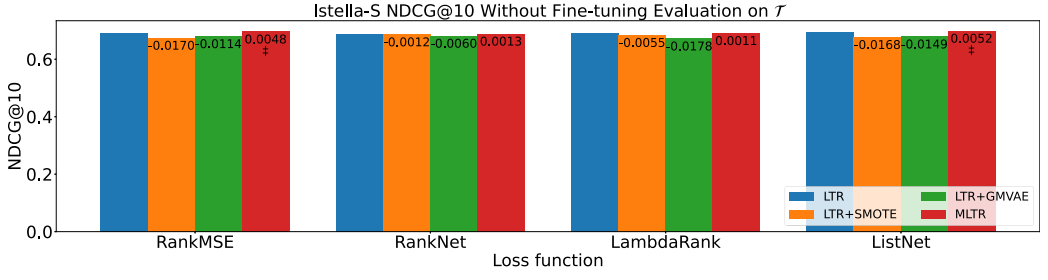
(a) MQ2007



(b) MQ2008



(c) MSLR-10K



(d) Istella-S

Fig. 4. Comparison of the performance of models without fine-tuning, using various loss functions and models, on the NDCG@10 metric of the entire test dataset $\mathcal{T}$ from four different public datasets. The symbol ‡ in the bar indicates a statistically significant improvement of MLTR without fine-tuning over the corresponding LTR models, as evidenced by a p-value < 0.01 in a two-tailed t-test.

Table 4. Comparative Analysis of LTR and MLTR Frameworks Using NDCG@1, NDCG@5, and NDCG@10 Metrics on the MSLR-10K Dataset

Model	Method	Without Fine-Tuning		With Fine-Tuning	
		NDCG@1/5/10	Percentage Increase	NDCG@1/5/10	Percentage Increase
LTR	LambdaRank	0.3359/0.3757/0.4086	-/-/-	0.3411/0.3809/0.4161	-/-/-
MLTR	LambdaRank	0.3555/0.3816/0.4158	5.83%/1.58%/1.76%	0.3582/0.3822/0.4181	5.00%/0.34%/0.48%
LTR	ListNet	0.3290/0.3703/0.4087	-/-/-	0.3328/0.3729/0.4107	-/-/-
MLTR	ListNet	0.3632/0.3898/0.4259	10.37%/5.26%/4.22%	0.3634/0.3898/0.4268	9.20%/4.55%/3.91%
LTR	RankMSE	0.3129/0.3418/0.3777	-/-/-	0.3136/0.3430/0.3785	-/-/-
MLTR	RankMSE	0.3631/0.3868/0.4183	16.06%/13.17%/10.76%	0.3631/0.3868/0.4188	15.78%/12.78%/10.66%
LTR	RankNet	0.3188/0.3545/0.3919	-/-/-	0.3191/0.3547/0.3923	-/-/-
MLTR	RankNet	0.3462/0.3805/0.4107	8.57%/7.32%/4.79%	0.3463/0.3805/0.4108	8.50%/7.26%/4.71%

This analysis reflects a training approach where each query is paired with one positive document and a random number of negative documents. The evaluation is conducted consistently on the same dataset.

that the majority of these improvements are statistically significant. This consistency aligns with the results shown in Table 3. On the other hand, we noted that the data augmentation-based LTR models, which aimed to rebalance the ratio of positive and negative samples in the training set using synthetic data, did not uniformly improve performance during testing. In fact, on some metrics, their performance was even worse than traditional LTR models. Furthermore, the results from these datasets highlight that while data augmentation helps mitigate the impact of imbalanced positive and negative samples, it does not effectively enhance the model's generalization ability when facing domain shift issues in testing.

5.2.3 MLTR with Query–Document Pairs (RQ3). To better validate the universality of the MLTR model, we introduced a new set of comparative experiments within the MLTR-10K dataset. For each query, we randomly selected 2 positive samples and 78 negative samples. During the meta-training process, instead of adhering to a fixed p1n39 positive-negative ratio, we opted for a variable number of negative samples while keeping one positive sample constant. This experimental setting is designed to test whether MLTR outperforms LTR in scenarios with varying numbers of documents per query. The results in Table 4 demonstrate that MLTR, even with dynamically adjusted numbers of documents per query, still shows a significant advantage over LTR methods across different optimization methods. We also compared results before and after fine-tuning. These results further confirm that MLTR consistently outperforms LTR models, regardless of fine-tuning, underscoring MLTR's superior adaptability to new tasks.

5.3 Robustness of MLTR (RQ4)

This section demonstrates the superiority of our meta-learning model over the baseline when dealing with extremely sparse data and a low positive-to-negative label ratio. The evaluation was conducted on the MSLR-10K dataset, and various experimental scenarios were simulated by sampling subsets of the data with varying ratios of positive and negative labels per query.

5.3.1 Experiment Setup. The factors to consider in this experiment include the number of sampled positive-/negative-labeled items per query in the training data, the number of sampled positive-/negative-labeled items per query in the test data, and the training model. For the model training, we compare the model performance between the baseline LTR model and the MLTR model with RankNet loss for both models, denoted by MLTR and LTR, respectively. We use NDCG@10 as the evaluation metric. There is no overlap between any sampled training and test data in order to ensure the fairness of the experiment. The sampled training and test data are introduced in

