COLTR: Semi-supervised Learning to Rank with Co-training and Over-parameterization for Web-Scale Search

YUCHEN LI, Shanghai Jiao Tong University, China and Baidu Inc., China
HAOYI XIONG, QINGZHONG WANG, Baidu Inc., China
LINGHE KONG, Shanghai Jiao Tong University, China
HAIYAN JIANG, HAIFANG LI, JIANG BIAN, SHUAIQIANG WANG, Baidu Inc., China
GUIHAI CHEN, Shanghai Jiao Tong University, China
DAWEI YIN, DEJING DOU, Baidu Inc., China

While *learning to rank* (LTR) has been widely used in web search to prioritize most relevant webpages among the retrieved contents subject to the input queries, the traditional LTR models fail to deliver decent performance due to two main reasons: 1) the lack of well-annotated query-webpage pairs with ranking scores to cover search queries of various popularity, and 2) ill-trained models based on a limited number of training samples with poor generalization performance. To improve the performance of LTR models, tremendous efforts have been done from above two aspects, such as enlarging training sets with pseudo-labels of ranking scores by self-training, or refining the features used for LTR through feature extraction and dimension reduction. Though LTR performance has been marginally increased, we still believe these methods could be further improved in the newly-fashioned "interpolating regime". Specifically, instead of lower the number of features used for LTR models, our work proposes to transform original data with random Fourier feature, so as to over-parameterize the downstream LTR models (e.g., GBRank or LightGBM) with features in ultra-high dimensionality and achieve superb generalization performance. Furthermore, rather than self-training with pseudo-labels produced by the same LTR model in a "self-tuned" fashion, the proposed method incorporates the diversity of prediction results between the listwise and pointwise LTR models while co-training both models with a cyclic labeling-prediction pipeline in a "ping-pong" manner. We deploy the proposed *Co-trained and Over-parameterized LTR* system **COLTR** at Baidu and evaluate **COLTR** with a large number of baseline methods. The results show that **COLTR** could achieve $\Delta NDCG_4$=3.64%∼4.92%, compared to baselines, under various ratios of labeled samples. We also conduct a 7-day A/B test using the realistic web traffics of Baidu search, where we can still observe significant performance improvement around $\Delta NDCG_4$=0.17%∼0.92% in real-world applications. **COLTR** performs consistently both in online and offline experiments.

CCS Concepts: • **Information systems** → **Learning to rank**; • **Computing methodologies** → **Semi-supervised learning settings**.

Additional Key Words and Phrases: Learning to Rank, Semi-supervised Learning, Over-parameterization

ACM Reference Format:
Yuchen Li, Haoyi Xiong, Qingzhong Wang, Linghe Kong, Haiyan Jiang, Haifang Li, Jiang Bian, Shuaiqiang Wang, Guihai Chen, and Dawei Yin, Dejing Dou. 2023. **COLTR**: Semi-supervised Learning to Rank with

Authors' addresses: Yuchen Li, Shanghai Jiao Tong University, Shanghai, China and Baidu Inc., Beijing, China; Haoyi Xiong, Qingzhong Wang, Baidu Inc., Beijing, China; Linghe Kong, Shanghai Jiao Tong University, Shanghai, China; Haiyan Jiang, Haifang Li, Jiang Bian, Shuaiqiang Wang, Baidu Inc., Beijing, China; Guihai Chen, Shanghai Jiao Tong University, Shanghai, China; Dawei Yin, Dejing Dou, Baidu Inc., Beijing, China.

XXXX-XXXX/2023/1-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Co-training and Over-parameterization for Web-Scale Search. 1, 1 (January 2023), 15 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

With a query (e.g., a string of texts) input by the user, a search engine needs first extract keywords/phrases from the query and recognize the user's intention [40, 47]. Given keywords/phrases extracted, the search engine compares the similarities/relevance between the query and webpages and then retrieves a number of relevant webpages from a database consisting of trillions of webpages [15, 22]. Further, the search engine sorts the retrieved webpages using the features of contents and click-throughs. The search engine tops the most relevant webpages in the response to the query [47]. To optimize the user experience of search, ranking the retrieved contents is a key step, where *Learning to Rank* (LTR) plays a critical role.

To achieve high accuracy for LTR, there needs to collect ultra-large training datasets and train LTR models for ranking. Specifically, given a large volume of queries with varying popularity (from highly frequent to infrequent queries), the search engine needs to first label the rank of every relevant webpage for every query and then train the LTR models through supervised learning. However, it is extremely expensive and time-consuming to label the ranks of relevant webpages for every query [34]. To address this issue, it frequently needs to incorporate both labeled and unlabeled query-webpage pairs to train LTR models in a semi-supervised learning setting. However, semi-supervised LTR at web-scale is not easy, as it might be necessary to leverage trillions of webpages under billions of queries without ranking scores to improve LTR models training based on an extremely small number of labeled samples.

A simple way to enable semi-supervised learning is *self-training* [44, 45], where the machine learning model was *(1)* first trained with the labeled samples, then *(2)* predicted over unlabeled data to obtain pseudo-labels. Later, *(3)* the self-trainer paired unlabeled samples with pseudo-labels and re-trained the model incorporating with both labeled and pseudo-labeled samples. Above steps *(2)* and *(3)* could recycle in a self-tuned manner until achieving the best validation performance. Compared to other semi-supervised learning algorithms, such as graph-based or metric learning based approach [4, 11, 36, 44], the *self-training* mechanism is a scalable yet effective method blessed by its low-complexity nature. For example, it has been used to boost the performance of ImageNet classification tasks through incorporating large-scale unlabeled images [39]. The performance of self-training however would be bottlenecked for LTR tasks by issues as follows.

- *Representation Learning.* Self-training could improve performance of deep learning in computer vision tasks, as it enhances representation learning of deep neural networks using unlabeled images [39]. Even, with mislabeled samples, deep neural networks could still learn the representations of images based on the weak and noisy supervision signals. However, in industry practices, LTR tasks usually use a small number of hand-crafted features (e.g., from several dozens to hundreds of features) [9, 28, 29] and are based on tree-based models such as Gradient Boosting Machines (GBM) [20, 23, 30, 41] and its derivatives to pursue high throughput under concurrency. In this way, there needs to transform *Learning to Rank* (LTR) to a representation learning task based on high-dimensional data and scale-up representation learning with statistical learners.
- *Diversifying Noisy Supervision Signals.* Yet another problem of self-training is over-fitting to the inaccurate pseudo-labels (e.g., noisy supervision signals), as the LTR model learns from a set of pseudo-labels derived from the (inaccurate) prediction results of a LTR model trained in the previous round. One way to solve the problem is to co-train the LTR model with multiple classifiers [14, 24, 43], so as to incorporate the diversity of prediction outputs from multiple classifiers in an ensemble learning fashion [10, 42]. It has been found that strong learners could

trained with limited labeled samples through making weak learners (producing inaccurate but diverse prediction results) teach each other [32]. In this way, there needs a way to co-train multiple LTR models while making their prediction results diverse, using the same set of labeled/unlabeled samples.

