S^{2}phere: Semi-Supervised Pre-training for Web Search over Heterogeneous Learning to Rank Data

Yuchen Li
Shanghai Jiao Tong University
Shanghai, China
yuchenli@sjtu.edu.cn

Haoyi Xiong[*]
Baidu Inc.
Beijing, China
haoyi.xiong.fr@ieee.org

Linghe Kong[*]
Shanghai Jiao Tong University
Shanghai, China
linghe.kong@sjtu.edu.cn

Qingzhong Wang
Baidu Inc.
Beijing, China
wangqingzhong@baidu.com

Shuaiqiang Wang
Baidu Inc.
Beijing, China
shqiang.wang@gmail.com

Guihai Chen
Shanghai Jiao Tong University
Shanghai, China
gchen@cs.sjtu.edu.cn

Dawei Yin
Baidu Inc.
Beijing, China
yindawei@acm.org

ABSTRACT

While Learning to Rank (LTR) models on top of transformers have been widely adopted to achieve decent performance, it is still challenging to train the model with sufficient data as only an extremely small number of query-webpage pairs could be annotated versus trillions of webpages available online and billions of web search queries everyday. In the meanwhile, industry research communities have released a number of open-source LTR datasets with well annotations but incorporating different designs of LTR features/labels (i.e., heterogeneous domains). In this work, inspired by the recent progress in pre-training transformers for performance advantages, we study the problem of pre-training LTR models using both labeled and unlabeled samples, especially we focus on the use of well-annotated samples in heterogeneous open-source LTR datasets to boost the performance of pre-training. Hereby, we propose **S^{2}phere**—*Semi-Supervised Pre-training with Heterogeneous LTR data* strategies for LTR models using both unlabeled and labeled query-webpage pairs across heterogeneous LTR datasets. **S^{2}phere** consists of a three-step approach: *(1) Semi-supervised Feature Extraction Pre-training via Perturbed Contrastive Loss*, *(2) Cross-domain Ranker Pre-training over Heterogeneous LTR Datasets* and *(3) End-to-end LTR Fine-tuning via Modular Network Composition*. Specifically, given an LTR model composed of a backbone (the feature extractor), a neck (the module to reason the orders) and a head (the predictor of ranking scores), **S^{2}phere** uses unlabeled/labeled data from the search engine to pre-train the backbone in Step (1) via semi-supervised learning; then Step (2) incorporates multiple open-source heterogeneous LTR datasets to improve pre-training of the neck module as shared parameters of cross-domain learning; and finally, **S^{2}phere** in Step (3) composes the backbone and neck with a randomly-initialized head into a whole LTR model and fine-tunes the model using search engine data with various learning strategies. Extensive experiments have been done with both offline experiments and online A/B Test on top of Baidu search engine. The comparisons against numbers of baseline algorithms confirmed the advantages of **S^{2}phere** in producing high-performance LTR models for web-scale search.

CCS CONCEPTS

• **Information systems** → **Learning to rank**; Web search engines.

KEYWORDS

Learning to Rank, Large-scale Web Search, Pre-training, Heterogeneous Data

ACM Reference Format:
Yuchen Li, Haoyi Xiong, Linghe Kong, Qingzhong Wang, Shuaiqiang Wang, Guihai Chen, and Dawei Yin. 2023. **S^{2}phere**: Semi-Supervised Pre-training for Web Search over Heterogeneous Learning to Rank Data. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3580305.3599935

1 INTRODUCTION

Learning to Rank (LTR) models on top of deep neural networks [24, 31], such as *transformers* [36], have been widely used in web search and achieved good performance in webpages ranking, such as higher Hit Rate (HR), Discounted Cumulative Gain [14] (DCG) and their derivatives. Given trillions of webpages available online and billions of queries for web search, a search engine usually employs annotators, who label the relevance score for a number of query-webpage pairs, to facilitate LTR model training.

While annotating massive collections of query-webpage pairs with relevance scores is resource-consuming for a search engine [29], the leading players in the community, such as Yahoo!, Microsoft Bing and Baidu, have released several representative open-source LTR datasets [6, 29, 44], incorporating well-annotated query-webpage pairs with precise but using different features (i.e., variables designed for ranking) and labels (i.e., scales of relevance scores for webpages ranking). These datasets were collected from different

[*] Corresponding authors are Linghe Kong and Haoyi Xiong.

KDD '23, August 6–10, 2023, Long Beach, CA, USA.

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599935

search engines but all for similar LTR tasks. We call these datasets as heterogeneous LTR datasets in this work. As it has become a common practice to leverage large-scale open-source datasets to pre-train models, especially transformers, for better performance in computer vision [13] and natural language processing [11] domains, we are wondering *whether we can incorporate the open-source LTR datasets into pre-training LTR models for enhanced ranking performance despite the heterogeneity in their features and labels design.*

In this work, inspired by the recent progress in pre-training transformers for performance advantages, we study the problem of pre-training LTR models for a search engine using a huge amount of unlabeled query-webpage pairs with few labeled samples. We especially focus on the use of well-annotated samples in multiple heterogeneous ***open-source LTR datasets***, where the feature and sample space of every dataset are different from each other. To achieve the goal, there remain several non-trivial issues as follows:

- *LTR representation pre-training over heterogeneous features.* The goal of pre-training over open-source heterogeneous LTR data is to learn the *"logic and reasoning"* of webpage ranking from these datasets. However, the feature sets used in these LTR datasets are different, and it is difficult to train a unified extractor working on all these feature sets. In this way, a unified set of representations should be learned from heterogeneous datasets for LTR, and then the *"logic and reasoning"* of webpage ranking could work on top of the unified representations despite feature heterogeneity. *Thus, there needs to pre-train the feature extractor and the module [12] for ranking separately, so as to extract features from search engine data while inheriting the ranking capacity from massive open-source data, while ensuring the feature extractor and module work together via unified representation in the final LTR model.*
- *LTR representation pre-training losses and objectives.* In computer vision and natural language processing practice, pre-training is commonly formulated as a supervised and/or self-supervised learning procedure, where the supervised learning relies on intensive data annotation and self-supervised learning needs well-designed losses to feedback models with meaningful supervision signals, such as reconstruction loss and contrastive loss. In our LTR settings, where labels for most query-webpage pairs at the search engine are not available, supervised pre-training is with limited capacity; in the meanwhile, the existing work that designs reconstruction losses or contrastive losses for self-supervision with LTR data is quite few. *Thus, semi-supervised learning is desired to pre-train the model with labeled data, unlabeled data (e.g., with pseudo labels), and novel self-supervision losses.*

To address the above issues, we modularize the LTR models into three components in a row—i.e., a feature extractor (backbone) learned from the search engine data to generate the unified representation, a ranker (neck) learned from both search engine and open-source data via the unified representation, and a scorer (head) that predicts ranking scores using the outputs of the ranking module. Specifically, we propose **S^{2}phere**—*Semi-Supervised Pre-training with Heterogeneous LTR data* strategies for LTR models using both unlabeled and labeled pairs across heterogeneous LTR datasets.

Specifically, on top of the backbone, neck and head, **S^{2}phere** consists of a three-step approach: *(1) Semi-supervised Feature Extraction Pre-training via Perturbed Contrastive Loss, (2) Cross-domain Ranker Pre-training over Heterogeneous LTR Datasets* and *(3) End-to-end LTR Fine-tuning via Modular Network Composition.* **S^{2}phere** pre-trains the backbone in Step (1) via semi-supervised learning using unlabeled/labeled data from the search engine. Then **S^{2}phere** in Step (2) incorporates multiple open-source heterogeneous LTR datasets to train multiple LTR models and obtains the neck as a set of *shared parameters* in these networks. Finally, **S^{2}phere** composes the backbone, neck and head into a whole LTR model. In Step (3), it randomly re-initializes the head and fine-tunes the model using search engine data via LTR loss using a warm-up fine-tuning strategy, where the warm-up strategy first freezes the weights of the neck to let the backbone and head adapt the input and output of the neck in first several epochs, then leverages end-to-end training with whole network and updates all weights to fit the data. In summary, the main contributions are summarized as follows:

- We study the problem of pre-training LTR models using both labeled/unlabeled data collected at the search engine and incorporating the open-source heterogeneous LTR datasets to boost the performance of webpage ranking for search. To the best of our knowledge, it is the first work in pre-training LTR models that incorporates both search engine data and open-source data by addressing data heterogeneity, modular networks and semi-supervised learning issues.
- We propose **S^{2}phere** consisting of three steps to pre-train the feature extractor and the ranking module separately using search engine and open-source data accordingly via various losses in a semi-supervised manner. Then **S^{2}phere** composes the feature extractor (backbone), ranker module (neck) and scorer module (head) to predict ranking scores together to form the end-to-end LTR model and fine-tune the model on the search engine data using warm-up strategies.
- We carry out extensive experiments with both offline experiments and online A/B tests on top of Baidu Search engine to verify the effectiveness of **S^{2}phere**. We compare **S^{2}phere** against a number of state-of-the-art baseline algorithms. Specifically, we test the performance of these algorithms with four ratios of labeled data, i.e., 5%, 10%, 15% and 20% of query-webpage pairs at the search engine side labeled with relevance scores. The comparisons confirmed the advantages of **S^{2}phere** in producing high-performance LTR models for web-scale search. In offline comparisons, **S^{2}phere** outperforms competitor systems with 1.03%~4.09% on *NDCG*@4. In the A/B Test with 5% real Baidu web search traffics, we observe the advantage of **S^{2}phere** which achieves $\Delta NDCG@4$ = 0.09%~0.75% in parallel comparisons.