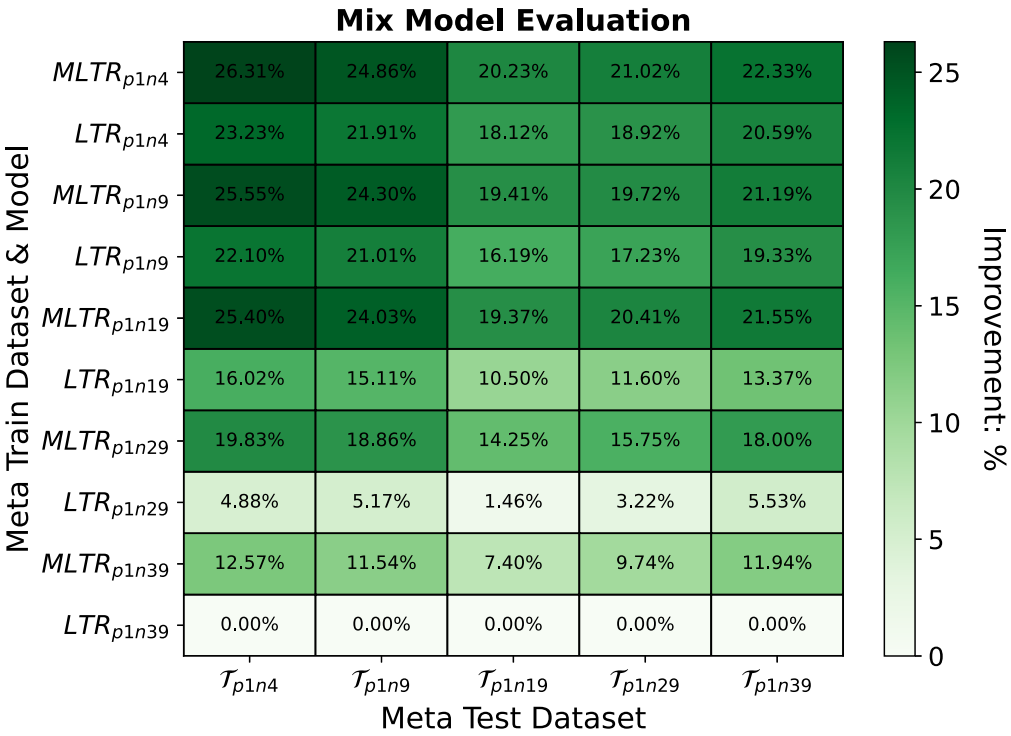


Fig. 5. Relative improvement experimental results of NDCG@10 from MLTR and LTR based on RankNet loss in variant sparsely labeled data setting on MSLR-10K dataset.

Section 4.2. Given a combination of the training model, $\mathcal{S}$, $\mathcal{T}$, we can obtain the NDCG@10 metrics on $\mathcal{T}_{eval}$ with the best model trained on $\mathcal{S}$ and fine-tuned on $\mathcal{T}_{tuning}$ with 1 epochs. The experimental results for all combinations are shown in Figure 5.

5.3.2 Data Distribution Shift Evaluation. Figure 5 shows the relative NDCG@10 gain on the worst-performing model LTR_{p1n39}. For each test data $\mathcal{T}$ (represented by x-axis) corresponding to a column, the NDCG@10 metrics are computed for a model (LTR or MLTR) trained on $\mathcal{S}$ (represented by y-axis). $LTR_{p\cdot n\cdot}$ and $MLTR_{p\cdot n\cdot}$ denote the model LTR and MLTR trained on specific training datasets, respectively; each grid in this column shows the relative NDCG@10 improvement compared against LTR_{p1n39} on the evaluation data $\mathcal{T}_{eval}$. The darker the color of each grid, the greater the improvement of the model in this grid relative to LTR_{p1n39}.

We have the following observations on Figure 5. First, for any given $\mathcal{S}$ and $\mathcal{T}$, MLTR performs better than LTR consistently, with a significant 1.86–14.95% improvement. For example, $MLTR_{p1n29}$ improves 12.53% over LTR_{p1n29} on the test data $\mathcal{T}_{p1n29}$ (15.75% for MLTR vs. 3.22% for LTR shown in Figure 5). Second, we observe that MLTR is much more stable and robust to the sparse data than LTR, by comparing all models' performance for a fixed test data $\mathcal{T}$ (corresponding to a column). As $\mathcal{S}$ gets more sparse, performance degradation is observed for both MLTR and LTR models; however, MLTR's NDCG performance decreases much slower compared to LTR. For example, looking at these models' NDCG metrics on $\mathcal{T}_{p1n4}$ (corresponding to column 1) as the training data get more sparse from $\mathcal{S}_{p1n4}$ to $\mathcal{S}_{p1n39}$, NDCG for LTR decreases by 23.23%, from 23.23% (LTR_{p1n4}) to 0% (LTR_{p1n39}), while NDCG for MLTR decreases by 13.74% from 26.31% ($MLTR_{p1n4}$) to 12.57% ($MLTR_{p1n39}$). In addition, as pointed out in Section 5.3, all the models evaluated in Figure 5 go

Table 5. NDCG@1 and NDCG@5 Gain Are Reported in Terms of the Percentage Lift for MLTR over LTR on Various Loss of Walmart Dataset

Loss	Gain NDCG@1	Gain NDCG@5
RankMSE	+0.44%[a]	+0.95%[a]
RankNet	+1.99%[a]	+0.92%[a]
LambdaRank	+2.58%[a]	+1.04%[a]
ListNet	+2.32%[a]	+1.51%[a]

[a]Denotes statistically significant improvement from LTR to MLTR with the p-value < 0.01 using the two-tailed t-test

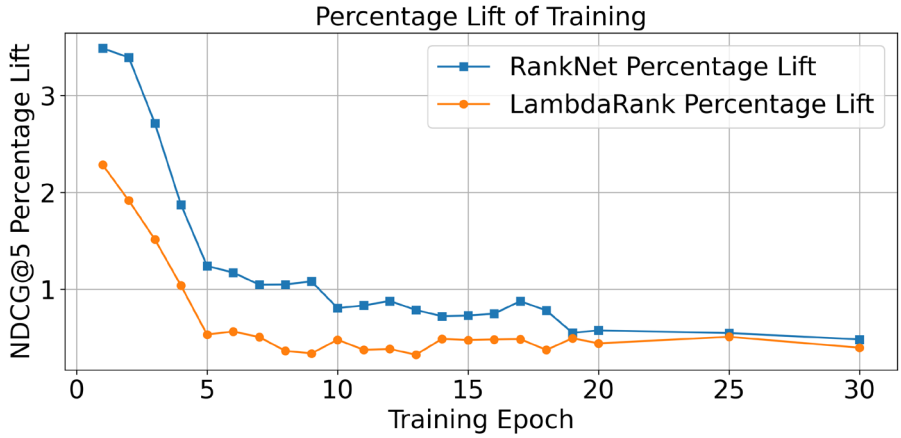
through only one training epoch on $\mathcal{T}_{tuning}$. With MLTR's significant improvement over LTR under all the scenarios, we show that the meta-based LTR models can generalize and adapt significantly better than the traditional LTR models under sparsely labeled data settings.