To scale-up semi-supervised learning for LTR at web-scale, we propose **COLTR** — *Co-trained and Over-parameterized LTR models*, where we solve aforementioned two technical issues with *random Fourier feature (RFF) based over-parameterization* and *multi-loss co-training* strategies respectively. Specifically, inspired by the recently observed phenomenon "double descent" of generalization performance [3] with increasing complexity of models, **COLTR** adopts feature-wise "double descent" and leverages random Fourier features (RFF) to extend the dimensions of features for LTR data (e.g., query-webpage pairs). With RFF transformed samples, **COLTR** could over-parameterize LTR models so as to enable the representation learning in the so-called interpolating regime [2] with superb performance improvement. Furthermore, **COLTR** co-trains dual LTR models with *listwise* and *pointwise* losses respectively, in a loop of multiple rounds. Specifically, **COLTR** first trains a LTR model based on the *listwise* loss using both labeled/unlabeled query-webpage pairs in a self-training manner, and then generates pseudo-labels for unlabeled samples to train another LTR model based on the *pointwise* loss. Later, **COLTR** makes these two LTR models teach each other in multiple rounds, with pseudo-labels updated. In summary, this work makes contributions as follows.

- We study the problem of semi-supervised *learning to rank* in the context of web-scale search, where we particularly focus on the technical challenges on enabling *representation learning* and diversifying *noisy supervision signals*. To the best of our knowledge, this work is the first to study semi-supervised training for LTR models with labeled/unlabeled query-webpage pairs by addressing the mathematical phenomenon of interpolating [2, 3] in LTR tasks and the diversity of LTR models trained with various loss (e.g., pointwise, pairwise, and listwise) functions.
- We design and implement **COLTR**, incorporating both labeled or unlabeled query-webpage pairs for training LTR models in semi-supervised manner. Specifically, **COLTR** consists of three steps: (1) *RFF-based over-parameterization* that pushes the limits of representation learning to interpolating regime [2], (2) *Listwise-based Self-training* that initializes pseudo-labels of unlabeled samples using the predictions of a LTR model trained by the listwise loss, and (3) *Multi-Loss Co-training for LTR* that makes pointwise and listwise models learn from each other for multiple rounds with pseudo-labels updated by predictions.
- We deploy **COLTR** at Baidu Search and evaluate the proposed algorithm using both offline experiments and online A/B tests in comparison with baseline algorithms. The experiment results show that, compared to the state of the art in webpage ranking, **COLTR** could achieve $\Delta NDCG_4$=3.64%~4.92% in offline experiments and $\Delta NDCG_4$=0.17%~0.92% in online A/B tests under fair comparisons. Ablations studies further confirm the effectiveness of *RFF-based model over-parameterization* and *multi-loss co-training* for LTR.

Note that we focus on low-complexity strategies for semi-super-vised LTR that can scale-up on real-world traffics, thus advanced methods with higher complexity are not in the scope of our study. Furthermore, please be advised that **COLTR** is a semi-supervised LTR component in Baidu Search. Experiment results reported in this study are based on A/B tests with the *status quo* of Baidu search, which has already secured excellent LTR performance.

2 SYSTEM OVERVIEW

In this section, we introduce the system design of Baidu search, where we first present the overall pipeline of Baidu search and then introduce the design of each component. As shown in Figure 1,

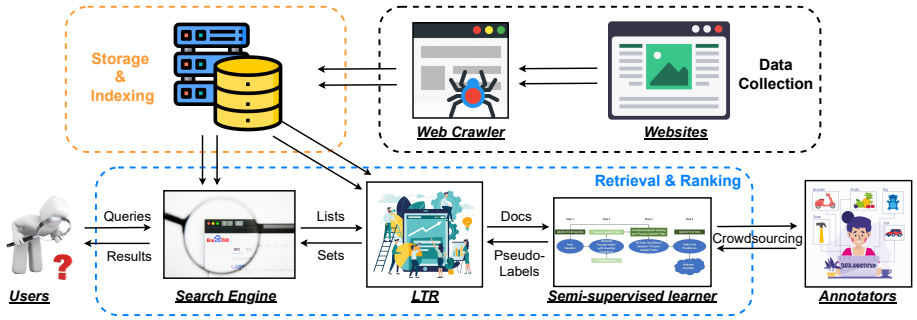


Fig. 1. An Overview of a Search System with Semi-supervised LTR.

Baidu search engine is essentially with three stages as follows: (1) *Data collection*, (2) *Storage & Indexing* and (3) *Retrieval & ranking*.

2.1 Data Collection

Given the massive webpages on the web, Baidu search engine adopts a high-performance crawler, namely Web Crawler, to fetch and download webpages from the web. Specifically, the Web Crawler screens a list of links (*i.e.*, URLs) for new webpages and updated ones, then stores the valid links (or URLs) with desired contents in a large downloading list. Later, the Web Crawler starts downloading the webpages on the list, upon to the real-time web traffics of the Baidu search engine. Note that once a new/updated webpage fetched, the Web Crawler would first parses the contents, finds all possible links, and adds them into the list for potential screening.

2.2 Storage & Indexing

Given the massive webpages downloaded from the web, Baidu search stores these contents in distributed archival storage systems with Fatman [25] and builds efficient indices for high-performance search based on DirectLoad [26]. While Fatman [25] could significantly reduces the costs of storage by using the elastic resources, such as underutilized servers and temporally spare storage, across multiple regional data centers of Baidu, DirectLoad [26] balances the loads of indexing over these data centers with superb I/O efficiency using novel key-value operations and in-memory computation.

2.3 Retrieval & Ranking

Given the massive webpages archived and indexed, when a new search query comes, Baidu search first parses the query to understand the user's intention and key phrases as a group of features [40], then Baidu search retrieves contents using the key phrases an accelerated system distributed query execution [27], and finally ranks these contents accordingly the relevance to the query and intentions [17]. In this component, LTR is an indispensable component to determine the order of webpages displayed in the search results and would affect users' search experiences. The problem of LTR could be formulated as follow.

2.3.1 LTR Settings. In this section, we formalize the LTR settings and propose our notations. Given a set of search queries $\mathcal{Q} = \{q_1, q_2, \dots\}$ and all archived webpages $\mathcal{D} = \{d_1, d_2, \dots\}$, for each query $q_i \in \mathcal{Q}$, the search engine could retrieve a set of relevant webpages denoted as $D_i = \{d_1^i, d_2^i, \dots\} \subset \mathcal{D}$. Through annotating, there also might exist a set of ranking scores $\boldsymbol{y}_i = \{y_1^i, y_2^i, \dots\}$ for q_i, which characterizes the relevance of each document $d_j^i \in D_i$ to the search query q_i. In our work, we follow

the settings in [28, 47] and scale the ranking score from 0 to 4 to represent levels of relevance (i.e., {**bad, fair, good, excellent, perfect**} the bigger the more relevant).

