COLTR: Semi-supervised Learning to Rank with Co-training and Over-parameterization for Web 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
HAO LIU, Hong Kong University of Science and Technology (Guangzhou), China
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 search 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.

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

*__This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.__

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; Hao Liu, Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China; 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

ACM Reference Format:
Yuchen Li, Haoyi Xiong, Qingzhong Wang, Linghe Kong, Hao Liu, Haifang Li, Jiang Bian, Shuaiqiang Wang, Guihai Chen, and Dawei Yin, Dejing Dou. 2023. **COLTR**: Semi-supervised Learning to Rank with Co-training and Over-parameterization for Web Search. 1, 1 (January 2023), 20 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Incorporating with billions of population and trillions of web content, search service like Google, Baidu, and Bing satisfy the daily searching needs of users. The rapid growth of web information and internet users surge the needs of web search engine. Nowadays, orchestrating with trillions of webpages archived and indexed for search, a large-scale industrial search engine serves hundreds of millions of daily active users and handles billions of queries per day. In addition to users, webpages, and computing resources, to improve the web service, a great number of most advanced technologies have been invented, ranging from pre-trained language models for content/query understanding [1–3] and ranking [4], domain-specific recommender systems for personalized recommendation [5], online query-Ads matching for sponsored search [6, 7] and advanced infrastructures with software/hardware co-design [8, 9] for handling web-scale traffic of online search.

With a query (e.g., a string of texts) input by the user, a search engine needs first to extract keywords/phrases from the query and recognize the user's intention. Given the extracted keywords or phrases, the search engine evaluates the relevance between the query and webpages, and after that, retrieves a list of webpages that are relevant from a database of trillions of webpages. Further, the search engine ranks the retrieved webpages based on their contents and click-through rates. In the response to the query, the search engine tops the most relevant webpages. 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 [10]. 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* [11, 12], 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 [12–15], 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 [16]. The performance of self-training however would be bottlenecked for LTR tasks by issues as follows:

- *Low-Cost Representation Learning*. For practical web search, LTR tasks usually use a small number of features (e.g., from several dozens to hundreds of features) [17–19] and are based

on statistical learners [20–23], which is incapable of representation learning from given features. Even when deep models, such as pre-trained language models, are incorporated for representation learning from raw queries/webpages, LTR models are usually not trained with the deep feature extractors in an end-to-end manner for industry practices. Actually, a post-hoc approach connecting deep feature extractors with RankSVM [24]/GBRank [25] is more preferred [26]. It is because language features would rarely change in a short period but the internet interests (news, celebrities, etc.) shift in an extremely fast manner. Thus, web-scale search engines request to re-train LTR models using the most recent collection of queries/webpages and update the online LTR models frequently. In this way, there needs to transform *Learning to Rank* (LTR) to a *low-cost representation learning* task with statistical learners.

- *Diversifying Noisy Supervision Signals.* Yet another problem of self-training is over-fitting to 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 an LTR model trained in the previous round. One way to solve the problem is to co-train the LTR model with multiple classifiers [27–29], so as to incorporate the diversity of prediction outputs from multiple classifiers in an ensemble learning fashion [30, 31]. It has been found that strong learners could be trained with limited labeled samples by 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 [33] 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 [34] 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 an 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 of enabling *low-cost statistical 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 [33, 34] in LTR tasks and the diversity of LTR models trained with various losses (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 a semi-supervised manner. Given a query and indexed webpages, **COLTR** relies on a pre-trained language models based retrieval method to pickup the candidates of webpages for ranking, constructs their LTR features using the outputs of

language models, and predicts the order of webpages among the candidates using LTR models. To train the LTR models, **COLTR** consists of three steps: (1) *RFF-based over-parameterization* that pushes the limits of representation learning to interpolating regime [34], (2) *Listwise-based Self-training* that initializes pseudo-labels of unlabeled samples using the predictions of an 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. Note that **COLTR** restores the RFF-transformed representation of LTR features for either queries or webpages as the immediate results of ranking and accelerates the inference procedure for online ranking accordingly.

- We deploy **COLTR** at Baidu Search[1] and evaluate the proposed algorithm using both offline experiments and online A/B Test 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 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*.