5.4 Real-World Application Case Study (RQ5)

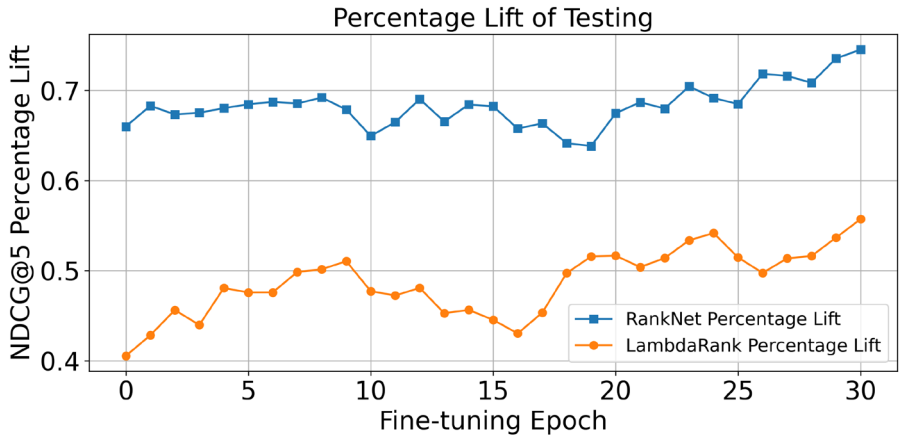
The study is performed on the real-world Walmart.com dataset, which has sparse positive-labeled queries as shown in Table 2. It is worth conducting the robustness experiments to evaluate the model generalization, as the data in the real world are often more dynamic with drifted distributions.

5.4.1 Experimental Results. Table 5 shows the percentage lift in NDCG@1 and NDCG@5 for MLTR over LTR on the Walmart dataset for various loss functions. The results from the Walmart dataset align with the patterns observed in the public datasets. Our MLTR models outperform the traditional LTR models in terms of both NDCG@1 and NDCG@5 metrics in sparsely labeled data scenarios. On the other hand, Figure 6(a) illustrates the NDCG@5 gain between the MLTR and LTR models using RankNet and LambdaRank losses. The same training method as used on the public dataset was employed, with the model fine-tuned for 1 epoch on the test support data at the end of each training epoch ($1 \leq e \leq 30$) during the training process. The NDCG@5 gain was then computed based on $(MLTR_{NDCG@5} - LTR_{NDCG@5})/LTR_{NDCG@5}$. The results demonstrate that the MLTR consistently outperforms the LTR models in real-world application datasets. In the early stages of training, the MLTR model exhibits a greater improvement over LTR, demonstrating the efficiency of the MLTR model and its faster convergence speed. As the number of training iterations increases, both MLTR and LTR models become relatively stable, but the MLTR model still performs better than the traditional LTR model. This conclusion holds true for both the human-annotated relevance label sparsity setting seen in the three public datasets and the engagement label sparsity setting demonstrated in the Walmart dataset. During the testing process illustrated in Figure 6(b), the MLTR model consistently outperforms the LTR models throughout. Similar results were observed in the implementation on public datasets.

After comparing the NDCG@5 gain ratios during both the training and fine-tuning processes depicted in Figure 6, it can be observed that the MLTR model consistently outperforms the LTR models. During training, the MLTR model exhibits a significant improvement over LTR, between 0.33% and 3.49%. Similarly, during fine-tuning under similar conditions, the MLTR model achieves varying levels of performance improvement, ranging between 0.41% and 0.73%. Although the improvement ratio during training is not as pronounced, it still indirectly validates the efficiency and compatibility of the MLTR model with respect to the data and task.



(a) NDCG@5 percentage lift comparison between LTR and MLTR models based on 30 training epochs in meta train evaluation



(b) NDCG@5 percentage lift comparison between LTR and MLTR models based on 30 fine-tuning epochs in meta test evaluation

Fig. 6. Meta-train/test evaluation of NDCG@5 percentage lift for MLTR and LTR models using RankNet and LambdaRank on the Walmart.com dataset.

5.4.2 Sampling Strategy in Meta-Training. With the design of the inner loop during local updating on the meta-train data in the MLTR, we can sample a data subset for the model's local update with different sample strategies. This sampling strategy during the local update aims to improve the model performance on the specific query-based tasks and thus improves the model's generalization performance on a new query. In this section, in order to further boost the model performance under the data sparsity setting, we explore different sampling strategies in the MLTR model by using subsets of the data as training data. Several sampling strategies we investigate are defined as follows:

—*All Data*: Use all data (256 items for each query) with baseline LTR model.
—*Fixed Sampler*: Fixed sample 1 positive and 19 negatives data with MLTR model.

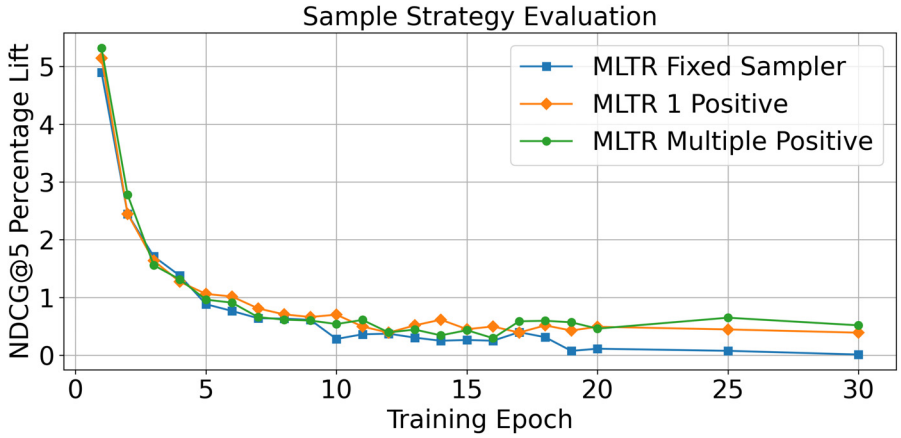


Fig. 7. NDCG@5 percentage lift in model performance using various sample strategies compared to $LTR_{Alldata}$ during the meta-train evaluation of the MLTR model with RankNet on Walmart.com dataset.