We denote a set of query-webpage pairs with ranking score annotations as a set of triples such as $\mathcal{T} = \{(q_1, D_1, \boldsymbol{y}_1), (q_2, D_2, \boldsymbol{y}_2), (q_3, D_3, \boldsymbol{y}_3), \dots\}$. We aim to gain a LTR scoring function $f : \mathcal{Q} \times \mathcal{D} \to [0, 4]$. Therefore, the goal is recast to learn a scoring function f which minimizes the loss as:

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

where ℓ represents the loss of the ranking prediction of webpage d_j^i of query q_i against the ground truth $\boldsymbol{y}_j^i$. Three types of loss functions are defined as follows.

The **pointwise** loss converts LTR into a regression task with a L2-loss such that $\ell_j^i = |f(q^i, d_j^i) - \boldsymbol{y}_j^i|^2$ for webpage d_j^i of query q_i.

The **pairwise** loss is based on the pairwise order of any two webpages $\{d_j^i, d_k^i\} \subset D_i$. Here, we follow[7], such that:

$$\ell_{j,k}^i = \log\left\{1 + e^{(y_j - y_k)}\right\} \cdot \left|\Delta Z_{jk}\right|, \tag{2}$$

where ΔZ_{jk} reflects the effects[18] by swapping the positions of two webpages.

The **listwise** loss measures, for each query q_i, the divergence between the probability distributions of the predicted ranking scores for every webpage in D_i and the ground truth $\boldsymbol{y}_i$. The divergence is estimated using cross-entropy, while the probability distribution of ranking scores is estimated using normalizers of softmax-alike [6].

2.3.2 Semi-supervised LTR. As annotators for web search can only label a small number of query-webpage pairs while annotating more query-webpage pairs might be expensive and time-consuming, the core problem of LTR is thus to incorporate some unlabeled query-webpage pairs, i.e., $\mathcal{T}' = \{(q_1', D_1'), (q_2', D_2'), \dots\} \subset \mathcal{Q} \times 2^{\mathcal{D}}$ and $|\mathcal{T}'| \gg |\mathcal{T}|$, to improve training. To this end, the *Semi-supervised Learner* plays to train LTR models using both labeled and unlabeled query-webpage pairs. In our research, we propose **COLTR** to enable semi-supervised LTR with pseudo-labels via predictions.

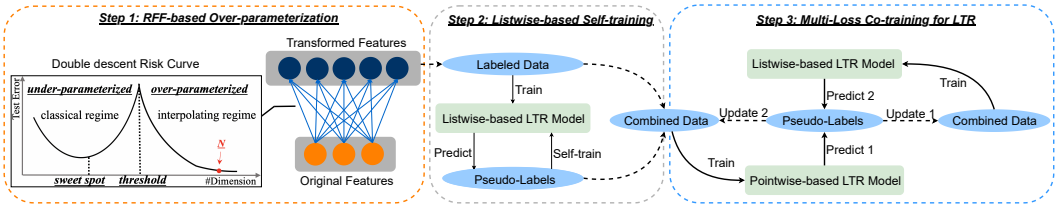


Fig. 2. The Pipeline of **COLTR** consisting of three steps: (1) *RFF-based over-parameterization*, (2) *Listwise-based Self-training*, and (3) *Multi-Loss Co-training for Semi-supervised LTR.*

3 COLTR: LTR WITH CO-TRAINED AND OVER-PARAMETERIZED MODELS

In this section, we present the technical details of our proposed algorithm **COLTR**. As illustrated in Figure 2, **COLTR** consists of three steps: (1) *RFF-based over-parameterization*, (2) *Listwise-based Self-training*, and (3) *Multi-Loss Co-training for LTR*. We first introduce *RFF-based over-parameterization* in section 3.1, and then describe *Listwise-based Self-training* and *Multi-Loss Co-training for LTR* together in section 3.2.

Algorithm 1: Random Fourier Feature with Gaussian Kernel

Output: the function $\mathbf{z}(x) : \mathcal{R}^m \rightarrow \mathcal{R}^N$.

1: Define Gaussian Distribution: $p(\omega) = (2\pi)^{-\frac{N}{2}} e^{-\frac{\|\omega\|_2^2}{2}}$;
2: Draw N i.i.d samples $\omega^1, \cdots, \omega^N \sim p(\omega)$;
3: Draw N i.i.d samples $b^1, \cdots, b^N \sim \mathcal{N}(0, 1)$;
4: Conduct $W = \left\{\omega^1, \cdots, \omega^N\right\}$ and $B = \left\{b^1, \cdots, b^N\right\}$;
5: Compute $\mathbf{z}(x) = \sqrt{\frac{2}{N}} \left[\cos\left(W^T x + B\right)\right]$;
6: **return** $\mathbf{z}(x)$;

3.1 RFF-based Over-parameterization

Given the overall set of queries $\mathcal{Q}$ and the set of all webpages $\mathcal{D}$, **COLTR** first obtains every possible query-webpage pair from the both datasets, denoted as (q_i, d_i^j) for $\forall q_i \in \mathcal{Q}$ 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), **COLTR** further extracts an m-dimensional feature vector $\boldsymbol{x}_{i,j}$ representing the features of the j^{th} webpage under the i^{th} query, using the pre-trained language models [22, 47].

Given $\boldsymbol{x}_{i,j} \in \mathcal{R}^m$, **COLTR** further maps the feature vector into an N-dimensional vector denoted $\boldsymbol{z}_{i,j} = \mathbf{z}(\boldsymbol{x}_{i,j})$ as using the feature transformation $\mathbf{z}(x)$. In this work, we use the transformation based on Random Fourier Features [31] to implement $\mathbf{z}(x)$, which is defined in **Algorithm 1**. Please be advised that the use of $\mathbf{z}(x)$ can map original features of LTR into a feature space of higher dimensions when $N \gg m$. With increasing number of dimensions N, the LTR model is being over-parameterized with more input features and would incorporate feature-wise *"double descent"* phenomenon of generalization errors in prediction [2, 3]. Through cross-validation on labeled set $\mathcal{T}$, **COLTR** determines the optimal setting of N to achieve the best generalization performance. Hence, with $\boldsymbol{z}_{i,j}$ for every query-webpage pair, the over-parameterized LTR model is expected to work in the interpolating regime [2] with superb generalization performance.