2.1 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 [35, 36], pairwise [20, 24] and listwise [37, 38]. The pointwise model (e.g., McRank [36]) formulates the ranking problem into regression tasks to fit the labels of query-webpage pairs. The pairwise model (e.g., RankNet[39]) converts two documents into a document pair and recasts the LTR tasks as binary classification problems. It pays more attention to finding the best one in each document pair. The listwise model (e.g., Softrank [37]) 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 [40–42]. 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. *In our work, **COLTR** considers the divergence between the prediction results of listwise and pointwise models and incorporates such divergence to improve co-training.*

2.2 Over-parameterization Method

Recently, the methods of balancing under-parameterization and over-parameterization have attracted growing research interests [33]. [43] proposes a parameter learning-based method to tackle LTR tasks. Moreover, there are some mixture feature transformation mechanisms are proposed to improve the performance[44, 45]. Nevertheless, under appropriate settings, over-parameterization could gain better performance on test data in the newly-fashioned "interpolation regime" with the

[1]https://www.baidu.com/

double descent curve. There are several over-parameterization methods [33, 34] which have superb performance. Random Fourier Feature [46] 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.*

2.3 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 [12, 14] and co-training [47]. For self-training, the basic idea is to generate pseudo-labels for unlabeled data and improve the performance with pseudo-labeled data [48]. 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 pseudo-labels; (3) adding the pseudo-labeled data into the label data and training another model with the combined data; (4) retraining the former model with labeled data. [49–51] present the effectiveness of co-training and gain significant improvements. Moreover, [43] proposes a semi-supervised learning method for LTR tasks with preference regularization. Therefore, it is illustrated that the co-training method is useful for web-scale search. *In this work,* ***COLTR*** *adopts co-training* [27–29] *for semi-supervised LTR, where listwise and pointwise models teach each other based on pseudo-labels via predictions.*

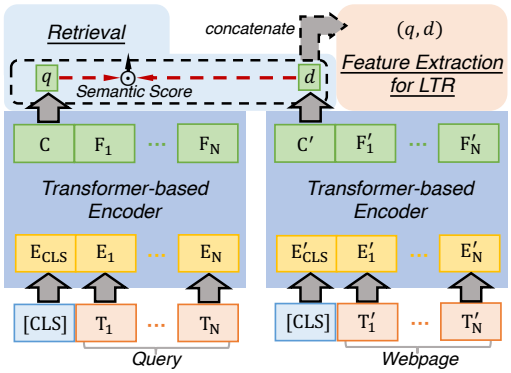


Fig. 1. The Process of Feature Extraction and Retrieval with ERNIE-based Semantic Retrieval Model. The Transformer-based encoders take the tokenized sequence of the query and webpage as the input and outputs encoded query and webpage embeddings, which are used to do semantic retrieval and conduct the query-webpage pair feature for LTR.

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

In this section, we present the design of **COLTR**. We first introduce the settings of **COLTR**, including query-webpage feature construction, retrieval & ranking candidates generation, and LTR problem formulation for **COLTR**. Then, we introduce the algorithms design of our proposed model **COLTR**. Table 1 lists the notations and their definitions.

3.1 LTR Settings for COLTR

Given the massive webpages archived and indexed (please see also in Section 4 for the deployment of **COLTR**, where storage & indexing would be introduced), compared with traditional approaches, such as text matching, **COLTR** leverages pre-trained language models [52] based semantic retrieval

Table 1. List of Notations

Notation	Definition
[CLS]	pseudo token for the subsequent matching
$\{T_1, \ldots, T_N\}$	tokenized sequence of the raw query
$\{T'_1, \ldots, T'_N\}$	tokenized sequence of the raw webpage
$\{F_1, \ldots, F_N\}$	encoded embedding of the query
$\{F'_1, \ldots, F'_N\}$	encoded embedding of the webpage
$\boldsymbol{Q}$	set of search query
$\mathcal{D}$	set of all archived webpages
q_i	the i^{th} query in $\boldsymbol{Q}$
d_i	the i^{th} webpage in $\mathcal{D}$
D_i	the set of relevant webpages for q_i
$\boldsymbol{y}_i$	set of ranking scores for q_i
y_j^i	ranking score of the j^{th} webpage for q_i
d_j^i	the j^{th} webpage of D_i
$\mathcal{T}$	set of query-webpage pairs with ranking scores
$\mathcal{T}'$	set of unlabeled query-webpage pairs
$\boldsymbol{x}_{i,j}$	feature vector of the query-webpage pair (q_i, d_i^j)
m	original dimensions of feature vectors
$\boldsymbol{z}_{i,j}$	transformed feature vector
N	transformed dimensions of feature vectors
$\mathcal{Z}^L$	set of transformed labeled query-webpage pairs
$\mathcal{Z}^U$	set of transformed unlabeled query-webpage pairs
$\mathcal{Z}^P$	set of pseudo-labeled data
$\mathcal{Z}^C$	set of combined data
LTR^{Po}	pointwise-based LTR model
LTR^{Li}	listwise-based LTR model
C	the number of rounds for co-training

algorithms to conduct an effective and efficient online webpages retrieval with given queries, and provide features extracted from webpages/queries for LTR.