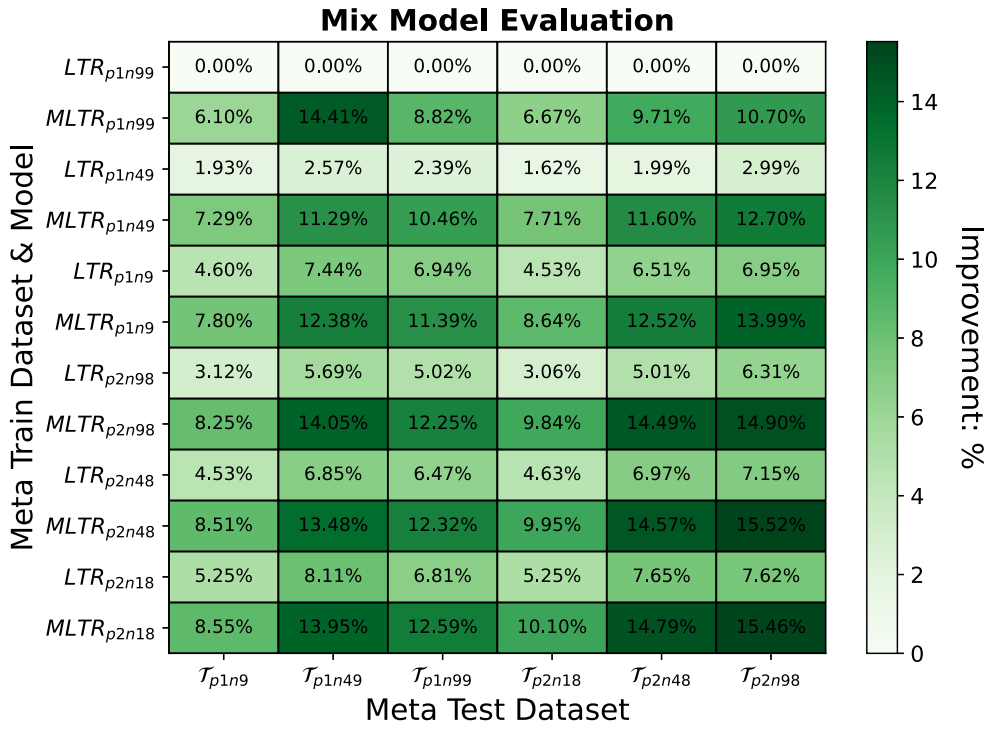
— *1 Positive Sampler*: Randomly sample 1 positive and 19 negatives data each time in training with MLTR model.
— *Multiple Positive Sampler*: Randomly sample 2 positives and 18 negatives data each time in training with MLTR model.

As shown in Figure 7, as the number of sampled positive data increases, the performance of the MLTR model shows a slight improvement, with the green line demonstrating the highest improvement, followed by the yellow line, and then the blue line. In addition, all the MLTR models with sampling strategies (green, yellow, and blue lines) outperform the baseline LTR model with *All data* (the NDCG@5 percentage lift above 0). We can see that compared to using all the data for training LTR models, using variant partial of the data for each inner loop during the meta-training can not only reduce the amount of computation during the training but also improve the model generalization.

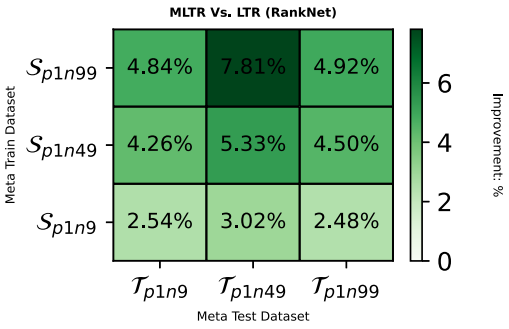
5.4.3 BERT with Fine-Tuning. As we mentioned in the previous section, we extend the features with query and document text embedding representation; this semantics information is available in the e-commerce dataset as the query and item title. We generate query text embedding and item title text embedding from the pre-trained distilled BERT model distillbert-base-uncased;[6] then the BERT model parameters are fine-tuned during the meta-learner update, similar as MeLu [43]. No significant improvement is observed by using the Bert-based query/item text embeddings. This is not surprising since the BERT-based embedding information for a *query*, *production* pair is already covered by a numeric feature in the e-commerce dataset, which is computed as the cosine similarity between a Bert-based query embedding vector and item embedding vector [47].

5.4.4 Data Distribution Shift in Walmart.Com. In previous experiments, we simulated sparsely labeled data settings on public datasets. However, the Walmart.com tail query dataset has an even sparser data distribution and more severe data shift issues. Therefore, it is worthwhile to conduct robustness experiments similar to those on public datasets to verify the model's performance on real-world application data. First, we used a configuration similar to the robustness experiments on public datasets and followed the same experimental training and validation process as Section 5.3.2.

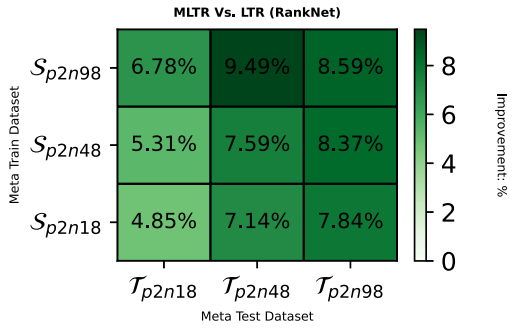
[6] https://huggingface.co/distilbert-base-uncased



(a) NDCG@5 percentage improvement over LTR_{p1n99}



(b) Single positive label comparison



(c) Multiple positive label comparison

Fig. 8. Relative improvement experimental results from MLTR and LTR based on RankNet loss in variant sparsely labeled data setting on the Walmart.com dataset.