In this way, **COLTR** transforms query-webpage pairs in labeled and unlabeled datasets $\mathcal{T}$ and $\mathcal{T}'$ into two sets of labeled and unlabeled feature vectors $\mathcal{Z}^L = \{(\mathbf{z}_{i,j}, \boldsymbol{y}_j^i) | \forall (q_i, D_i, \boldsymbol{y}) \in \mathcal{T}$ and $\forall d_j^i \in D_i\}$ and $\mathcal{Z}^U = \{\mathbf{z}_{i,j} | \forall (q_i, D_i) \in \mathcal{T}'\}$, respectively.

3.2 Listwise-based Self-training and Multi-Loss Co-training for Learning to Rank

Given the labeled and unlabeled sets of feature vectors $\mathcal{Z}^L$ and $\mathcal{Z}^U$, **COLTR** further takes the next two steps in the pipeline to accomplish semi-supervised LTR. **Algorithm 2** lists the pseudo codes of the two steps of *Listwise-based Self-training* and *Multi-Loss Co-training for LTR*, respectively.

3.2.1 Listwise-based Self-training. First of all, **COLTR** trains a listwise LTR model LTR^{Li} using both $\mathcal{Z}^L$ and $\mathcal{Z}^U$ through self-training, where LTR^{Li} is first trained using $\mathcal{Z}^L$ through supervised learning. Then, LTR^{Li} predicts the ranking score for each feature vector in $\mathcal{Z}^U$ and *pseudo-labels* the feature vector with the prediction result. Later, **COLTR** combines $\mathcal{Z}^L$ with pseudo-labeled data $\mathcal{Z}^P$ and retrains LTR^{Li} using the *combined data* $\mathcal{Z}^C$.

3.2.2 Multi-Loss Co-training for LTR. Given the *combined data*, $\mathcal{Z}^L$ and $\mathcal{Z}^U$, **COLTR** (1) trains a pointwise model LTR^{Po} using $\mathcal{Z}^C$ and predicts pseudo-labels for every feature vector in $\mathcal{Z}^U$ using trained LTR^{Po}. Later, **COLTR** (2) updates $\mathcal{Z}^P$ with the prediction results of LTR^{Po} and combines $\mathcal{Z}^L$ with $\mathcal{Z}^P$ to obtain $\mathcal{Z}^C$. **COLTR** further (3) retrains LTR^{Li} using the $\mathcal{Z}^C$ and predicts ranking scores for each feature vector in $\mathcal{Z}^U$ using trained LTR^{Li}. Finally, **COLTR** (4) updates $\mathcal{Z}^P$ with the

Algorithm 2: Listwise-based Self-training and Multi-Loss Co-training for LTR

Input: labeled data $\mathcal{Z}^L$, unlabeled data $\mathcal{Z}^U$, the number of rounds for co-training C.
Output: Trained pointwise model LTR^{Po}

```
/*Listwise-based Self-training*/
Train listwise model LTR^Li on Z^L;
Z^P ← pseudo-labels Z^U using predictions of LTR^Li;
Z^C ← Z^P ∪ Z^L;
/*Multi-Loss Co-training for LTR*/
for i = 1 to C do
    Train pointwise model LTR^Po on Z^C;
    Z^P ← pseudo-labels Z^U using predictions of LTR^Po;
    Z^C ← Z^P ∪ Z^L;
    Train listwise model LTR^Li on Z^C;
    Z^P ← pseudo-labels Z^U using predictions of LTR^Li;
    Z^C ← Z^P ∪ Z^L;
end for
return LTR^Po;
```

prediction results of LTR^{Li} and combines $\mathcal{Z}^L$ with $\mathcal{Z}^P$ to obtain $\mathcal{Z}^C$. **COLTR** repeats above (1)–(4) steps with C rounds and returns the pointwise model LTR^{Po}.

3.2.3 Serving LTR with Pointwise Model. Given the pointwise model LTR^{Po} obtained through C rounds of co-training with the listwise model, **COLTR** adopts LTR^{Po} to serve the LTR tasks in Baidu search.

4 EXPERIMENTS

To demonstrate the effectiveness of our proposed model, we present extensive experiments in Baidu search engine system comparing with a large number of baseline methods. Firstly, we introduce the experimental details in terms of the dataset, evaluation methodology, competitor system and experimental settings. Then, we introduce the results of offline experiments. Finally, the online A/B Test performance shows the effectiveness of **COLTR** in Baidu search.

4.1 Dataset

We collect the dataset with 15,000 queries and over 770,000 query-webpage pairs from Baidu search engine system. The dataset is split into training set (12,000 queries), validation set (1,000 queries) and test set (2,000 queries), which contain 616,314 query-webpage pairs, 51,360 query-webpage pairs and 103,872 query-webpage pairs, respectively. For the semi-supervised learning experiments, we randomly select four ratios of labeled data from training set, where we utilize $\alpha = \{0.05, 0.1, 0.15, 0.2\}$ to represent the four ratios, respectively. Table 1 shows the statistics of the dataset.

4.2 Evaluation Methodology

To evaluate the performance of our proposed method, we use Normalized Discounted Cumulative Gain (NDCG) [18], which has been widely adopted to evaluate the relevance of in the context of ad-hoc search engine. Before introducing NDCG, we first introduce the Discounted Cumulative

Table 1. Statistics of the dataset.

Dataset	#Query	#Query-webpage pairs
Training Set	12,000	616,314
Validation Set	1,000	51,360
Test Set	2,000	103,872

Gain (DCG) as:

$$DCG_N = \sum_{i=1}^{N} \frac{G_i}{\log_2(i+1)}, \tag{3}$$

where G_i denotes the weight assigned to the document's label at position i. A higher G_i indicates that the webpage is more relevant to the query and correspondingly a better LTR model. However, due to the different lengths of various queries, it makes no sense to compare the DCG among them. Then, we utilize the following implementation of NDCG to take a mean across all scores as:

$$NDCG_N = \frac{DCG_N}{IDCG_N}, \tag{4}$$

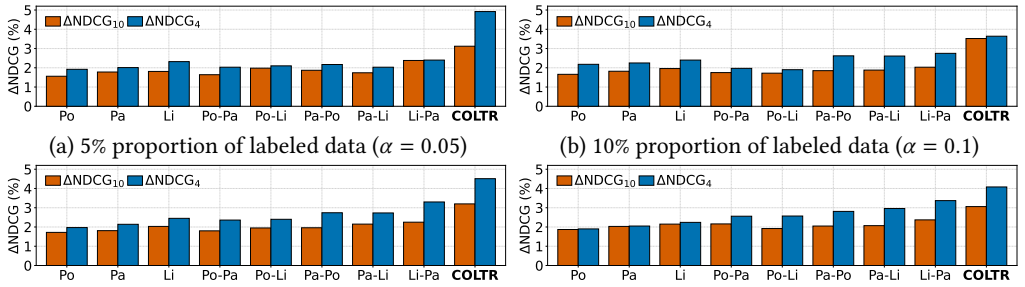
where $IDCG_N$ is the ideal order to normalize the scores. Moreover, the value of NDCG is in the range of $[0, 1]$. Similarly, a higher $NDCG_N$ indicates a better LTR model. ***From the perspective of research and business to evaluate the models' performance, we consider the NDCG of the top 10 and 4 ranking results, i.e.,*** $NDCG@10$ and $NDCG@4$.