2 S^{2}PHERE DESIGN AND ALGORITHM

In this section, we first formulate the research problem of LTR, then detail the **S^{2}phere** design and algorithm.

2.1 Problem Formulation

In this section, we introduce the formalization of the LTR task. Given a set of search queries $Q = \{q_1, q_2, \dots\}$ and all archived webpages $\mathcal{D} = \{d_1, d_2, \dots\}$, for each query $q_i \in Q$, the search engine could retrieve a set of relevant webpages denoted as $D_i = \{d_1^i, d_2^i, \dots\} \subset \mathcal{D}$. By employing annotation, a collection of ranking scores $\boldsymbol{y}_i = \{y_1^i, y_2^i, \dots\}$ can be associated with q_i, effectively

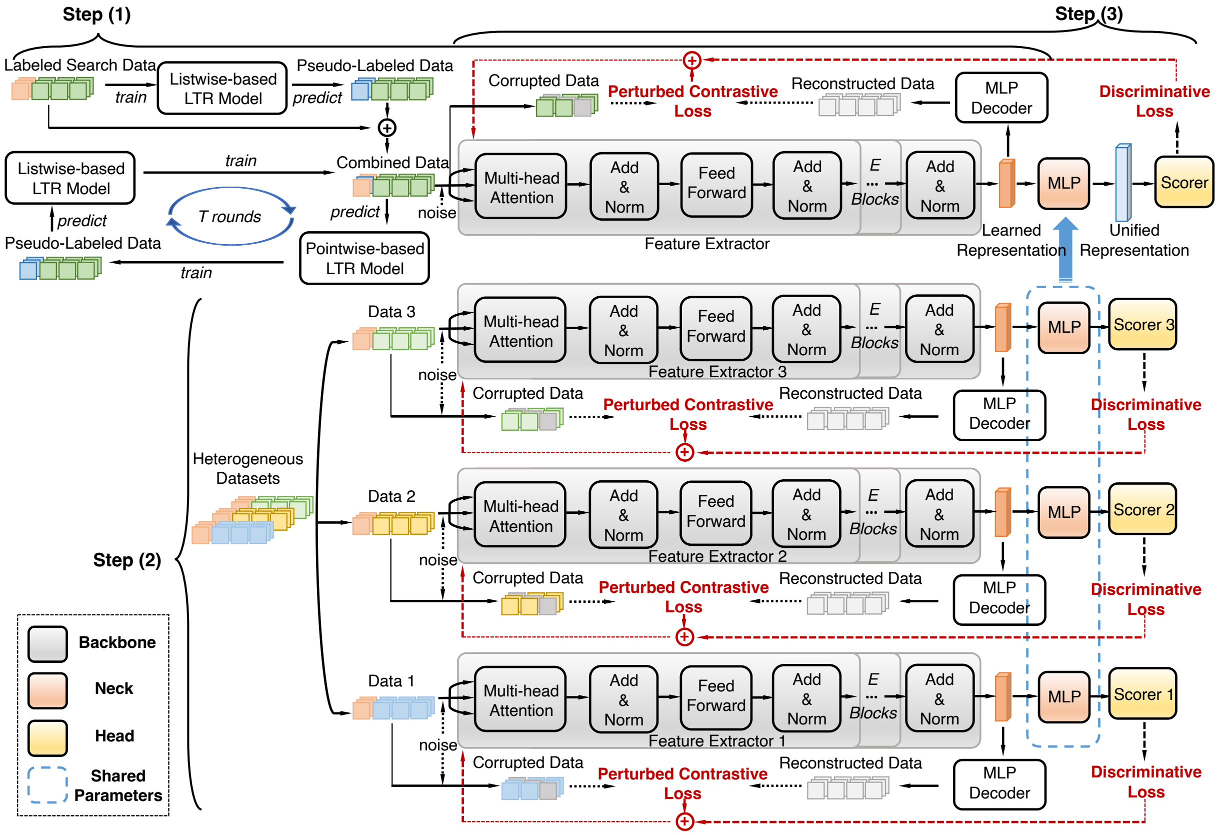


Figure 1: S^{2}phere in three steps: (1) *Semi-supervised Pre-training via Perturbed Contrastive Loss*, (2) *Cross-domain Ranker Pre-training over Heterogeneous LTR Datasets*, and (3) *End-to-end LTR Fine-tuning via Modular Network Composition.*

capturing the relevance of each webpage $d_j^i \in D_i$ to the search query q_i. We adhere to the settings established in [29], where we adopt a relevance label scaling approach from 0 to 4 to indicate varying levels of relevance. Note that the features of search queries and webpages are extracted by a pre-trained *language model* and constructed through Baidu's *retrieval system* [23]. All these raw features are used as the inputs of webpage ranking.

Given the set of query-webpage pairs with relevance label annotations, we utilize a set of triples (i.e., $\mathcal{T}^L = \{(q_1, D_1, \boldsymbol{y}_1),\ (q_2, D_2, \boldsymbol{y}_2),$ $(q_3,\ D_3, \boldsymbol{y}_3), \dots\}$) to represent. This work aims to obtain an LTR scoring function $f: \mathcal{Q} \times \mathcal{D} \to [0, 4]$, where the learning objective of LTR is redefined to learn a scoring function f which minimizes the ranking loss as

$$\mathcal{L} = \frac{1}{|\mathcal{T}^L|}\sum_{i=1}^{|\mathcal{T}^L|}\left(\frac{1}{|D_i|}\sum_{j=1}^{|D_i|}\ell(\boldsymbol{y}_j^i, f(q^i, d_j^i))\right), \qquad (1)$$

where ℓ denotes the ranking loss between the predicted relevance of webpage d_j^i for query q_i and the corresponding ground truth relevance label $\boldsymbol{y}_j^i$. Attributing the outstanding scalability, **S^{2}phere** is applicable with standard loss functions (i.e., pointwise, pairwise, and listwise). Given that annotators have limitations in labeling query-webpage pairs due to cost and time constraints, incorporating unlabeled query-webpage pairs becomes crucial in LTR. Therefore, to formulate the semi-supervised LTR setting, we introduce a set of unlabeled query-webpage pairs denoted as $\mathcal{T}^U = \{(q_1', D_1'),\ (q_2', D_2'), \dots\} \subset \mathcal{Q} \times 2^{\mathcal{D}}$, where the number of instances in $\mathcal{T}^U$ greatly exceeds the number of instances in $\mathcal{T}^L$.

The research goal is to optimize above LTR problem by advancing representation learning on top of the *legacy LTR system* [45] and deploy the solution in the search engine introduced in Section 4.

2.2 Overall Framework Design

As illustrated in Figure 1, **S^{2}phere** consists of three steps: (1) *Semi-supervised Feature Extraction Pre-training via Perturbed Contrastive Loss*, (2) *Cross-domain Ranker Pre-training over Heterogeneous LTR Datasets*, and (3) *End-to-end LTR Fine-tuning via Modular Network Composition*. Specifically, in *Step (1)*, **S^{2}phere** first generates high-quality pseudo labels for each unlabeled query-webpage pair through semi-supervised learning of multiple/diverse LTR models based on various ranking losses. Then, **S^{2}phere** learns generalizable representations with the transformer network (i.e., the backbone), which uses the contrastive loss for the reconstruction of perturbed data. In *Step (2)*, given the generalizable representations of query-webpage pairs, **S^{2}phere** leverages a Multi-Layer Perception (MLP) mechanism (i.e., the neck) to conduct a cross-domain pre-trained ranker, which utilizes discriminative loss to execute the ranking task on heterogeneous LTR datasets. Eventually, in *Step (3)*, **S^{2}phere** combines the transformer network (backbone), the pre-trained MLP mechanism (neck), and a scorer (i.e., the head) to conduct an end-to-end modular LTR network and fine-tunes the modular network on the pseudo-labeled dataset collected from Baidu Search.

2.3 Semi-supervised Feature Extraction Pre-training via Perturbed Contrastive Loss

This step consists of two algorithms: *Self-tuned Label Propagation* and *Feature Extraction Pre-training via Perturbed Contrastive Loss*.

2.3.1 *Self-tuned Label Propagation*.

Given the overall sets of queries Q and the set of all webpages $\mathcal{D}$, **S^{2}phere** first obtains each possible query-webpage pair from the both datasets, denoted as (q_i, d_i^j) for $\forall q_i \in Q$ and $\forall d_i^j \in D_i \subset \mathcal{D}$, i.e., the j^{th} webpage retrieved for the i^{th} query. For each query-webpage pair (q_i, d_i^j), **S^{2}phere** further extracts an m-dimensional feature vector $\boldsymbol{x}_{i,j}$ representing the features of the j^{th} webpage under the i^{th} query. Then, the labeled and unlabeled sets of feature vectors can be presented as $\mathcal{X}^L = \{(\boldsymbol{x}_{i,j}, \boldsymbol{y}_j^i) | \forall (q_i, D_i, \boldsymbol{y}) \in \mathcal{T}^L \text{ and } \forall d_j^i \in D_i\}$ and $\mathcal{X}^U = \{\boldsymbol{x}_{i,j} | \forall (q_i, D_i) \in \mathcal{T}^U\}$. **S^{2}phere** further takes self-tuning approach [21] to propagate labels from annotated query-webpage pairs to those unlabeled ones.

Specifically, **S^{2}phere** first trains an LTR model based on $\mathcal{X}^L$ only and then predicts the ranking scores of unlabeled pairs in $\mathcal{X}^U$ using the trained model. While **S^{2}phere** setting the predicted scores as pseudo labels for $\mathcal{X}^U$, it repeats training a new/updated LTR model using both labeled/pseudo-labeled data and predicts new/updated scores for unlabeled data with the new/updated model accordingly. After T repeats of self-tuning, **S^{2}phere** assigns every unlabeled query-webpage pair in $\mathcal{X}^U$ an accurate estimate of the ranking score, and then fuses them with $\mathcal{X}^L$ into a new labeled set $\mathcal{X}^C$.