The difference was that we collected a more sparse ratio of positive and negative samples to validate our model's performance in a true application setting.

Figure 8(a) shows the relative NDCG@5 gain on the worst-performing model LTR_{p1n99}. For each test data $\mathcal{T}$ (represented by x-axis) corresponding to a column, the NDCG@5 metrics are computed for a model (*LTR* or *MLTR*) trained on $\mathcal{S}$ (represented by y-axis), $LTR_{p \cdot n \cdot}$ and $MLTR_{p \cdot n \cdot}$ denote the model *LTR* and *MLTR* train on specific training dataset, separately; each grid in this column shows

the relative NDCG@5 improvement compared against BR_{p1n99} on the same test data $\mathcal{T}$. The darker the color of each grid, the greater the improvement of the model in this grid relative to LTR_{p1n99}.

First, we observed that, similarly to the public dataset experiments, *MLTR* consistently outperformed *LTR* for any given $\mathcal{S}$ and $\mathcal{T}$, with a significant improvement of 3.2–10.7%. For instance, in the test data $\mathcal{T}_{p1n99}$, $MLTR_{p1n49}$ shows an improvement of 8.07% over LTR_{p1n49} (as shown in Figure 8(a)), with *MLTR* achieving a 10.46% improvement compared to a 2.39% improvement for *LTR*. Second, we noticed that *MLTR* is considerably more stable and robust in the face of sparse data than *LTR*. This is evident when comparing the performance of all models for a fixed test dataset $\mathcal{T}$ (corresponding to a column), even in the case of more sparse datasets. As $\mathcal{S}$ gets more sparse, performance degradation is observed for both *MLTR* and *LTR* models; however, *MLTR*'s NDCG performance decreased much slower compared to *LTR*, this observation applies to both scenarios of one positive-labeled item $p1n\cdot$ and two positive-labeled items $p2n\cdot$ per query. For example, looking at these models' NDCG metrics on $\mathcal{T}_{p1n9}$ (corresponding to column 1) as the training data get more sparse from $\mathcal{S}_{p2n18}$ to $\mathcal{S}_{p2n98}$, NDCG for *MLTR* decreases by 2.13%, from 5.25% (LTR_{p2n18}) to 3.12% (LTR_{p2n98}), while NDCG for *MLTR* decreases only by 0.3% from 8.55% ($MLTR_{p2n18}$) to 8.25% ($MLTR_{p2n98}$).

We further compared the relative NDCG gain of MLTR over LTR for a combination of $\mathcal{S}$ and $\mathcal{T}$. Figure 8(b) and (c) correspond to the cases with one positive-labeled item and with two positive-labeled item per query, respectively. Each grid in Figure 8(b) and (c) represents the relative NDCG@5 lift of MLTR over LTR when both models are trained on $\mathcal{S}$ (represented by y-axis) and tested on $\mathcal{T}$ (represented by the x-axis). Take the lower right corner grid in Figure 8(b) as an example, it shows a relative 2.48% improvement of $MLTR_{p1n9}$ over LTR_{p1n9} on the test data $\mathcal{T}_{p1n99}$. First, looking at models' performance on each test data $\mathcal{T}$ (corresponding to each column), we see consistent patterns in Figure 8(b) and (c) that the sparser the positive labels in $\mathcal{S}$, the higher the relative NDCG lift of MLTR over LTR. Second, fixing the training data $\mathcal{S}$ (corresponding to each row) in both Figure 8(b) and (c), we find out that the sparse the test data is, the more relative improvement of MLTR over LTR overall, with an exception of the most extreme case $\mathcal{T}_{p1n99}$. Third, focusing on the diagonal grids (when $\mathcal{S}$ and $\mathcal{T}$ have the same number of positive-/negative-labeled items for each query), we see that the sparser the data, the bigger gap between MLTR and LTR in overall except the case of ratio 1:99. Overall, the sparser the training data and the test data, the more significant NDCG lift of MLTR over LTR can be observed. This shows the MLTR model improves the generalization performance over the baseline LTR model on the sparse setting in real-world Walmart dataset.

6 Conclusion and Future Work

In this article, we introduce a novel MLTR framework that improves the generalization capability of LTR models for search and ranking tasks with sparsely labeled queries. Our proposed model enables quick adaptation to new queries with limited supervision and produces query-specific rankers with optimal model parameters. Comprehensive experiments demonstrate the versatility and flexibility of our approach, which can be applied to any existing LTR models. Real-world application experiments also further illustrate the effectiveness of our proposed approach.

This work is an initial step toward a promising research direction. First of all, we will explore the application of the proposed framework to neural ranking models and see if the improvement is sustained. We also want to explore strategies for integrating unbiased LTR methods into the training process within the current MLTR framework, further optimize and enhance the model's performance. We will also experiment with larger pre-trained language models as ranking models in our framework. We will study whether meta-learning can provide additional benefit on top of

transfer learning. Last but not the least, the proposed framework allows utilization of the meta-learning approach for more real-world applications, such as head-to-tail transfer learning, as well as the multi-task learning problem.

Acknowledgments

The authors would like to express their deepest gratitude to the team at Walmart Global Tech for providing them with the opportunity to undertake this research project. Their guidance, expertise, and resources were invaluable in helping the authors to complete this work.

References

[1] Aman Agarwal, Xuanhui Wang, Cheng Li, Michael Bendersky, and Marc Najork. 2019. Addressing Trust Bias for Unbiased Learning-to-Rank. In *WWW*. ACM, 4–14.

[2] Qingyao Ai, Keping Bi, Jiafeng Guo, and W. Bruce Croft. 2018. Learning a Deep Listwise Context Model for Ranking Refinement. In *SIGIR*. ACM, 135–144.