4.3 Competitor Systems and Baselines

For all experiments, the baseline model is a ***pairwise-based self-trained LTR model without RFF-based over-parameterization***. Specifically, the baseline model adopts a self-trained LightGBM [20]-based LTR model with a L2 loss function, which has been deployed in Baidu search. Moreover, in order to demonstrate the effectiveness of **COLTR** sufficiently, we choose seven semi-supervised LTR models as the comparative models. All the competitor models are summarized as follows:

- **Base** is a *pairwise-based* self-trained LTR model without *RFF-based over-parameterization*.
- **Po** refers to a *RFF-overparameterized* self-trained LTR model with the pointwise loss function.
- **Pa** refers to a *RFF-overparameterized* self-trained LTR model with the pairwise loss function.
- **Li** refers to a *RFF-overparameterized* self-trained LTR model with the listwise loss function.
- **Po-Pa** refers to a *RFF-overparameterized multi-loss co-training* LTR model with the pointwise-to-pairwise setting.
- **Po-Li** refers to a *RFF-overparameterized multi-loss co-training* LTR model with the pointwise-to-listwise setting.
- **Pa-Po** refers to a *RFF-overparameterized multi-loss co-training* LTR model with the pairwise-to-pointwise setting.
- **Pa-Li** refers to a *RFF-overparameterized multi-loss co-training* LTR model with the pairtwise-to-listwise setting.
- **Li-Pa** refers to a *RFF-overparameterized multi-loss co-training* LTR model with the listwise-to-pairwise setting.

In this work, due to the restriction of business information disclosures, we only report $\Delta NDCG$ to measure the difference between our proposed model and the **Base** model.



(a) 5% proportion of labeled data ($\alpha = 0.05$) (b) 10% proportion of labeled data ($\alpha = 0.1$)

(c) 15% proportion of labeled data ($\alpha = 0.15$) (d) 20% proportion of labeled data ($\alpha = 0.2$)

Fig. 3. Offline comparative results ($\Delta NDCG_{10}$ and $\Delta NDCG_4$) of **COLTR** and baselines under various ratios of labeled data. We use "Po", "Pa" and "Li" to represent RFF-overparameterized *self-trained* LTR models of pointwise , pairwise and listwise respectively. We use "Po-Pa", "Po-Li", "Pa-Po", "Pa-Li", and "Li-Pa" to represent the RFF-overparameterized *multi-loss co-training* LTR models under pointwise-to-pairwise, pointwise-to-listwise, pairwise-to-pointwise, pairwise-to-listwise, and listwise-to-pairwise settings, where the terminology is based on the order of LTR models used for co-training, such that **COLTR** is in the listwise-to-pointwise ("Li-Po") setting.

4.4 Experimental Settings

In this work, all the offline experiments are implemented on *PaddlePaddle*[1] cloud platform with 64G Memory, 4 NVIDIA Tesla V100 GPU and 12T Disk. The online experiments are deployed on Baidu search engine system. We choose LightGBM, which is the most popular tree-based ranker, as the base ranking model with the number of trees as 200 and the learning rate as 0.01. The experiments consist of nine self-training and co-training models under four ratios of labeled data for ten rounds. Besides, for the *RFF-based over-parameterization* experiments, we set the ratio of the number of transformed dimension N and the number of original dimension m as N/m.

4.5 Offline Experimental Results

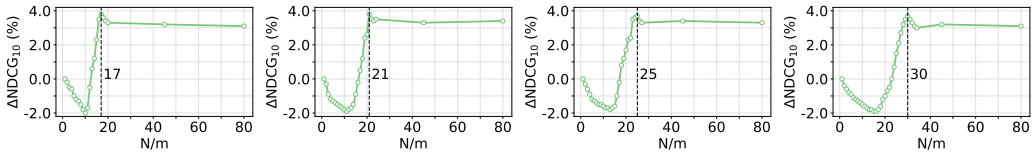
To comprehensively evaluate our proposed method, we conduct experiments to answer the following questions:

RQ1 How does **COLTR** perform compared with the baseline for LTR tasks?
RQ2 Which number of the transformed dimension can make LTR models gain the best performance under different ratios?
RQ3 Is the *RFF based over-parameterization* in **COLTR** necessary for improving performance?
RQ4 How does the number of rounds impact the performance of **COLTR**?

4.5.1 ***Comparative Results: RQ1.*** In Figure 3, we report the offline performance of **COLTR** compared with other baselines under four different ratios of labeled data on $\Delta NDCG_{10}$ and $\Delta NDCG_4$. For each semi-supervised learning model, we choose the model with the best performance of all validation rounds for testing.

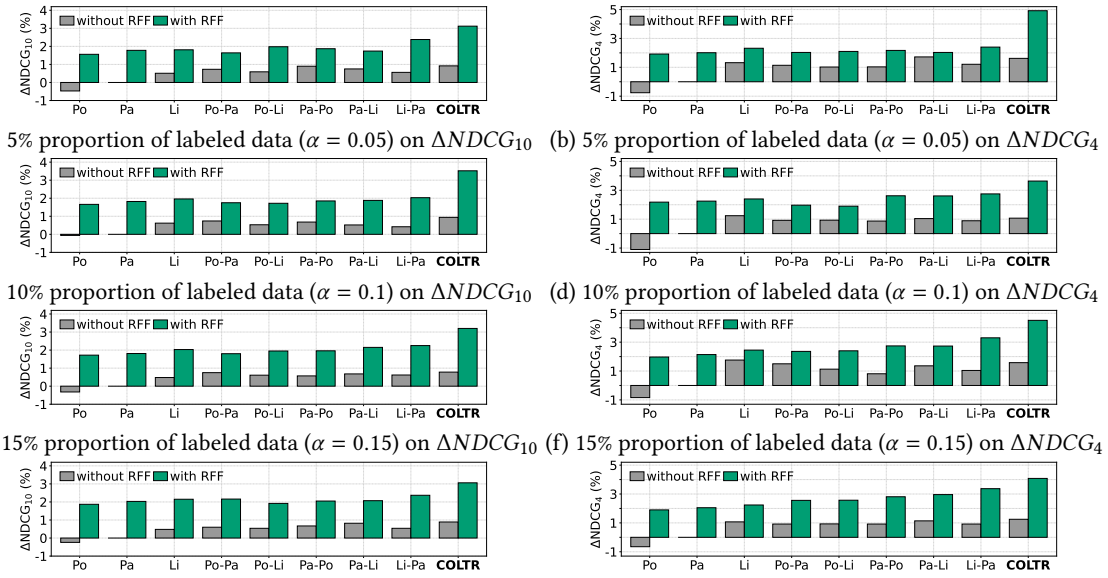
Intuitively, we could see that **COLTR** gains the best performance compared with other baselines under four ratios of labeled data on both two metrics. Specifically, **COLTR** achieves **4.92%** improvement on $\Delta NDCG_4$ under 5% ratio of labeled data. **COLTR** incorporates the diversity of prediction results of listwise model and pointwise model in a loop of multiple rounds. As the stronger learner, listwise model predicts more accurate pseudo-labels for pointwise model. Then the weaker model, pointwise model, generates relative inaccurate but diversity pseudo-labels to