2.3.2 *Feature Extraction Pre-training via Perturbed Contrastive Loss*.

As shown in Figure 1, **S^{2}phere** leverages self-attentive transformers to learn generalizable representations of query-webpage pairs via perturbed contrastive loss in *Feature Extraction Pre-training via Perturbed Contrastive Loss*.

Firstly, given an m-dimensional feature vector $\tilde{\boldsymbol{x}}_{i,j}$ of a query-webpage pair $(\tilde{\boldsymbol{x}}_{i,j}, \boldsymbol{y}_j^i) \in \mathcal{X}^C$, **S^{2}phere** leverages a self-attentive encoder to learn a generalizable representation $\boldsymbol{z}_{i,j}$. **S^{2}phere** (1) is fed a vector into a fully connected layer and produces a hidden representation. Later, **S^{2}phere** (2) feeds the hidden representation into a self-attentive autoencoder, which consists of E encoder blocks of transformer. Specifically, each encoder block incorporates a multi-head attention layer and a feed-forward layer, both followed by layer normalization. **S^{2}phere** (3) generates the learned representation $\boldsymbol{z}_{i,j}$ from the last encoder block. For each original feature vector $\tilde{\boldsymbol{x}}_{i,j}$, the whole training process can be formulated as $\boldsymbol{z}_{i,j} = f_{\tilde{\theta}}(\tilde{\boldsymbol{x}}_{i,j})$, where $\tilde{\theta}$ is the set of parameters.

Then, **S^{2}phere** utilizes a simple yet useful MLP mechanism for the reconstruction task. Specifically, for each representation $\boldsymbol{z}_{i,j}$ produced from the self-attentive autoencoder, **S^{2}phere** leverages the MLP mechanism to map $\boldsymbol{z}_{i,j}$ to a generalizable representation $\boldsymbol{z}'_{i,j}$, which has the same dimension with the original feature vector $\tilde{\boldsymbol{x}}_{i,j}$. The whole training process can be formulated as $\boldsymbol{z}'_{i,j} = g_{\theta'}(\boldsymbol{z}_{i,j})$, where the θ' is the set of parameters.

Eventually, **S^{2}phere** jointly optimizes the parameter sets $\tilde{\theta}$ and θ' to minimize the perturbed contrastive loss as

$$\mathcal{L}_{Contra} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\left(\frac{1}{|D_i|}\sum_{j=1}^{|D_i|}\ell_{Contra}\left(\tilde{\boldsymbol{x}}_{i,j}, \boldsymbol{z}'_{i,j}\right)\right) \tag{2}$$

where ℓ_{Contra} is the squared error, which could be defined as

$$\ell_{Contra}\left(\tilde{\boldsymbol{x}}_{i,j}, \boldsymbol{z}'_{i,j}\right) = \|\tilde{\boldsymbol{x}}_{i,j} - \boldsymbol{z}'_{i,j}\|^2. \tag{3}$$

2.4 Cross-domain Ranker Pre-training over Heterogeneous LTR Datasets

As presented in Figure 1, given both search data and open-source data, **S^{2}phere** in Step (2) trains multiple LTR models using heterogeneous LTR datasets via discriminative (LTR) learning, where a set of parameters (in MLP) is shared as the neck of these models and is expected to obtain the capability of ranking from all these datasets.

Given the learned representation $\boldsymbol{z}_{i,j}$ generated from *Feature Extraction Pre-training via Perturbed Contrastive Loss*, **S^{2}phere** adopts an MLP model with a fully-connected layer to calculate the predicted score $\boldsymbol{s}_{i,j}$. The whole process can be formulated as $s_{i,j} = f_\theta(\boldsymbol{z}_{i,j})$, where θ is the set of discriminative parameters. Against the ground truth, **S^{2}phere** leverages the discriminative loss function, which can be defined as

$$\mathcal{L}_{Disc} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\left(\frac{1}{|D_i|}\sum_{j=1}^{|D_i|}\ell_{LTR}\left(\boldsymbol{y}_j^i, s_{i,j}\right)\right) \tag{4}$$

where ℓ_{LTR} represents the standard LTR loss function (i.e., pointwise, pairwise and listwise).

To accomplish both perturbed contrastive tasks (*Step (1)*) and discriminative (*Step (2)*) simultaneously, **S^{2}phere** jointly optimizes the perturbed contrastive loss $\mathcal{L}_{Contra}$ and the discriminative loss $\mathcal{L}_{Disc}$ as the final loss function as

$$\mathcal{L}_{Final} = \alpha\mathcal{L}_{Contra} + (1-\alpha)\mathcal{L}_{Disc}, \tag{5}$$

where the loss weights $\alpha \in [0, 1]$ is a hyper-parameter to balance two loss functions.

Given heterogeneous LTR datasets $\{\mathcal{X}_1^C, \ldots, \mathcal{X}_N^C\}$, the goal of **S^{2}phere** is to obtain the pre-trained neck, which gains the reasoning capacity. In order to get the pre-trained neck on heterogeneous LTR datasets, **S^{2}phere** adopts the pre-training procedure shown in Algorithm 1.

2.5 End-to-end LTR Fine-tuning via Modular Network Composition

Given the pre-trained neck utilizing the above two steps, **S^{2}phere** combines a backbone, the pre-trained neck, and a head to conduct a modular network for the downstream LTR task. Finally, **S^{2}phere** fine-tunes the modular network to execute the LTR task on the pseudo-labeled search dataset, which accomplishes the end-to-end LTR fine-tuning.

3 TRAINING PARADIGM

The conventional training paradigm of pre-training and fine-tuning has been routinely leveraged and obtained significant achievements

Algorithm 1: Pre-train $\mathbf{S^2}$phere on Heterogeneous LTR Datasets

```
Input: the number of epochs $eps$; hyper-parameters for final loss $\alpha, \beta$; heterogeneous LTR datasets $\{\mathcal{X}_1^C, \dots, \mathcal{X}_N^C\}$; backbones $\{B_1, \dots, B_N\}$; heads $\{H_1, \dots, H_N\}$
Output: $N_{pre}$ — the pre-trained neck
for $i \in \{1, \dots, N\}$ do
    Choose the LTR dataset $\mathcal{X}_i^C$;
    for $ep \in \{1, \dots, eps\}$ do
        Choose backbone $B_i$ and $H_i$;
        Evaluate contrastive loss $\mathcal{L}_{Contra}$;
        Evaluate discriminative loss $\mathcal{L}_{Disc}$;
        Calculate final loss $\mathcal{L}_{Final} = \alpha\mathcal{L}_{Contra} + \beta\mathcal{L}_{Disc}$;
    end for
    Update $N_{pre}$;
end for
return $N_{pre}$;
```

across many tasks. Nevertheless, this paradigm has not been demonstrated to be appropriate for pre-training a reasoning neck on a large-scale ranking system for web search. Therefore, we propose a novel training paradigm for $\mathbf{S^2}$**phere** to fit the ranking task at Baidu Search. Specifically, the training paradigm takes a three-stage strategy: (1) *Pre-training on Heterogeneous Datasets*, (2) *Warm-up Fine-tuning*, and (3) *Post-fine-tuning*. According to practical experience in deploying the ranking system for large-scale search engines, the training paradigm is designed to be more effective and efficient.

Stage 1: Pre-training on Heterogeneous Datasets. As referred to in Section 2, we leverage $\mathbf{S^2}$**phere** to accomplish *pre-training on heterogeneous datasets*. $\mathbf{S^2}$**phere** is pre-trained on heterogeneous datasets, i.e. MSLR-Web30K, MQ2007 and MQ2008, to learn the reasoning capability for the neck by cross-domain ranking-task learning. Specifically, we first train $\mathbf{S^2}$**phere** on MSLR-Web30K and obtain the pre-trained neck. Then, the pre-trained neck combines a backbone and a head to compose a new modular network, which is trained on MQ2007 to execute the pre-training procedure. Eventually, we pre-train a new modular network with the trained neck on MQ2008. In particular, three pre-trained datasets contain massively heterogeneous features which could enhance the reasoning capability of $\mathbf{S^2}$**phere**.

Stage 2: Warm-up Fine-tuning. In this stage, we propose *Warm-up Fine-tuning* to initialize the modular network. Specifically, following *Pre-training on Heterogeneous Datasets*, we first add a fully-connected layer to the head and tail of the pre-trained neck, respectively. Next, we freeze the weights of layers in the pre-trained neck and fine-tune the rest parts of $\mathbf{S^2}$**phere** in the first several epochs on the dataset collected from Baidu Search. In this way, $\mathbf{S^2}$**phere** could be initialized in a low-cost and rapid way instead of jointly fine-tuning the whole modular network.

Stage 3: Post-fine-tuning. In the final stage, given the warm-up fine-tuned modular network, we jointly fine-tune the whole modules on the dataset collected from Baidu Search. In particular, we leverage the warm-up fine-tuned $\mathbf{S^2}$**phere** to simultaneously accomplish the discriminative learning (LTR) and perturbed contrastive learning tasks. After stage *Post-fine-tuning*, we finish the proposed training paradigm and accomplish the end-to-end pre-training and fine-tuning for $\mathbf{S^2}$**phere**.