[3] Marcin Andrychowicz, Misha Denil, Sergio Gomez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. 2016. Learning to Learn by Gradient Descent by Gradient Descent. In *NIPS*, 3981–3989.

[4] Javed A. Aslam, Evangelos Kanoulas, Virgiliu Pavlu, Stefan Savev, and Emine Yilmaz. 2009. Document Selection Methodologies for Efficient and Effective Learning-to-Rank. In *SIGIR*. ACM, 468–475.

[5] Trapit Bansal, Rishikesh Jha, and Andrew McCallum. 2020. Learning to Few-Shot Learn across Diverse Natural Language Classification Tasks. In *COLING*. International Committee on Computational Linguistics, 5108–5123.

[6] Rukshan Batuwita and Vasile Palade. 2010. Efficient Resampling Methods for Training Support Vector Machines with Imbalanced Datasets. In *IJCNN*. IEEE, 1–8.

[7] Jonathan Baxter. 2000. A Model of Inductive Bias Learning. *J. Artif. Intell. Res.* 12 (2000), 149–198.

[8] Luiz Henrique Bonifacio, Hugo Queiroz Abonizio, Marzieh Fadaee, and Rodrigo Frassetto Nogueira. 2022. InPars: Data Augmentation for Information Retrieval Using Large Language Models. arXiv:2202.05144. Retrieved from https://arxiv.org/abs/2202.05144

[9] Chris J. C. Burges. 2010. *From RankNet to LambdaRank to LambdaMART: An Overview*. Technical Report.

[10] Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. 2005. Learning to Rank Using Gradient Descent. In *ICML*, Vol. 119, ACM, 89–96.

[11] Ethem F. Can, W. Bruce Croft, and R. Manmatha. 2014. Incorporating Query-Specific Feedback into Learning-to-Rank Models. In *SIGIR*. ACM, 1035–1038.

[12] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to Rank: From Pairwise Approach to Listwise Approach. In *ICML*, Vol. 227, ACM, 129–136.

[13] Vitor R. Carvalho, Jonathan L. Elsas, William W. Cohen, and Jaime G. Carbonell. 2008. A Meta-Learning Approach for Robust Rank Learning. In *SIGIR Workshop on Learning to Rank for Information Retrieval Singapore, Vol. 1.*

[14] Nitesh V. Chawla. 2003. C4. 5 and Imbalanced Data Sets: Investigating the Effect of Sampling Method, Probabilistic Estimate, and Decision Tree Structure. In *ICML*, Vol. 3, 66.

[15] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE: Synthetic Minority Over-Sampling Technique. *J. Artif. Intell. Res.* 16 (2002), 321–357.

[16] David Cossock and Tong Zhang. 2008. Statistical Analysis of Bayes Optimal Subset Ranking. *IEEE Trans. Inf. Theory* 54, 11 (2008), 5140–5154.

[17] Nick Craswell, Onno Zoeter, Michael J. Taylor, and Bill Ramsey. 2008. An Experimental Comparison of Click Position-Bias Models. In *WSDM*. ACM, 87–94.

[18] Yue Cui, Hao Sun, Yan Zhao, Hongzhi Yin, and Kai Zheng. 2022. Sequential-Knowledge-Aware Next POI Recommendation: A Meta-Learning Approach. *ACM Trans. Inf. Syst.* 40, 2 (2022), 23:1–23:22.

[19] Zhuyun Dai and Jamie Callan. 2019. Deeper Text Understanding for IR with Contextual Neural Language Modeling. In *SIGIR*. ACM, 985–988.

[20] Zhuyun Dai, Vincent Y. Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith B. Hall, and Ming-Wei Chang. 2023. Promptagator: Few-Shot Dense Retrieval from 8 Examples. In *ICLR*. OpenReview.net. Retrieved from https://openreview.net/forum?id=gmL46YMpu2J

[21] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. 2017. Neural Ranking Models with Weak Supervision. In *SIGIR*. ACM, 65–74.

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. Association for Computational Linguistics, 4171–4186.