[1]https://www.paddlepaddle.org.cn/



(a) $\alpha = 0.05$ (b) $\alpha = 0.1$ (c) $\alpha = 0.15$ (d) $\alpha = 0.2$

Fig. 4. Ablation studies of *RFF-based Over-parameterization* to choose the proper transformed dimensions under various ratios of labeled data on the validation set. We choose the values of N/m marked by dashed lines.



(a) 5% proportion of labeled data ($\alpha = 0.05$) on $\Delta NDCG_{10}$ (b) 5% proportion of labeled data ($\alpha = 0.05$) on $\Delta NDCG_4$

(c) 10% proportion of labeled data ($\alpha = 0.1$) on $\Delta NDCG_{10}$ (d) 10% proportion of labeled data ($\alpha = 0.1$) on $\Delta NDCG_4$

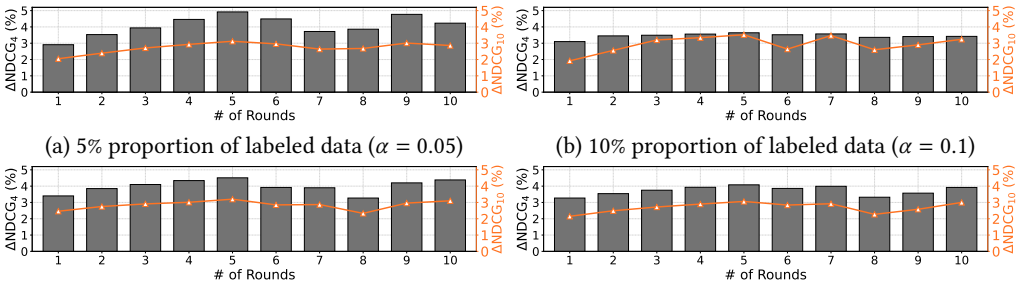
(e) 15% proportion of labeled data ($\alpha = 0.15$) on $\Delta NDCG_{10}$ (f) 15% proportion of labeled data ($\alpha = 0.15$) on $\Delta NDCG_4$

(g) 20% proportion of labeled data ($\alpha = 0.2$) on $\Delta NDCG_{10}$ (h) 20% proportion of labeled data ($\alpha = 0.2$) on $\Delta NDCG_4$

Fig. 5. Ablation studies of *RFF-based Over-parameterization* for **COLTR** and baselines under different ratios of labeled data.

train the stronger model. In such *Multi-Loss Co-training* mechanism, **COLTR** gains the significant performance. Moreover, there are two findings in the comparative results. First, when $\alpha = 5\%$, **COLTR** significantly outperforms other self-traning models and co-training models compared with other three ratios of labeled data on $\Delta NDCG_4$. Next, although co-training models can not obtain the performance like **COLTR**, some models, such as "Pa-Po", "Pa-Li","Li-Pa", also outperform three self-training models when $\alpha = \{0.1, 0.15, 0.2\}$ on $\Delta NDCG_{10}$.

4.5.2 ***Ablation Study: RQ2*** . In this study, we conduct a series of experiments to investigate the proper value of transformed dimension under four kinds of ratios of labeled data for LTR tasks. In order to show the results clearly, we choose the performance on $\Delta NDCG_{10}$ instead of "test error" to present the curves. Intuitively, Figure 4 represents the margin curves for the pointwise model under four different ratios of labeled data on validation set. In Figure 4 (a), we present that the pointwise model gains the best performance when $N/m = 17$ under 5% labeled training data. As depicted in the figure, at the beginning of the curve, the value of $\Delta NDCG_{10}$ first rises and then falls in the



(a) 5% proportion of labeled data ($\alpha = 0.05$)

(b) 10% proportion of labeled data ($\alpha = 0.1$)

(c) 15% proportion of labeled data ($\alpha = 0.15$)

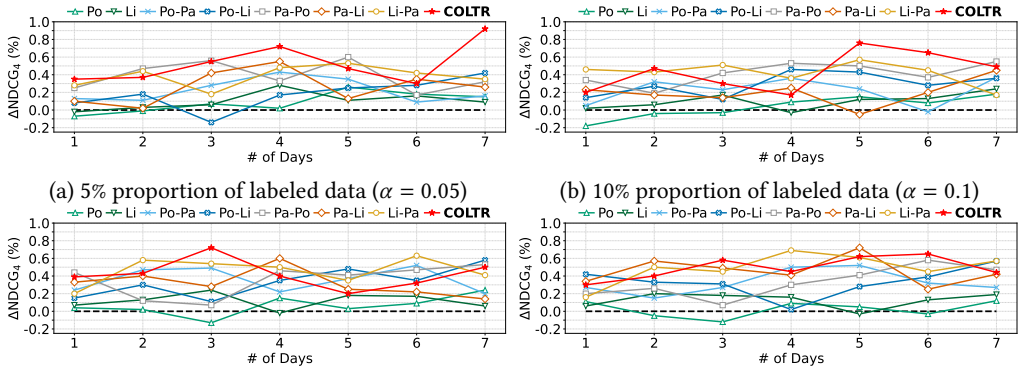
(d) 20% proportion of labeled data ($\alpha = 0.2$)

Fig. 6. Performance ($\Delta NDCG_4$ and $\Delta NDCG10$) of **COLTR** under four ratios of labeled data in ten rounds on the validation set.

"classical regime". Once the value of N/m exceeds the threshold, the pointwise model performs well again and gains the best performance at $N/m = 17$ in the "interpolating regime". For different ratios of labeled data, the value of N/m is various in terms of the number of labeled samples. Figure 4 (b) shows the most proper value of N/m is 21 under 10% of labeled data. Similarity, when $\alpha = 0.15$, the chosen value of N/m is 25. For 20% proportion of label data, the chosen value of N/m is 30. We can see that transforming the original query-webpage vector into the proper dimension could improve the performance of downstream LTR models in the newly-fashioned "interpolating regime".