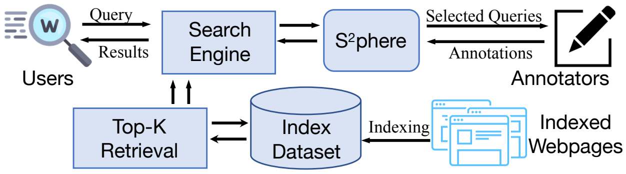


Figure 2: Deployment of $\mathbf{S^2}$phere at Baidu Search.

4 DEPLOYMENT OF $\mathbf{S^2}$PHERE

In this section, we present the deployment details of $\mathbf{S^2}$**phere** in the context of Baidu Search. As illustrated in Figure 2, Baidu Search is essential with three stages as follows: (1) *Webpage Collection*, (2) *Webpage Indexing* and (3) *Retrieval and Ranking*.

Webpage Collection. To handle the vast number of webpages available on the web, Baidu Search employs an efficient and powerful *Web Crawler*. Web Crawler is responsible for collecting and downloading webpages. The process begins with Web Crawler scanning a list of links to identify new webpages and those that have been updated. It selectively stores the valid links containing the desired content in a large downloading list. Then, based on the real-time web traffic of Baidu Search, Web Crawler initiates the downloading process for the websites present in the list.

Webpage Indexing. Baidu Search efficiently stores the vast amount of downloaded webpages in distributed archival storage systems, utilizing the capabilities of *Fatman* [26]. To achieve high-performance search, Baidu Search builds efficient indices by employing *DirectLoad* [27]. *Fatman* leverages elastic resources, such as underutilized servers and temporarily available storage, across multiple regional data centers of Baidu, significantly reducing storage costs. In parallel, *DirectLoad* ensures load balancing of indexing tasks across these data centers.

Retrieval and Ranking. To facilitate the content retrieval by search queries, Baidu adopts an extremely large-scale Chinese pre-trained language model [32] for feature extraction and builds up an ERNIE-based semantic content retrieval system [23]. Given the retrieved contents for a query, the Baidu search engine leverages LTR systems, such as $\mathbf{S^2}$**phere** or the *legacy system* [45], for webpage ranking. Note that the raw features of webpages and search queries for LTR (introduced in Section 2.1) are extracted by the language model and the retrieval system [23].

- *Offline Training.* In terms of webpages ranking in online settings, following the proposed training paradigm mentioned in Section 3, $\mathbf{S^2}$**phere** maintains an end-to-end fine-tuned LTR model via pre-training on heterogeneous LTR datasets and fine-tuning on the dataset streamed and annotated by Baidu Search.
- *Online Serving.* For online inference, $\mathbf{S^2}$**phere** adopts the fine-tuned LTR model to serve the online ranking tasks. Given a query and a webpage for online ranking, $\mathbf{S^2}$**phere** first transforms them into an extracted query-webpage feature. Then, $\mathbf{S^2}$**phere** picks up their feature representations and passes them to the fine-tuned LTR model for inference in a fast manner at Baidu Search. Note that, to further accelerate online DNN inference procedures, features/embedding of queries/webpages and the weights of DNN

models might be cached or re-stored over heterogeneous memory/storage devices [22] for fast look-ups, reads and updates.

5 EXPERIMENTS

To investigate the effectiveness of **S^{2}phere**, we conduct extensive offline and online experiments on a large-scale search engine. In this section, we first introduce the experimental details. Then, we present the result of offline and online experiments to prove the effectiveness of our proposed model.

5.1 Datasets

5.1.1 *Heterogeneous Open-source LTR Datasets.*

We leverage three standard, publicly available datasets for pre-training.

- **MSLR-Web30K** contains 30,000 queries. Each query-webpage pair is represented as a 136-dimensional feature vector.
- **MQ2007** contains 1,692 queries. Each query-webpage pair of MQ2007 is represented as a 46-dimensional feature vector.
- **MQ2008** contains 784 queries. Similar to MQ2007, each query-webpage pair of MQ2008 is represented as a 46-dimensional feature vector.

Each query-webpage pair of the above three datasets is associated with a relevance label on a scale from 0 to 4 to represent levels of relevance from irrelevant to perfectly relevant. In our experiments, we perform the five-fold cross-validation [29] and report the average results across five folds.

5.1.2 *Search Dataset.*

We collect the dataset with 65,000 queries and over 3,340,000 query-webpage pairs from Baidu Search. For each query, we collect webpages from each stage of the search pipeline to ensure the diversity of the dataset. The dataset is annotated on our crowdsourcing platform, where a group of professionals annotate each query-webpage pair with an integer score that ranges from 0 (bad) to 4 (perfect). Each query-webpage pair is also represented as a real-valued feature vector. We randomly split the dataset into *training set* (39,000 queries), *validation set* (13,000 queries), and test set (13,000 queries). In our experiments, features are standardized before feeding them into LTR models.

5.2 Evaluation Methdology

Normalized Discounted Cumulative Gain (NDCG) [15] is a widely employed metric for assessing relevance in ad-hoc search engine contexts. It provides a comprehensive evaluation of the quality of a ranking list generated by an LTR model for a given query and its associated webpages. The LTR model predicts scores for each webpage and generates the ranking list by sorting the scores in descending order, considering the graded relevance of the webpages. Additionally, the value of NDCG ranges from $[0, 1]$, and a higher NDCG@N indicates a better LTR model. *In this work, we consider the NDCG of the top 4 and 10 results (i.e., NDCG@4 and NDCG@10) for business and research purposes.*

Positive-Negative Ratio (PNR) is a widely used pairwise metric for assessing the performance of search relevance in the industry. Given a query q and its associated ranked webpages D_q, PNR can be defined as the ratio of concordant pairs to discordant pairs as

$$PNR = \frac{\sum_{d_i,d_j \in D_q} \mathbf{1}\left\{y_i > y_j\right\} \cdot \mathbf{1}\left\{f\left(q, d_i\right) > f\left(q, d_j\right)\right\}}{\sum_{d_m,d_n \in D_q} \mathbf{1}\left\{y_m > y_n\right\} \cdot \mathbf{1}\left\{f\left(q, d_m\right) < f\left(q, d_n\right)\right\}}, \quad (6)$$

where $\mathbf{1}\{x > y\}$ is an indicator function (i.e., $\mathbf{1}\{x > y\} = 1$ if $x > y$, and 0 otherwise). PNR evaluates the consistency between the ground truth and the ranking score. *In our offline experiments, we report the average values over all test queries.*

Interleaving [9] is a widely used metric for evaluating the performance of an industrial search engine. In interleaved comparison, two results generated from different systems are delivered to users, whose click-through actions would be attributed to the system that delivers the corresponding results. Specifically, the gain of the new system A over the base system B could be formalized as

$$\Delta_{AB} = \frac{wins(A) + 0.5 \times ties(A, B)}{wins(A) + wins(B) + ties(A, B)} - 0.5, \quad (7)$$

where $wins(A)$ (or $wins(B)$) is a counter to calculate the number of times the user clicks the result generated from system A (or B), and $ties(A, B)$ is otherwise increased by 1. Therefore, $\Delta_{AB} > 0$ indicates that A is superior to B.

Good vs. Same vs. Bad (GSB) [41] is an online pairwise metric evaluated by professional annotators. In manual comparison, two results produced by the new system and the base system are provided to human experts that are required to judge which result is better. Specifically, GSB could be computed as

$$\Delta\text{GSB} = \frac{\#\text{Good} - \#\text{Bad}}{\#\text{Good} + \#\text{Same} + \#\text{Bad}}, \quad (8)$$

where #Good (or #Bad) is the number of results generated from the new system better (or worse) than the base system, and #Same indicates two results are equally good or bad. *In our online evaluation, we conduct balanced interleaving and manual evaluation to compare two online systems side-by-side.*

5.3 Loss Function and Competitor Systems

To evaluate **S^{2}phere** comprehensively, we adopt different state-of-the-art ranking losses as follows:

- **Root Mean Square Error (RMSE)** is a commonly employed *pointwise* loss of predicated relevancy.
- **RankNet** [4] and **LambdaRank** [4] are two popular *pairwise* losses for LTR tasks both in research and industry.
- **ListNet** [5] and **ListMLE** [38] are two *listwise* losses, which optimize the agreement between the prediction and ground truth.
- **ApproxNDCG** [28]and **NeuralNDCG** [25] are also two *listwise* losses that directly optimize the evaluation metric (i.e., $NDCG$).

Regarding the ranking model, we compare **S^{2}phere** with the state-of-the-art ranking model as follows:

- **MLP** refers to a popular ranking model and has been extensively employed in the industry and research.
- **Context-Aware Ranker (CAR)** [24] refers to a ranking model based on transformer architecture, which inputs raw feature vectors of items in the same list and outputs real-world scores.
- **XGBoost** [8] refers to a *Gradient Boosting Decision Tree* (GBDT)-based ranking model with a pairwise loss function.