[23] Thomas G. Dietterich. 2000. Ensemble Methods in Machine Learning. In *Multiple Classifier Systems, 1st International Workshop*, Lecture Notes in Computer Science, Vol. 1857, Springer, 1–15.
[24] Nat Dilokthanakul, Pedro A. M. Mediano, Marta Garnelo, Matthew C. H. Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. 2016. Deep Unsupervised Clustering with Gaussian Mixture Variational Autoencoders. arXiv:1611.02648. Retrieved from http://arxiv.org/abs/1611.02648
[25] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *ICML*, Vol. 70, PMLR, 1126–1135.
[26] Tianyu Gao, Xu Han, Hao Zhu, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2019. FewRel 2.0: Towards More Challenging Few-Shot Relation Classification. In *EMNLP-IJCNLP*. Association for Computational Linguistics, 6249–6254.
[27] Xiubo Geng, Tie-Yan Liu, Tao Qin, Andrew Arnold, Hang Li, and Heung-Yeung Shum. 2008. Query Dependent Ranking Using K-Nearest Neighbor. In *SIGIR*. ACM, 115–122.
[28] Phillip I. Good. 2005. *Resampling Methods: A Practical Guide to Data Analysis*. Birkhauser.
[29] Weiwei Guo, Xiaowei Liu, Sida Wang, Huiji Gao, Ananth Sankar, Zimeng Yang, Qi Guo, Liang Zhang, Bo Long, Bee-Chung Chen, et al. 2020. Detext: A Deep Text Ranking Framework with BERT. In *CIKM*, 2509–2516.
[30] Shashank Gupta, Philipp Hager, Jin Huang, Ali Vardasbi, and Harrie Oosterhuis. 2023. Recent Advances in the Foundations and Applications of Unbiased Learning to Rank. In *SIGIR*. ACM, 3440–3443.
[31] Hui Han, Wenyuan Wang, and Binghuan Mao. 2005. Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In *ICIC*, Vol. 3644, Springer, 878–887.
[32] Shuguang Han, Xuanhui Wang, Mike Bendersky, and Marc Najork. 2020. Learning-to-Rank with BERT in TF-Ranking. arXiv:2004.08476. Retrieved from https://arxiv.org/abs/2004.08476
[33] Haibo He, Yang Bai, Edwardo A. Garcia, and Shutao Li. 2008. ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning. In *IJCNN*. IEEE, 1322–1328.
[34] Katja Hofmann, Anne Schuth, Shimon Whiteson, and Maarten de Rijke. 2013. Reusing Historical Interaction Data for Faster Online Learning to Rank for IR. In *WSDM*. ACM, 183–192.
[35] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. 2013. Learning Deep Structured Semantic Models for Web Search Using Clickthrough Data. In *CIKM*. ACM, 2333–2338.
[36] Xiaowen Huang, Jitao Sang, Jian Yu, and Changsheng Xu. 2022. Learning to Learn a Cold-Start Sequential Recommender. *ACM Trans. Inf. Syst.* 40, 2 (2022), 30:1–30:25.
[37] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.* 20, 4 (2002), 422–446.
[38] Xiang Jiang, Mohammad Havaei, Gabriel Chartrand, Hassan Chouaib, Thomas Vincent, Andrew Jesson, Nicolas Chapados, and Stan Matwin. 2018. On the Importance of Attention in Meta-Learning for Few-Shot Text Classification. arXiv:1806.00852. Retrieved from http://arxiv.org/abs/1806.00852
[39] Thorsten Joachims. 2002. Optimizing Search Engines Using Clickthrough Data. In *SIGKDD*. ACM, 133–142.
[40] Thorsten Joachims, Laura A. Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2005. Accurately Interpreting Clickthrough Data as Implicit Feedback. In *SIGIR*. ACM, 154–161.
[41] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased Learning-to-Rank with Biased Feedback. In *WSDM*. ACM, 781–789.
[42] Evangelos Kanoulas, Stefan Savev, Pavel Metrikov, Virgiliu Pavlu, and Javed A. Aslam. 2011. A Large-Scale Study of the Effect of Training Set Characteristics over Learning-to-Rank Algorithms. In *SIGIR*. ACM, 1243–1244.
[43] Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, and Sehee Chung. 2019. MeLU: Meta-Learned User Preference Estimator for Cold-Start Recommendation. In *SIGKDD*. ACM, 1073–1082.
[44] Tie-Yan Liu. 2009. Learning to Rank for Information Retrieval. *Found. Trends Inf. Retr.* 3, 3 (2009), 225–331.
[45] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. 2009. Exploratory Undersampling for Class-Imbalance Learning. *IEEE Trans. Syst. Man Cybern. Part B* 39, 2 (2009), 539–550.
[46] Claudio Lucchese, Franco Maria Nardini, Raffaele Perego, and Salvatore Trani. 2017. The Impact of Negative Samples on Learning to Rank. In *ICTIR (CEUR Workshop Proceedings, Vol. 2007)*, CEUR-WS.org. Retrieved from https://ceur-ws.org/Vol-2007/LEARNER2017_short_1.pdf
[47] Alessandro Magnani, Feng Liu, Suthee Chaidaroon, Sachin Yadav, Praveen Reddy Suram, Ajit Puthenputhussery, Sijie Chen, Min Xie, Anirudh Kashi, Tony Lee, et al. 2022. Semantic Retrieval at Walmart. In *SIGKDD*. ACM, 3495–3503.
[48] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian J. Goodfellow. 2015. Adversarial Autoencoders. arXiv:1511.05644. Retrieved from http://arxiv.org/abs/1511.05644
[49] Harrie Oosterhuis. 2021. Computationally Efficient Optimization of Plackett-Luce Ranking Models for Relevance and Fairness. In *SIGIR*. ACM, 1023–1032.
[50] Harrie Oosterhuis. 2022. Learning-to-Rank at the Speed of Sampling: Plackett-Luce Gradient Estimation with Minimal Computational Complexity. In *SIGIR*. ACM, 2266–2271.