4.5.3 ***Ablation Results: RQ3***. In order to demonstrate the effectiveness of *RFF-based Over-parameterization* part of **COLTR**, we conduct a series of ablation studies. Figure 5 shows the ablation studies of *RFF-based Over-parameterization* for **COLTR** and baselines under various ratios of labeled data. As we can see, all the semi-supervised learning models with *RFF-based Over-parameterization* could obtain better performance compared with semi-supervised learning models without *RFF-based Over-parameterization*. As depicted in Figure 5 (b), *RFF-based Over-parameterization* achieves the improvement with **3.30%** for **COLTR** in average under 5% proportion of labeled data on $\Delta NDCG_4$, which is the largest improvement of **COLTR**. Besides, the pointwise-based self-training model with *RFF-based over-parameterization* obtains the improvement with **3.38%** in average, which is the largest improvement among all experiments. Semi-supervised learning LTR model without *RFF-based Over-parameterization* takes a small number of hand-crafted features and is based on tree-based models and their derivatives to pursue high throughput under concurrency poorly. Obviously, *RFF-based Over-parameterization* could tackle that issue well.

4.5.4 ***Parameter Sensitivity: RQ4***. In Figure 6, we report the studies of **COLTR** under various ratios of labeled data in ten co-training rounds on the validation set. The results show that **COLTR** gains the best performance at the 5th round in all experiments. For each round, the listwise model is trained on the combined data and generates pseudo-labels. Next, the pseudo-labeled data is combined with the labeled data. The pointwise model is trained on the combined data and generates the results on $\Delta NDCG$. As we can see in Figure 6 (a), with the number of rounds increasing, **COLTR** gains a rising performance and obtains the best performance at the 5th round. Next, the performance starts to decrease and is in a state of shock. Finally, the best results can be obtained at the 5th round. The other experiments have the similar phenomenons like Figure 6 (a). Therefore, we choose the trained **COLTR** at the 5th round for online experiments.



(a) 5% proportion of labeled data ($\alpha = 0.05$)

(b) 10% proportion of labeled data ($\alpha = 0.1$)

(c) 15% proportion of labeled data ($\alpha = 0.15$)

(d) 20% proportion of labeled data ($\alpha = 0.2$)

Fig. 7. Online comparative performance ($\Delta NDCG_4$) of **COLTR** and baselines for 7 days (t-test with $p < 0.05$ over the baseline). **COLTR** could boost the performance compared with the online base system in all days, which demonstrates **COLTR** is practical for improving the performance of Baidu search engine.

4.6 Online Experimental Results

To demonstrate the effectiveness of **COLTR**, we conduct a series of online A/B tests with real-world web traffics and compare it with the baseline models on Baidu search engine. According to the offline experimental results, we choose the trained **COLTR** under four various ratios of labeled data in the 5th round and conduct the online experiments with 0.5% real-world web traffics of Baidu search. The online A/B tests last for 7 days from January 3rd., 2022 to January 9th., 2022.

Figure 7 illustrates the comparison of **COLTR** with the baselines on $\Delta NDCG_4$. Firstly, **COLTR** could boost the performance compared with the online base system in all days, which demonstrates **COLTR** is practical for improving the performance of Baidu search engine. Furthermore, we can find that **COLTR** achieves significant improvements on Baidu search engine. Specifically, we observe that **COLTR** trained on 5% proportion of labeled data outperforms the online base model by a large margin on $\Delta NDCG_4$ with **0.92%** relative improvement. The largest improvements of trained **COLTR** on 10%, 15% and 20% proportion of labeled data are 0.76%, 0.72% and 0.65% on $\Delta NDCG_4$, respectively. These significant improvements reveal the effectiveness of **COLTR**. Finally, we also compare **COLTR** with other semi-supervised learning models used in our offline experiments. We notice that **COLTR** outperforms several semi-supervised learning models in all days, which proves that **COLTR** is more useful and sound to improve the accuracy for real-world online search engine. Basically, the online performance is consistent with our offline experiment result.

5 RELATED WORK

In this section, we review and discuss related works from the following three aspects: (1) *Learning to Rank*, (2) *Over-parameterization Method* and (3) *Semi-supervised Learning*.

5.1 Learning to Rank

To improve user experience in terms of searching , ranking the retrieved contents is a key step, where the LTR model plays a critical role. According to the loss function, we could categorize the LTR models into three families: pointwise [13, 21], pairwise [19, 41] and listwise [8, 38]. The pointwise model (e.g., McRank [21]) formulates the ranking problem into regression tasks to fit the labels of query-webpage pairs. The pairwise model (e.g., GBrank [41]) converts two documents into a document pair and recasts the LTR tasks as binary classification problems. It pays more attention

to find the best one in each document pair. The listwise model (e.g., Softrank [38]) treats the whole document list as a sample and directly optimizes the evaluation metrics, such as the utilized metric in this work, i.e., NDCG [18]. In general, listwise models can gain the best performance among the three LTR methods. However, pairwise and pairwise models are generally deployed in real-world applications for being easy to apply and having less computational complexity. Moreover, *In our work, **COLTR** considers the divergence between the prediction results of listwise and pointwise models, and incorporates such divergence to improve co-training.*

5.2 Over-parameterization Method

Recently, the methods of balancing under-parameterization and over-parameterization have attracted growing research interests [3]. [37] proposes a parameter learning based method to tackle LTR tasks. Moreover, there are some mixture feature transformation mechanisms are proposed to improve the performance[35, 46]. Nevertheless, under appropriate settings, over-parameterization could gain better performance on test data in the newly-fashioned "interpolation regime" with the *double descent curve*. There are several over-parameterization methods [2, 3] which have superb performance. Random Fourier Feature [31] adopts a kernel technique to generate features for most inner product-based models, which has gained great improvements. ***COLTR** follows this line of research and is the first to leverage RFF-based over-parameterization to improve LTR models.*

5.3 Semi-supervised Learning

Nowadays, semi-supervised learning methods have been widely adopted in machine learning tasks, such as classification, regression, etc. Two effective categories of semi-supervised learning methods are self-training [36, 44] and co-training [5]. For self-training, the basic idea is to generate pseudo-labels for unlabeled data and improve the performance with pseudo-labeled data [33]. The learning process of semi-supervised learning could follow the four-step strategy: (1) training a model with labeled data; (2) using the trained model to generate pesudo-labels; (3) adding the pesudo-labeled data into the label data and train another model with the combined data; (4) retraining the former model with labeled data. [1, 12, 16] present the effectiveness of co-training and gain significant improvements. Moreover, [37] proposes a semi-supervised learning method for LTR tasks with preference regularization. Therefore, it is illustrated that co-training method is useful for web-scale search. *In this work, **COLTR** adopts co-training* [14, 24, 43] *for semi-supervised LTR, where listwise and pointwise models teach each other based on pseudo-labels via predictions.*