Table 1: Offline comparative results on *NDCG*@4 **and** *NDCG*@10 **under various ratios of labeled data.**

Model	5%		10%		15%		20%	
	NDCG@4	NDCG@10	NDCG@4	NDCG@10	NDCG@4	NDCG@10	NDCG@4	NDCG@10
RMSE	49.09 ± 0.12	52.80 ± 0.43	53.48 ± 0.15	57.32 ± 0.26	55.76 ± 0.09	59.82 ± 0.34	57.78 ± 0.25	63.18 ± 0.17
RankNet	48.76 ± 0.27	52.45 ± 0.19	53.02 ± 0.35	56.90 ± 0.08	55.52 ± 0.14	59.26 ± 0.34	57.56 ± 0.26	63.82 ± 0.22
LambdaRank	50.21 ± 0.18	53.63 ± 0.24	54.18 ± 0.35	58.09 ± 0.43	56.48 ± 0.31	60.47 ± 0.06	59.46 ± 0.15	63.73 ± 0.23
ListNet	49.64 ± 0.42	53.04 ± 0.13	53.89 ± 0.19	57.62 ± 0.38	56.19 ± 0.17	59.96 ± 0.10	58.13 ± 0.08	63.29 ± 0.17
ListMLE	48.09 ± 0.26	52.03 ± 0.19	52.50 ± 0.08	56.24 ± 0.20	54.84 ± 0.09	58.81 ± 0.44	56.80 ± 0.38	62.20 ± 0.25
ApproxNDCG	48.37 ± 0.35	52.21 ± 0.17	52.83 ± 0.22	56.53 ± 0.14	55.18 ± 0.45	59.07 ± 0.33	57.17 ± 0.31	62.52 ± 0.24
NeuralNDCG	50.08 ± 0.43	53.42 ± 0.48	54.27 ± 0.36	57.84 ± 0.41	56.50 ± 0.05	60.33 ± 0.19	59.42 ± 0.16	63.45 ± 0.20
CAR+RMSE	49.73 ± 0.37	52.43 ± 0.05	53.78 ± 0.32	57.45 ± 0.14	56.20 ± 0.03	59.97 ± 0.02	58.63 ± 0.28	63.48 ± 0.29
CAR+RankNet	50.01 ± 0.27	52.86 ± 0.27	53.29 ± 0.14	57.43 ± 0.35	57.32 ± 0.28	60.02 ± 0.20	58.54 ± 0.31	63.85 ± 0.49
CAR+LambdaRank	51.26 ± 0.38	54.37 ± 0.23	54.85 ± 0.40	58.82 ± 0.36	57.87 ± 0.26	61.57 ± 0.13	59.56 ± 0.27	64.78 ± 0.30
CAR+ListNet	51.09 ± 0.42	54.42 ± 0.38	54.63 ± 0.22	58.90 ± 0.19	57.65 ± 0.31	61.04 ± 0.37	60.01 ± 0.07	65.43 ± 0.11
CAR+ListMLE	50.14 ± 0.23	53.08 ± 0.17	53.40 ± 0.25	57.63 ± 0.18	56.48 ± 0.26	60.03 ± 0.34	58.85 ± 0.42	64.28 ± 0.28
CAR+ApproxNDCG	50.72 ± 0.32	53.75 ± 0.47	54.34 ± 0.36	58.29 ± 0.12	57.32 ± 0.13	60.65 ± 0.26	59.87 ± 0.10	65.13 ± 0.07
CAR+NeuralNDCG	50.83 ± 0.34	54.08 ± 0.36	54.57 ± 0.24	58.43 ± 0.34	57.46 ± 0.24	61.08 ± 0.35	60.03 ± 0.26	65.43 ± 0.34
XGBoost	48.37 ± 0.10	52.12 ± 0.36	52.83 ± 0.19	56.45 ± 0.45	56.14 ± 0.38	60.03 ± 0.31	58.03 ± 0.17	63.61 ± 0.38
LightGBM	51.01 ± 0.16	54.86 ± 0.27	54.98 ± 0.35	58.43 ± 0.21	57.32 ± 0.42	61.02 ± 0.40	59.54 ± 0.14	64.85 ± 0.23
S^{2}phere+RMSE	50.84 ± 0.26	54.07 ± 0.46	54.94 ± 0.39	58.27 ± 0.42	56.93 ± 0.28	60.75 ± 0.19	60.04 ± 0.09	64.79 ± 0.35
S^{2}phere+RankNet	51.26 ± 0.45	54.52 ± 0.32	55.02 ± 0.43	58.56 ± 0.35	57.02 ± 0.28	61.04 ± 0.21	60.35 ± 0.34	65.30 ± 0.43
S^{2}phere+LambdaRank	51.84 ± 0.34	55.34 ± 0.27	55.71 ± 0.18	59.42 ± 0.25	57.71 ± 0.19	61.80 ± 0.33	60.87 ± 0.17	66.28 ± 0.15
S^{2}phere+ListNet	52.09 ± 0.47	55.78 ± 0.36	56.04 ± 0.30	59.63 ± 0.21	58.02 ± 0.13	62.07 ± 0.23	61.46 ± 0.08	66.57 ± 0.06
S^{2}phere+ListMLE	51.48 ± 0.24	54.82 ± 0.44	55.35 ± 0.26	58.64 ± 0.07	57.28 ± 0.05	61.34 ± 0.13	60.58 ± 0.28	65.52 ± 0.09
S^{2}phere+ApproxNDCG	52.33 ± 0.19	55.87 ± 0.26	56.26 ± 0.17	59.75 ± 0.05	58.21 ± 0.47	62.25 ± 0.48	**61.73 ± 0.04**	66.73 ± 0.40
S^{2}phere+NeuralNDCG	**52.46 ± 0.09**	**56.01 ± 0.24**	**56.85 ± 0.32**	**59.87 ± 0.16**	**58.90 ± 0.15**	**62.38 ± 0.08**	61.72 ± 0.47	**66.84 ± 0.09**

- **LightGBM** [18] is the most popular tree-based ranking model. In our work, we implement LighGBM with a listwise loss function, which outperforms other baselines [30].

Owing to the prior experience and high cost of deploying ranking models, we compare **S^{2}phere** with the above models without more previous ranking models (e.g., RankSVM [17], GSF [2], DLCM [1], et al.). *For online evaluation, due to the restriction of business information disclosures, this work reports the improvement to measure the difference between **S^{2}phere** and the **legacy system** [45].*

5.4 Experimental Settings

For *Self-tuned Label Propagation*, we adopt LightGBM as the ranking model and set the number of trees as 200, and the learning rate as 0.01. Moreover, we replace 5% query-webpage pairs under each query of $\mathcal{X}^C$ with Gaussian noise to conduct the corrupted input. We choose the encoder part of transformer, the MLP-based network, and the scorer with various LTR loss functions as the backbone, neck and head. Specifically, for the self-attentive autoencoder, the number of encoder blocks E is set as 4. Moreover, the number of attention heads is set as 2 for RMSE and RankNet. For LambdaRank, ListNet, ListMLE, ApproxNDCG, and NeuralNDCG loss, the number of attention heads is 4. We set hyper-parameter α as 0.1. For the training paradigm, the learning rate is set as 0.001 for 10 epochs in *Warm-up Fine-tuning*. Then, we jointly fine-tune the whole modules to update all weights to fit search data in *Post-fine-tuning*.

5.5 Offline Experimental Results

5.5.1 ***Comparative Results.*** Table 1 presents the offline result of **S^{2}phere** compared with competitor systems on *NDCG*@4 and *NDCG*@10. Intuitively, we could obverse that **S^{2}phere** outperforms all competitors with different losses under four ratios of labeled data. Specifically, our proposed model gains the best performance against all competitors. Particularly, **S^{2}phere** gains the largest margin with 4.37%, 4.35%, 4.06% and 4.93% improvements on *NDCG*@4 and 3.98%, 3.63%, 3.57%, 4.64% on *NDCG*@10. Furthermore, the performance of our model improves consistently as the ratio of labeled data increases. **S^{2}phere** utilizes *Semi-supervised Feature Extraction Pre-training via Perturbed Contrastive Loss* to incorporate diverse signals and reconstruct the corrupted data. *Cross-domain Ranker Pre-training over Heterogeneous LTR Datasets* could pre-train the neck to learn more cross-domain information via heterogeneous LTR datasets. The pre-trained modular network is fine-tuned following the proposed training paradigm. Moreover, there are more phenomena to be observed from the competitor results. First, CAR-based LTR models achieve the best performance among competitor systems. Specifically, CAR+LambdaRank shows the best result for all CAR-based methods. Secondly, for the tree-based methods, LightGBM performs better than XGBoost under four ratios of labelled data. Finally, MLP+LambdaRank outperforms other MLP-based LTR models. Thus, we choose MLP+LamdaRank, CAR+LambdaRank, LightGBM and **S^{2}phere** + NeuralNDCG to conduct another offline evaluation and sample the result on 5% labeled data in Table 2. We observe that our proposed model outperforms the other three competitors. Specifically, **S^{2}phere**+NeuralNDCG gains 2.157 on *PNR* and advances MLP+LambdaRank by 8.83%. Moreover, the tree-based model and CAR-based model also outperform the MLP-based LTR model, which is consistent in two offline evaluations. More comparative results on *PNR* can be found in Appendix A.1.

Table 2: Offline results on *PNR* **under 5% labeled data.**

Model	PNR	Improvement
MLP + LambdaRank	1.982	-
CAR + LambdaRank	2.146	8.27%
LightGBM	2.109	6.41%
S^{2}phere + NeuralNDCG	**2.157**	**8.83%**

5.5.2 ***Ablation Results.*** To assess the effectiveness of three key components in **S^{2}phere**, we performed extensive ablation experiments in this study. Specifically, **S^{2}phere** w/o *SLP* (*Self-tuned Label*

Table 3: Ablation studies of *Step (1) Semi-supervised Feature Extraction Pre-training via Perturbed Contrastive Loss* and *Step (3) End-to-end LTR Fine-tuning via Modular Network Composition* on $NDCG@4$ for S^{2}phere with NeuralNDCG.

Model	*NDCG@4*			
	5%	10%	15%	20%
S^{2}phere	**52.46**	**56.85**	**58.90**	**61.72**
S^{2}phere w/o *SLP of Step (1)*	51.50	55.91	58.43	61.26
S^{2}phere w/o *FEPPC of Step (1)*	51.81	56.01	58.19	61.39
S^{2}phere w/o *Warm-up Fine-tuning*	51.94	56.15	58.49	61.47

Table 4: Ablation studies of *Step (2) Cross-domain Ranker Pre-training over Heterogeneous LTR Dataset* on $NDCG@4$ for S^{2}phere with NeuralNDCG.