[51] Harrie Oosterhuis. 2023. Doubly Robust Estimation for Correcting Position Bias in Click Feedback for Unbiased Learning to Rank. *ACM Trans. Inf. Syst.* 41, 3 (2023), 61:1–61:33. DOI: https://doi.org/10.1145/3569453
[52] Harrie Oosterhuis and Maarten de Rijke. 2018. Differentiable Unbiased Online Learning to Rank. In *CIKM*. ACM, 1293–1302.
[53] Harrie Oosterhuis and Maarten de Rijke. 2020. Policy-Aware Unbiased Learning to Rank for Top-k Rankings. In *SIGIR*. ACM, 489–498.
[54] Harrie Oosterhuis and Maarten de Rijke. 2021. Unifying Online and Counterfactual Learning to Rank: A Novel Counterfactual Estimator that Effectively Utilizes Online Interventions (Extended Abstract). In *IJCAI*. ijcai.org, 4809–4813.
[55] Harrie Oosterhuis, Anne Schuth, and Maarten de Rijke. 2016. Probabilistic Multileave Gradient Descent. In *ECIR*, Lecture Notes in Computer Science, Vol. 9626, Springer, 661–668.
[56] Zhiyuan Peng, Xuyang Wu, and Yi Fang. 2023. Soft Prompt Tuning for Augmenting Dense Retrieval with Large Language Models. arXiv:2307.08303. DOI: https://doi.org/10.48550/ARXIV.2307.08303
[57] Kun Qian and Zhou Yu. 2019. Domain Adaptive Dialog Generation via Meta Learning. In *ACL*. Association for Computational Linguistics, 2639–2649.
[58] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 Datasets. arXiv:1306.2597. Retrieved from https://arxiv.org/pdf/1306.2597
[59] Zhen Qin, Le Yan, Honglei Zhuang, Yi Tay, Rama Kumar Pasumarthi, Xuanhui Wang, Mike Bendersky, and Marc Najork. 2021. Are Neural Rankers Still Outperformed by Gradient Boosted Decision Trees? In *ICLR*. Retrieved from https://openreview.net/pdf?id=Ut1vF_q_vC
[60] Zi-Hao Qiu, Ying-Chun Jian, Qing-Guo Chen, and Lijun Zhang. 2021. Learning to Augment Imbalanced Data for Re-Ranking Models. In *CIKM*. ACM, 1478–1487.
[61] Filip Radlinski, Madhu Kurup, and Thorsten Joachims. 2008. How Does Clickthrough Data Reflect Retrieval Quality? In *CIKM*. ACM, 43–52.
[62] Paul Rosenbaum and Donald Rubin. 1983. The Central Role of the Propensity Score in Observational Studies for Causal Effects. *Biometrika* 70 (04 1983), 41–55. DOI: https://doi.org/10.1093/biomet/70.1.41
[63] Mark Sanderson. 2010. Test Collection Based Evaluation of Information Retrieval Systems. *Found. Trends Inf. Retr.* 4, 4 (2010), 247–375.
[64] Anne Schuth, Harrie Oosterhuis, Shimon Whiteson, and Maarten de Rijke. 2016. Multileave Gradient Descent for Fast Online Learning to Rank. In *WSDM*. ACM, 457–466.
[65] Shiliang Sun and David R. Hardoon. 2010. Active Learning with Extremely Sparse Labeled Examples. *Neurocomputing* 73, 16–18 (2010), 2980–2988.
[66] Si Sun, Yingzhuo Qian, Zhenghao Liu, Chenyan Xiong, Kaitao Zhang, Jie Bao, Zhiyuan Liu, and Paul Bennett. 2021. Few-Shot Text Ranking with Meta Adapted Synthetic Weak Supervision. In *IJCNLP*. Association for Computational Linguistics, 5030–5043.
[67] Aleksei Ustimenko and Liudmila Prokhorenkova. 2020. StochasticRank: Global Optimization of Scale-Free Discrete Functions. In *ICML*, Proceedings of Machine Learning Research, Vol. 119, PMLR, 9669–9679.
[68] Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. 2016. Learning to Rank with Selection Bias in Personal Search. In *SIGIR*. ACM, 115–124.
[69] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. 2018. Position Bias Estimation for Unbiased Learning to Rank in Personal Search. In *WSDM*. ACM, 610–618.
[70] Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. 2018. The Lambdaloss Framework for Ranking Metric Optimization. In *CIKM*, 1313–1322.
[71] Yuan Wang, Zhiqiang Tao, and Yi Fang. 2022. A Meta-Learning Approach to Fair Ranking. In *SIGIR*. ACM, 2539–2544.
[72] Bin Wu, Zaiqiao Meng, Qiang Zhang, and Shangsong Liang. 2022. Meta-Learning Helps Personalized Product Search. In *WWW*. Frédérique Laforest, Raphaël Troncy, Elena Simperl, Deepak Agarwal, Aristides Gionis, Ivan Herman, and Lionel Médini (Eds.), ACM, 2277–2287.
[73] Xuyang Wu, Alessandro Magnani, Suthee Chaidaroon, Ajit Puthenputhussery, Ciya Liao, and Yi Fang. 2022. A Multi-Task Learning Framework for Product Ranking with BERT. In *WWW*. ACM, 493–501.
[74] Rong Xiao, Jianhui Ji, Baoliang Cui, Haihong Tang, Wenwu Ou, Yanghua Xiao, Jiwei Tan, and Xuan Ju. 2019. Weakly Supervised Co-Training of Query Rewriting and Semantic Matching for E-Commerce. In *WSDM*, 402–410.
[75] Ji Yang, Xinyang Yi, Derek Zhiyuan Cheng, Lichan Hong, Yang Li, Simon Xiaoming Wang, Taibai Xu, and Ed H. Chi. 2020. Mixed Negative Sampling for Learning Two-Tower Neural Networks in Recommendations. In *WWW*. ACM/IW3C2, 441–447.
[76] Shaowei Yao, Jiwei Tan, Xi Chen, Keping Yang, Rong Xiao, Hongbo Deng, and Xiaojun Wan. 2021. Learning a Product Relevance Model from Click-Through Data in E-Commerce. In *WWW*, 2890–2899.

[77] Dawei Yin, Yuening Hu, Jiliang Tang, Tim Daly, Mianwei Zhou, Hua Ouyang, Jianhui Chen, Changsung Kang, Hongbo Deng, Chikashi Nobata, et al. 2016. Ranking Relevance in Yahoo Search. In *SIGKDD*, 323–332.

[78] Qian Yu and Wai Lam. 2019. Data Augmentation Based on Adversarial Autoencoder Handling Imbalance for Learning to Rank. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, Article 51, 411–418. DOI: https://doi.org/10.1609/aaai.v33i01.3301411

[79] Jing Yuan, Christian Geißler, Weijia Shao, Andreas Lommatzsch, and Brijnesh J. Jain. 2023. When Algorithm Selection Meets Bi-Linear Learning to Rank: Accuracy and Inference Time Trade Off with Candidates Expansion. *Int. J. Data Sci. Anal.* 16, 2 (2023), 173–189.

[80] Yisong Yue and Thorsten Joachims. 2009. Interactively Optimizing Information Retrieval Systems as a Dueling Bandits Problem. In *ICML*, ACM International Conference Proceeding Series, Vol. 382, ACM, 1201–1208.

[81] Alexey Zabashta, Ivan Smetannikov, and Andrey Filchenkov. 2015. Study on Meta-Learning Approach Application in Rank Aggregation Algorithm Selection. In *ECMLPKDD,* CEUR Workshop Proceedings, Vol. 1455, CEUR-WS.org, 115–116.

[82] Zhi-Hua Zhou, De-Chuan Zhan, and Qiang Yang. 2007. Semi-Supervised Learning with Very Few Labeled Training Examples. In *AAAI*. AAAI Press, 675–680.

[83] Barret Zoph and Quoc V. Le. 2017. Neural Architecture Search with Reinforcement Learning. In *ICLR*. OpenReview.net. Retrieved from https://openreview.net/forum?id=r1Ue8Hcxg

Received 8 May 2023; revised 21 August 2024; accepted 1 September 2024