6 CONCLUSION

To incorporate the large amount of unlabeled data archived in search engine, we design, implement and deploy **COLTR** – namely *Co-trained and Over-parameterzed LTR* system at Baidu search for learning to rank tasks under semi-supervised settings in this work. **COLTR** consists of three steps: (1) *RFF-based over-parameterization* enabling representation learning in the interpolating regime, (2) *Listwise-based Self-training* initializing pseudo-labels of unlabeled samples with a self-supervised listwise LTR model, and (3) *Multi-Loss Co-training for LTR* making pointwise and listwise models learn from each other. To the best of our knowledge, this work is the first to study semi-supervised training for LTR models with labeled/unlabeled query-webpage pairs by addressing the mathematical phenomenon of interpolating in LTR tasks and the diversity of LTR models trained with various loss functions. To verify the effectiveness of **COLTR**, we conduct extensive offline and online experiments compared with a large number of baseline methods. Offline experiment results show that **COLTR** could achieve significant gain over baselines on $\Delta NDCG_4$ under various ratios of labeled samples. Furthermore, **COLTR** significantly boosts the online ranking performance in real-world applications, which is consistent with the offline results. In the future, we plan

to investigate the interpretation of the feature transformation and the model for LTR tasks. In particular, we attempt to study the interpretability of neural LTR approaches and their feature transformation mechanisms for web-scale search.

REFERENCES

[1] Steven Abney. 2002. Bootstrapping. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 360–367.

[2] Mikhail Belkin. 2021. Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation. *Acta Numerica* 30 (2021), 203–248.

[3] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. 2019. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences* 116, 32 (2019), 15849–15854.

[4] Mikhail Bilenko, Sugato Basu, and Raymond J Mooney. 2004. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the twenty-first international conference on Machine learning*. 142–149.

[5] Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*. 92–100.

[6] Sebastian Bruch. 2021. An alternative cross entropy loss for learning-to-rank. In *Proceedings of the Web Conference 2021*. 118–126.

[7] Christopher Burges, Robert Ragno, and Quoc Le. 2006. Learning to rank with nonsmooth cost functions. *Advances in neural information processing systems* 19 (2006), 193–200.

[8] 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.

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

[10] Hugh Chipman, Edward George, and Robert McCulloch. 2006. Bayesian ensemble learning. *Advances in neural information processing systems* 19 (2006).

[11] Yanwen Chong, Yun Ding, Qing Yan, and Shaoming Pan. 2020. Graph-based semi-supervised learning: A review. *Neurocomputing* 408 (2020), 216–230.

[12] Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

[13] 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.

[14] Luca Didaci and Fabio Roli. 2006. Using co-training and self-training in semi-supervised multiple classifier systems. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. Springer, 522–530.

[15] Miao Fan, Zehao Dou, Mingming Sun, Ping Li, and Haifeng Wang. 2021. Method and device for information retrieval, device and computer readable storage medium. US Patent 11,055,374.

[16] Sally Goldman and Yan Zhou. 2000. Enhancing supervised learning with unlabeled data. In *ICML*. Citeseer, 327–334.

[17] Jizhou Huang, Haifeng Wang, Wei Zhang, and Ting Liu. 2020. Multi-task learning for entity recommendation and document ranking in web search. *ACM Transactions on Intelligent Systems and Technology (TIST)* 11, 5 (2020), 1–24.

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

[19] 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.

[20] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems* 30 (2017), 3146–3154.

[21] 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.

[22] 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*. 3365–3375.

[23] Claudio Lucchese, Cristina Ioana Muntean, Franco Maria Nardini, Raffaele Perego, and Salvatore Trani. 2017. Rankeval: An evaluation and analysis framework for learning-to-rank solutions. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1281–1284.

[24] Siyuan Qiao, Wei Shen, Zhishuai Zhang, Bo Wang, and Alan Yuille. 2018. Deep co-training for semi-supervised image recognition. In *Proceedings of the european conference on computer vision (ECCV)*. 135–152.

[25] 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.

[26] 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)*. IEEE, 1790–1801.

[27] An Qin, Yuan Yuan, Dai Tan, Pengyu Sun, Xiang Zhang, Hao Cao, Rubao Lee, and Xiaodong Zhang. 2017. Feisu: Fast Query Execution over Heterogeneous Data Sources on Large-Scale Clusters. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. IEEE, 1173–1182.

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

[29] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. 2010. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval* 13, 4 (2010), 346–374.

[30] Zhen Qin, Le Yan, Honglei Zhuang, Yi Tay, Rama Kumar Pasumarthi, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2020. Are Neural Rankers still Outperformed by Gradient Boosted Decision Trees?. In *International Conference on Learning Representations*.

[31] Ali Rahimi, Benjamin Recht, et al. 2007. Random Features for Large-Scale Kernel Machines.. In *NIPS*, Vol. 3. Citeseer, 5.

[32] Elaheh Raisi and Bert Huang. 2017. Co-trained ensemble models for weakly supervised cyberbullying detection. In *NIPS Workshop on Learning with Limited Labeled Data*.

[33] Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. 2005. Semi-supervised self-training of object detection models. *WACV/MOTION, 2* (2005).

[34] Burr Settles. 2011. From theories to queries: Active learning in practice. In *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010*. JMLR Workshop and Conference Proceedings, 1–18.

[35] Mehak Sheikh, Muhammad Adeel Asghar, Ruqia Bibi, Muhammad Noman Malik, Mohammad Shorfuzzaman, Raja Majid Mehmood, and Sun-Hee Kim. 2022. DFT-Net: Deep Feature Transformation Based Network for Object Categorization and Part Segmentation in 3-Dimensional Point Clouds. *Sensors* 22, 7 (2022), 2512.

[36] Amarnag Subramanya and Partha Pratim Talukdar. 2014. Graph-based semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 8, 4 (2014), 1–125.

[37] Martin Szummer and Emine Yilmaz. 2011. Semi-supervised learning to rank with preference regularization. In *Proceedings of the 20th ACM international conference on Information and knowledge management*. 269–278.

[38] 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.

[39] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. 2020. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10687–10698.

[40] Shiqi Zhao, Haifeng Wang, and Ting Liu. 2010. Paraphrasing with search engine query logs. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*. 1317–1325.

[41] 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.

[42] Zhi-Hua Zhou. 2021. Ensemble learning. In *Machine learning*. Springer, 181–210.

[43] Zhi-Hua Zhou, Ming Li, et al. 2005. Semi-supervised regression with co-training.. In *IJCAI*, Vol. 5. 908–913.

[44] Xiaojin Zhu and Andrew B Goldberg. 2009. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning* 3, 1 (2009), 1–130.

[45] Xiaojin Jerry Zhu. 2005. Semi-supervised learning literature survey. (2005).

[46] Honglei Zhuang, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2020. Feature transformation for neural ranking models. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1649–1652.

[47] 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.