Model	*NDCG@4*			
	5%	10%	15%	20%
S^{2}phere w/ *LTR Dataset 1 & 2 & 3*	**52.46**	**56.85**	**58.90**	**61.72**
S^{2}phere w/ *LTR Dataset 1*	51.28	55.36	58.60	61.42
S^{2}phere w/ *LTR Dataset 2*	50.82	54.87	57.59	60.13
S^{2}phere w/ *LTR Dataset 3*	50.89	54.69	57.47	60.09
S^{2}phere w/ *LTR Dataset 1 & 2*	51.86	56.53	58.69	61.67
S^{2}phere w/ *LTR Dataset 1 & 3*	51.42	56.45	58.71	61.50
S^{2}phere w/ *LTR Dataset 2 & 3*	50.91	55.17	57.68	60.48

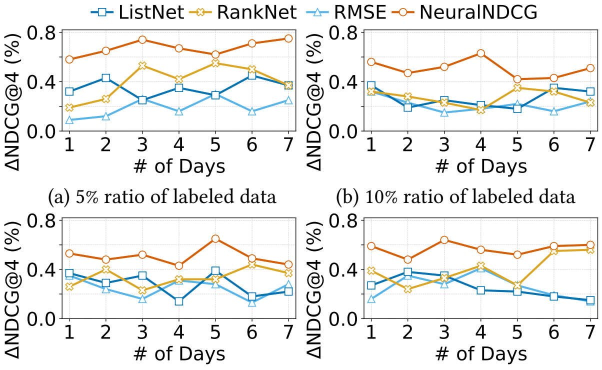
Propagation) of *Step (1)* leverages a pointwise-based self-training approach to generate pseudo-labels. **S^{2}phere** w/o *FEPPC* (*Feature Extraction Pre-training via Perturbed Contrastive Loss*) of *Step (3)* directly utilizes the MLP-based LTR model which has the same structure as the neck on the combined data with Gaussian noise. **S^{2}phere** w/o *Cross-domain Ranker Pre-training over Heterogeneous LTR Dataset* is the proposed model pre-trained without three cross-domain LTR datasets. Eventually, **S^{2}phere** w/o *Step (3)* can be considered as **S^{2}phere** without *Warm-up Fie-tuning* in terms of our proposed training paradigm.

As illustrated in Table 3, we present ablation study results of **S^{2}phere**+NeuralNCDG w/o *Step (1)* and *Step (3)*. We could see that the two steps contribute to positive improvements for **S^{2}phere** under various ratios of labeled data. Table 3 presents that *SLP* of *Step (1)* achieves the improvement with 0.96%, 0.94%, 0.47% and 0.46% on $NDCG@4$ for **S^{2}phere**+NeuralNDCG on average under 5%, 10%, 15% and 20%, respectively. Similarly, *FEPPC* of *Step (1)* also improve the performance for **S^{2}phere**+NeuralNCDG with 0.65%, 0.84%, 0.71%, and 0.33% on $NDCG@4$ under four ratios of label data. Besides, the ablation study results of **S^{2}phere** w/o *Warm-up Fine-tuning* could demonstrate the effectiveness of our proposed training paradigm. Specifically, *Warm-up Fine-tuning* in our proposed training paradigm averagely boosts the performance of **S^{2}phere** with 0.52%, 0.70%, 0.41% and 0.25% on $NDCG@4$.

Table 4 reports the ablation study results of *Cross-domain Ranker Pre-training over Heterogeneous LTR Datasets* (*Step (2)*) of **S^{2}phere**+ NeuralNDCG. Specifically, we set the notation of *LTR Dataset 1*, *LTR Dataset 2* and *LTR Dataset 3* as MSLR-Web30K, MQ2007 and MQ2008, respectively. In general, *Cross-domain Ranker Pre-training over Heterogeneous LTR Datasets* gains the best performance against other competitors. Specifically, *Cross-domain Ranker Pre-training over Heterogeneous LTR Datasets* averagely gains the largest improvement for **S^{2}phere** with 1.64%, 2.16%, 1.43% and 1.63% on $NDCG@4$ under four ratios of labeled data, respectively. We also observe that pre-trained **S^{2}phere** with *LTR Dataset 1* obtains the best performance among three single datasets and has the smallest margin with *LTR Dataset 1 & 2 & 3*. Among all compositions of two LTR datasets, *LTR Dataset 1 & 2* achieves the best performance. Moreover, we could infer that *LTR Dataset 1* has the most enormous impact among the three datasets.

Table 5: Performance improvements of the online evaluation.

Model	Δ_{AB}		ΔGSB	
	Random	Long-Tail	Random	Long-Tail
The Legacy System [45]	-	-	-	-
CAR+LambdaRank	0.17%	0.25%	3.96%	3.13%
LightGBM	0.14%	**0.32%**	3.50%	**7.50%**
S^{2}phere+NeuralNDCG	**0.21%**	0.27%	**5.00%**	6.50%



(a) 5% ratio of labeled data

(b) 10% ratio of labeled data

(c) 15% ratio of labeled data

(d) 20% ratio of labeled data

Figure 3: Online comparative performance ($\Delta NDCG@4$) of S^{2}phere with various losses for 7 days (*t*-test with $p < 0.05$ over the baseline). S^{2}phere could boost the performance compared with *the legacy system* on all days.

5.6 Online Experimental Results

5.6.1 ***Interleaving and Manual Evaluation*.** Table 5 illustrates performance improvements of three models on Δ_{AB} and ΔGSB. We first find that **S^{2}phere**+NeuralNDCG trained under 20% labeled data achieves substantial improvements for the online system on two metrics, which demonstrates the practicability and effectiveness of our proposed model. Specifically, our proposed model outperforms the *legacy system* with 0.21% and 5.00% on Δ_{AB} and ΔGSB, respectively. Also, we observe that **S^{2}phere** outperforms the *legacy system* for long-tail queries whose search frequencies are lower than 10 per week. Particularly, the advantages of Δ_{AB} and ΔGSB are 0.27% and 6.50%. Besides, we see that LightGBM gains the largest margin for long-tail queries, which reveals that tree-based models could be better adapted to this scenario.

5.6.2 ***Online A/B Test*.** During the online A/B test, we performed a seven-day experiment to compare the ranking system deployed with **S^{2}phere** against the *legacy system*. The implementation details of the online A/B Test could be found in the Appendix. Figure 3 illustrates the improvement of **S^{2}phere** with various losses compared with the *legacy system* on $\Delta NDCG@4$. **S^{2}phere** consistently enhances performance across all days compared to the *legacy system*, proving its practicality in elevating the performance of Baidu Search. Moreover, notable advancements are observed as $\mathbf{S^2}$**phere** delivers significant improvements on top of the Baidu Search framework. Specifically, it is obvious that the largest improvements of trained $\mathbf{S^2}$**phere**+NeuralNDCG on four ratios of labled data are 0.60%, 0.63%, 0.65% and 0.75% on $\Delta NDCG@4$, respectively. The notable advancements observed clearly indicate the effectiveness of $\mathbf{S^2}$**phere**. Eventually, we could observe that $\mathbf{S^2}$**phere** performs stably on all days.

6 RELATED WORK

Learning to Rank. We could divide LTR models into three families according to loss functions: pointwise [10, 20], pairwise [16, 43] and listwise [5, 34]. The pointwise model transformers ranking tasks into regression or classification problems to accurately match labels with query-webpage pairs. However, the pairwise model formulates a pair of webpages into a webpage pair and redefines LTR tasks as classification problems. The listwise model considers the entire webpage list as a single sample and directly optimizes the evaluation metrics (e.g., NDCG [3, 15, 39]) without decomposing it into pairwise or pointwise comparisons. *In our work,* $\mathbf{S^2}$***phere*** *leverages both labeled/unlabeled data collected at the search engine and incorporates open-source heterogeneous LTR datasets to boost the performance of webpage ranking for search.*

Semi-supervised Learning for LTR. Semi-supervised learning has been utilized for pseudo-label generation in many machine learning tasks [33, 40], however, the use of semi-supervised learning in LTR has not been well investigated. [21] takes a semi-supervised approach to consider the divergence between the prediction of various LTR models and incorporates such divergence to improve co-training performance in industrial practice. *In this work, inspired by [21],* $\mathbf{S^2}$***phere*** *takes a self-tuning approach to propagate labels from annotated query-webpage pairs to unlabeled ones.*

Perturbed Contrastive Learning. The essence of perturbed contrastive learning is to reconstruct corrupted data and learn the joint probability distribution of samples via the training process. Variational autoencoder structures [19, 35] have been used to reconstruct data. Specifically, [35] adopts a cascaded residual autoencoder adapted from the denoised autoencoder [37] to calculate the residual and reconstruct the corrupted multi-modal data sequence. *In this work,* $\mathbf{S^2}$***phere*** *utilizes a transformer-based denoised autoencoder to reconstruct structured data via perturbed contrastive learning.*

Open-source LTR Datasets. Open-source LTR datasets significantly contribute to the development of the research and application of LTR. Nowadays, many large-scale search companies have proposed their standard and publicly available LTR datasets, i.e., MSLR-Web30K [29], MQ2007 [29], MQ2008 [29], Yahoo! LTR dataset [6], and Baidu-ULTR [44]. Specifically, all the datasets contain features, relevance judgments and data partitioning. In particular, according to the application scenarios where data is collected, the features contain special domain information. *This work leverages heterogeneous LTR datasets, which contain three open-source datasets, to pre-train the neck for learning cross-domain information and obtaining reasoning capability.*

7 DISCUSSIONS

In this work, we propose $\mathbf{S^2}$**phere** to pre-train LTR transformers for large-scale web search, incorporating labeled/unlabeled data and multiple heterogeneous open-source LTR datasets. We further deploy $\mathbf{S^2}$**phere** in the context of a real-world large-scale search engine for performance evaluation. Some open issues are as follows.

First of all, $\mathbf{S^2}$**phere** aims at using both labeled/unlabeled query-webpage pairs for pre-training and fine-tuning in LTR, where a *simple-yet-effective* strategy derived from self-tuning has been adopted and generates pseudo labels for unlabeled pairs. Apparently, using advanced semi-supervised learning algorithms [7, 42] could further improve the performance of $\mathbf{S^2}$**phere**. However, the primary focus of this work is to leverage heterogeneous open-source LTR datasets to pre-train the model. Proposing new semi-supervised learning algorithms, hereby, might be out of our scope. Actually, we have carried out extensive experiments to investigate and compare the performance of different self-tuning strategies/settings in generating pseudo labels. The results show they could achieve similar performance. Please refer to Appendix A.2 and A.3 for details.

In addition to algorithm design, we evaluate $\mathbf{S^2}$**phere** using both offline and online experiments to demonstrate the performance advantages of the proposed algorithms. It is reasonable to doubt whether our experiment settings are representative for LTR at industry-scale. For online experiments, we include the comparisons $\mathbf{S^2}$**phere** against the *legacy system* and multiple baseline LTR models, where we use the real-world web search traffics from the Baidu Search engine for A/B tests. Please refer to Appendix B for more details on online experiment settings.

8 CONCLUSIONS

In this work, we investigate the problem of pre-training LTR models using both labeled and unlabeled samples, especially we focus on the use of well-annotated samples in heterogeneous open-source LTR datasets to boost the performance of pre-training. We propose $\mathbf{S^2}$**phere**—*Semi-Supervised Pre-training with Heterogeneous LTR data* strategies for LTR tasks. Specifically, $\mathbf{S^2}$**phere** consists of three steps: *(1) Semi-supervised Feature Extraction Pre-training via Perturbed Contrastive Loss, (2) Cross-domain Ranker Pre-training over Heterogeneous LTR Datasets and (3) End-to-end LTR Fine-tuning via Modular Network Composition.* To demonstrate the effectiveness of $\mathbf{S^2}$**phere**, we performed comprehensive offline and online experiments, comparing its performance against numerous competitor systems. Offline experiment results verify the superior performance of $\mathbf{S^2}$**phere** compared to other competitors. Moreover, the real-world application of $\mathbf{S^2}$**phere** exhibits a substantial improvement in online ranking performance, which is consistent with the offline results.

ACKNOWLEDGMENTS

This work was supported in part by National Key R&D Program of China (No. 2021ZD0110303), NSFC grant 62141220, 61972253, U1908212, 62172276, 61972254, the Program for Professor of Special Appointment (Eastern Scholar) at Shanghai Institutions of Higher Learning, Shanghai Science and Technology Development Funds 23YF1420500, Open Research Projects of Zhejiang Lab No. 2022NL0AB01.

REFERENCES

[1] Qingyao Ai, Keping Bi, Jiafeng Guo, and W Bruce Croft. 2018. Learning a deep listwise context model for ranking refinement. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 135–144.

[2] Qingyao Ai, Xuanhui Wang, Sebastian Bruch, Nadav Golbandi, Michael Bendersky, and Marc Najork. 2019. Learning groupwise multivariate scoring functions using deep neural networks. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*. 85–92.

[3] Sebastian Bruch, Masrour Zoghi, Michael Bendersky, and Marc Najork. 2019. Revisiting Approximate Metric Optimization in the Age of Deep Neural Networks. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR*. 1241–1244.

[4] Christopher J. C. Burges, Robert Ragno, and Quoc Viet Le. 2006. Learning to Rank with Nonsmooth Cost Functions. In *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*. 193–200.

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

[6] Olivier Chapelle and Yi Chang. 2011. Yahoo! Learning to Rank Challenge Overview. In *Proceedings of the Yahoo! Learning to Rank Challenge, held at ICML (JMLR Proceedings, Vol. 14)*. JMLR.org, 1–24.

[7] Baixu Chen, Junguang Jiang, Ximei Wang, Pengfei Wan, Jianmin Wang, and Mingsheng Long. 2022. Debiased Self-Training for Semi-Supervised Learning. In *Processing of the 36th Annual Conference on Neural Information Processing Systems*.

[8] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 785–794.

[9] Aleksandr Chuklin, Anne Schuth, Ke Zhou, and Maarten De Rijke. 2015. A comparative analysis of interleaving methods for aggregated search. *ACM Transactions on Information Systems (TOIS)* 33, 2 (2015), 1–38.

[10] William S Cooper, Fredric C Gey, and Daniel P Dabney. 1992. Probabilistic retrieval based on staged logistic regression. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*. 198–210.

[11] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *Advances in neural information processing systems* 32 (2019).

[12] Suchin Gururangan, Mike Lewis, Ari Holtzman, Noah Smith, and Luke Zettlemoyer. 2022. DEMix Layers: Disentangling Domains for Modular Language Modeling. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics*. 5557–5576.

[13] Kaiming He, Ross Girshick, and Piotr Dollár. 2019. Rethinking imagenet pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4918–4927.

[14] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.

[15] Kalervo Järvelin and Jaana Kekäläinen. 2017. IR evaluation methods for retrieving highly relevant documents. In *ACM SIGIR Forum*, Vol. 51. 243–250.

[16] Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international Conference on Knowledge Discovery & Data Mining*. 133–142.

[17] Thorsten Joachims. 2006. Training linear SVMs in linear time. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 217–226.

[18] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*. 3146–3154.

[19] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *Processing of the 2nd International Conference on Learning Representations*.

[20] Ping Li, Qiang Wu, and Christopher Burges. 2008. McRank: Learning to Rank Using Multiple Classification and Gradient Boosting. In *Advances in Neural Information Processing Systems*. 65–72.

[21] Yuchen Li, Haoyi Xiong, Qingzhong Wang, Linghe Kong, Hao Liu, Haifang Li, Jiang Bian, Shuaiqiang Wang, Guihai Chen, Dejing Dou, et al. 2023. COLTR: Semi-supervised Learning to Rank with Co-training and Over-parameterization for Web Search. *IEEE Transactions on Knowledge and Data Engineering* (2023).

[22] Hao Liu, Qian Gao, Jiang Li, Xiaochao Liao, Hao Xiong, Guangxing Chen, Wenlin Wang, Guobao Yang, Zhiwei Zha, Daxiang Dong, et al. 2021. Jizhi: A fast and cost-effective model-as-a-service system for web-scale online inference at Baidu. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3289–3298.

[23] Yiding Liu, Weixue Lu, Suqi Cheng, Daiting Shi, Shuaiqiang Wang, Zhicong Cheng, and Dawei Yin. 2021. Pre-trained Language Model for Web-scale Retrieval in Baidu Search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*.

[24] Przemysław Pobrotyn, Tomasz Bartczak, Mikołaj Synowiec, Radosław Białobrzeski, and Jarosław Bojar. 2020. Context-aware learning to rank with self-attention. *arXiv preprint arXiv:2005.10084* (2020).

[25] Przemysław Pobrotyn and Radosław Białobrzeski. 2021. Neuralndcg: Direct optimisation of a ranking metric via differentiable relaxation of sorting. *arXiv preprint arXiv:2102.07831* (2021).

[26] An Qin, Dianming Hu, Jun Liu, Wenjun Yang, and Dai Tan. 2014. Fatman: Cost-saving and reliable archival storage based on volunteer resources. *Proceedings of the VLDB Endowment* 7, 13 (2014), 1748–1753.

[27] An Qin, Mengbai Xiao, Jin Ma, Dai Tan, Rubao Lee, and Xiaodong Zhang. 2019. DirectLoad: A Fast Web-scale Index System across Large Regional Centers. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 1790–1801.

[28] Tao Qin, Tie-Yan Liu, and Hang Li. 2010. A general approximation framework for direct optimization of information retrieval measures. *Inf. Retr.* 13, 4 (2010), 375–397. https://doi.org/10.1007/s10791-009-9124-x

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

[30] Zhen Qin, Le Yan, Honglei Zhuang, Yi Tay, Rama Kumar Pasumarthi, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2021. Are Neural Rankers still Outperformed by Gradient Boosted Decision Trees?. In *Proceedings of the 9th International Conference on Learning Representations, ICLR*.

[31] Zi-Hao Qiu, Quanqi Hu, Yongjian Zhong, Lijun Zhang, and Tianbao Yang. 2022. Large-scale Stochastic Optimization of NDCG Surrogates for Deep Learning with Provable Convergence. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA (Proceedings of Machine Learning Research, Vol. 162)*. PMLR, 18122–18152.

[32] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223* (2019).

[33] Martin Szummer and Emine Yilmaz. 2011. Semi-supervised learning to rank with preference regularization. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management*. 269–278.

[34] Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. 2008. Softrank: optimizing non-smooth rank metrics. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*. 77–86.

[35] Luan Tran, Xiaoming Liu, Jiayu Zhou, and Rong Jin. 2017. Missing Modalities Imputation via Cascaded Residual Autoencoder. In *2017 IEEE Conference on Computer Vision and Pattern Recognition*. 4971–4980.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*. 5998–6008.

[37] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *Journal of machine learning research (JMLR)* 11 (2010), 3371–3408.

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

[39] Tianbao Yang. 2022. Algorithmic Foundation of Deep X-risk Optimization. *arXiv preprint arXiv:2206.00439* (2022).

[40] Xiangli Yang, Zixing Song, Irwin King, and Zenglin Xu. 2022. A survey on deep semi-supervised learning. *IEEE Transactions on Knowledge and Data Engineering* (2022).

[41] Shiqi Zhao, Haifeng Wang, Chao Li, Ting Liu, and Yi Guan. 2011. Automatically generating questions from queries for community-based question answering. In *Proceedings of 5th international joint conference on natural language processing*. 929–937.

[42] Zhen Zhao, Luping Zhou, Lei Wang, Yinghuan Shi, and Yang Gao. 2022. LaSSL: label-guided self-training for semi-supervised learning. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*. 9208–9216.

[43] Zhaohui Zheng, Keke Chen, Gordon Sun, and Hongyuan Zha. 2007. A regression framework for learning ranking functions using relative relevance judgments. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. 287–294.

[44] Lixin Zou, Haitao Mao, Xiaokai Chu, Jiliang Tang, Wenwen Ye, Shuaiqiang Wang, and Dawei Yin. 2022. A Large Scale Search Dataset for Unbiased Learning to Rank. In *NeurIPS*.

[45] Lixin Zou, Shengqiang Zhang, Hengyi Cai, Dehong Ma, Suqi Cheng, Shuaiqiang Wang, Daiting Shi, Zhicong Cheng, and Dawei Yin. 2021. Pre-trained language model based ranking in Baidu search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 4014–4022.

Table 6: Offline comparative results on PNR under various ratios of labeled data.

Model	5%		10%		15%		20%	
	PNR	Improvement	PNR	Improvement	PNR	Improvement	PNR	Improvement
MLP + LambdaRank	1.982	-	2.077	-	2.348	-	2.537	-
CAR + LambdaRank	2.146	8.27%	2.236	7.66%	2.464	4.94%	2.604	2.64%
LightGBM	2.109	6.41%	2.179	4.91%	2.397	2.09%	2.583	1.81%
S^{2}phere + NeuralNDCG	**2.157**	**8.83%**	**2.246**	**8.14%**	**2.486**	**5.88%**	**2.625**	**3.47%**

Table 7: Comparative results on $NDCG@4$ of *Self-tuned Label Propagation* in *Step (1)* and other semi-supervised LTR models under various ratios of labeled data.

Model	$NDCG@4$			
	5%	10%	15%	20%
Pointwise Self-training	49.74	54.80	56.82	59.21
Pairwise Self-training	49.95	54.91	56.81	59.50
Listwise Self-training	50.01	54.98	57.32	59.54
Self-training w/ Pointwise-to-Pairwise	49.83	55.72	57.00	60.13
Self-training w/ Pointwise-to-Listwise	49.88	55.48	57.85	60.07
Self-training w/ Pairwise-to-Pointwise	50.04	56.05	57.67	60.39
Self-training w/ Pairwise-to-Listwise	49.97	56.20	57.64	60.52
Self-training w/ Listwise-to-Pairwise	50.28	56.12	57.56	60.71
Self-tuned Label Propagation	**50.54**	**56.36**	**58.19**	**60.83**

A OFFLINE EVALUATION

A.1 Offline Comparative Results

According to the offline experimental results of **S^{2}phere** compared with competitor systems under various ratios of labeled data on $NDCG@4$, we choose MLP+LamdaRank, CAR+LambdaRank, LightGBM and **S^{2}phere**+NeuralNDCG, which are the best LTR models in MLP-based models, CAP-based models, tree-based models and **S^{2}phere**-based models, to conduct another offline evaluation on PNR and report the result and improvement in Table 6 (referred to Section 5.5.1). We could see that the experimental results are consistent with Table 1. As illustrated in Table 1, our proposed model gains the best performance on PNR and achieves the largest improvement under various ratios of labeled data. Specifically, **S^{2}phere** + NeuralNDCG reaches 2.157, 2.246, 2.486 and 2.625 on PNR under 5%, 10%, 15% and 20%, respectively. **S^{2}phere** + NeuralNDCG advances MLP+LambdaRank by 8.83%, 8.14%, 5.88% and 3.47% under four ratios of labeled data. Moreover, there are two phenomena to be observed from the competitor results. CAR + LambdaRank outperforms the MLP-based model and LightGBM under all ratios of labeled data. On the other hand, the MLP-based model achieves worst results than other competitors. In general, the results of the two offline comparative experiments are consistent.

A.2 Comparative Results of *Self-tuned Label Propagation*

For *Self-tuned Label Propagation* (*SLP*) in *Step (1)*, we choose LightGBM as the based ranking model with various ranking loss functions (i.e., pointwise: RMSE, pairwise: RankNet and listwise: LambdaMart). As shown in Table 7, we conduct extensive experiments to investigate the performance of *SLP* compared with other semi-supervised learning models. For each semi-supervised model, we choose the model with the best performance of all validation rounds for testing. Intuitively, we observe that *SLP* gains the best performance compared with other semi-supervised LTR baselines under four ratios of labeled data on $NDCG@4$. **S^{2}phere** uses *SLP* to incorporate the diversity of prediction results of the listwise model and the pointwise model in a loop of multiple rounds. As the stronger learner, the listwise model predicts more accurate pseudo-labels for the pointwise model. Then the weaker model, the pointwise model, generates relatively inaccurate but diverse pseudo-labels to train the stronger model. In such co-training mechanism, *SLP* gains the best performance. Moreover, we could observe that *SLP* obtains the largest improvement with 1.59% on $NDCG@4$ under the 20% ratio of labeled data. Although co-training models can not obtain the performance of *SLP*, all co-training models also outperform the three self-training models.

A.3 Parameter Sensitivity: T

In this study, we conduct a series of experiments to investigate how the number of rounds (i.e., T) impacts the performance of *Self-tuned Label Propagation*. As shown in Table 8, we report the results of *SLP* under various ratios of labeled data in 10 co-training rounds on the validation set. Intuitively, the results show that *SLP* gains the best performance at the 4^{th}, 6^{th}, 5^{th} and 5^{th} round on $NDCG@4$ under various ratios of labeled data, respectively. For each round, the LightGBM-based LTR model with listwise loss function on the combined data and generate pseudo-labels for the unlabeled data. Then, the pseudo-labeled data is combined with the labeled data. Next, the pointwise-based LTR model is trained on the combined data and generates the results. According to the evaluation results, we choose the trained *SLP* at the round with the best performance for LTR tasks.

A.4 Offline Comparative Results

According to the offline experimental results of **S^{2}phere** compared with competitor systems under various ratios of labeled data on $NDCG@4$, we choose MLP+LamdaRank, CAR+LambdaRank, LightGBM and **S^{2}phere**+NeuralNDCG, which are the best LTR models in MLP-based models, CAP-based models, tree-based models and **S^{2}phere**-based models, to conduct another offline evaluation on PNR and report the result and improvement in Table 6 (referred to Section 5.5.1). We could see that the experimental results are consistent with Table 1. As illustrated in Table 1, our proposed model gains the best performance on PNR and achieves the largest improvement under various ratios of labeled data. Specifically, **S^{2}phere** + NeuralNDCG reaches 2.157, 2.246, 2.486 and 2.625 on PNR under 5%, 10%, 15% and 20%, respectively. **S^{2}phere** + NeuralNDCG advances MLP+LambdaRank by 8.83%, 8.14%, 5.88% and 3.47% under

Table 8: Performance of *Self-tuned Label Propagation* in *Step (1)* with LightGBM-based LTR model on $NDCG@4$ under various ratios of labeled data in 10 rounds.

Ratio	# of Rounds									
	1	2	3	4	5	6	7	8	9	10
5%	49.86	49.97	49.34	**50.54**	49.10	49.25	49.53	49.73	49.46	49.55
10%	53.68	53.15	53.61	54.66	55.13	**56.37**	53.07	53.65	54.71	53.91
15%	55.63	56.37	56.55	57.12	**58.19**	56.59	56.28	54.86	56.06	56.34
20%	60.24	60.30	60.54	60.71	**60.83**	59.64	59.75	59.79	59.93	60.18

four ratios of labeled data. Moreover, there are two phenomena to be observed from the competitor results. CAR + LambdaRank outperforms the MLP-based model and LightGBM under all ratios of labeled data. On the other hand, the MLP-based model achieves worst results than other competitors. In general, the results of the two offline comparative experiments are consistent.

B IMPLEMENTATION DETAILS OF ONLINE A/B TEST

In this section, we present the implementation detail of the online A/B test. We conduct the experiment that compares the new ranking system, which deploys our proposed model, with the *legacy system* for seven days on top of Baidu Search[1] engine.

More specifically, the online baseline is a pre-trained language model [45] with a simple MLP-based ranking regressor to carry out the ranking task, which has been deployed at Baidu Search as the legacy ranking system. During the test, we replace the original LTR model of the *legacy system* with **S^{2}phere** to accomplish the ranking task. Each day, we perform preprocessing by filtering out pornographic and legally prohibited webpages. To obtain relevance scores for the selected query-webpage pairs, we enlist the help of eight common annotators. Quality control measures are implemented by our professional annotators, ensuring that the accuracy of the annotations exceeded 85%. The relevance scores derived from the annotations are then used to train our proposed model, employing a weighted average approach. The online experiments are conducted using 5% of real-world web traffic from Baidu Search, focusing on metrics that directly impact the user experience. We evaluate the *NDCG* of the top 4 ranking results and calculate $\Delta NDCG_4$ between the chosen model and online *legacy system*. More online experimental results and analysis are discussed in Section 5.6.2.

[1] Baidu Search, a large-scale search engine, https://www.baidu.com/