*3.1.1 Retrieval and Ranking Candidates for **COLTR**.* Specifically, we adopt the ERNIE-based [53] semantic retrieval model to enhance the conventional retrieval approach [54]. For conventional retrieval, it performs numerous operations on query texts, such as word segmentation and stop-word filtering, utilizes term indexes to accomplish keyword matching, and gets a set of relevant webpages. Meanwhile, the retrieval method based on pre-trained models first leverages an ERNIE transformer module to obtain the embeddings of queries and webpages. More specifically, the transformer encoder [55] first takes the tokenized sequence of the raw query or webpage, i.e., {[CLS], $T_1, \ldots, T_N$} or {[CLS], $T'_1, \ldots, T'_N$}, as the input, where the pseudo token [CLS] aggregates data in the encoder for the subsequent matching, and outputs an encoded embedding of the query or webpage, i.e., {C, $F_1, \ldots, F_N$} or {C, $F'_1, \ldots, F'_N$} in Figure 1. Note that, the retrieval model together with the ERNIE module is end-to-end trainable given a bucket of feedback, such as click-throughs, dwell time, and the raw text-to-text matching results by conventional retrievals, as supervision signals [26]

Given a query and a webpage, the ERNIE-based retrieval estimates a semantic score between the query and webpage through computing the cosine similarity or inner-product between their embeddings. Then, under a query, **COLTR** tops the webpages with highest semantic scores as the results of retrieval and also the **candidates for webpages ranking**. Then **COLTR** forwards embeddings of the query and the top-retrieved webpages (as well as their semantic scores and other statistical features, e.g., click-through of webpages and etc.) as the feature inputs for LTR.

*3.1.2 Semi-supervised LTR Problems for **COLTR**.* 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 $q_i \in \mathcal{Q}$, (as discussed in Section 3.1.1) 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 [4, 18] 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).

Learning Objective of LTR. 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.

(1) 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.
(2) 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[56], 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[40] by swapping the positions of two webpages.
(3) 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 [57].

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.

3.2 LTR Algorithms of 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*. In the technical detail section,

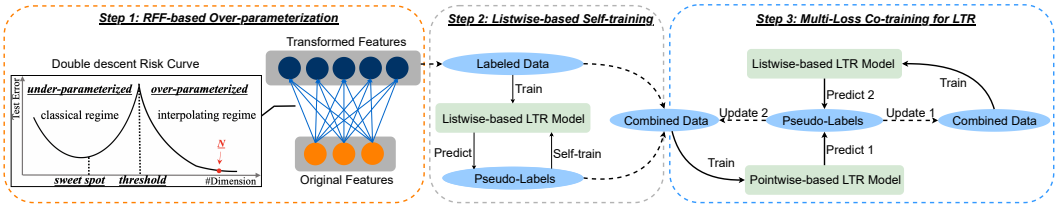


Fig. 2. The Pipeline of **COLTR** consisting of three steps: (1) *RFF-based over-parameterization*, (2) *Listwise-based Self-training*, and (3) *Multi-Loss Co-training for Semi-supervised LTR*.

Algorithm 1: Random Fourier Feature with Gaussian Kernel

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

we first introduce *RFF-based over-parameterization* in section 3.2.1, and then describe *Listwise-based Self-training* and *Multi-Loss Co-training for LTR* together in section 3.2.2.

3.2.1 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 [4, 26]. For more details about feature extraction, please refer to Section 3.1.1 and Figure 1.

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 [46] 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 the 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 [33, 34]. 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 [34] 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.

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}
begin
  /* Listwise-based Self-training */
  Train listwise model LTR^{Li} on $\mathcal{Z}^L$;
  $\mathcal{Z}^P \leftarrow$ pseudo-labels $\mathcal{Z}^U$ using predictions of LTR^{Li};
  $\mathcal{Z}^C \leftarrow \mathcal{Z}^P \cup \mathcal{Z}^L$;
  /* Multi-Loss Co-training for LTR */
  for *i = 1 to C* **do**
    Train pointwise model LTR^{Po} on $\mathcal{Z}^C$;
    $\mathcal{Z}^P \leftarrow$ pseudo-labels $\mathcal{Z}^U$ using predictions of LTR^{Po};
    $\mathcal{Z}^C \leftarrow \mathcal{Z}^P \cup \mathcal{Z}^L$;
    Train listwise model LTR^{Li} on $\mathcal{Z}^C$;
    $\mathcal{Z}^P \leftarrow$ pseudo-labels $\mathcal{Z}^U$ using predictions of LTR^{Li};
    $\mathcal{Z}^C \leftarrow \mathcal{Z}^P \cup \mathcal{Z}^L$;
  end
  return LTR^{Po};
end

3.2.2 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.

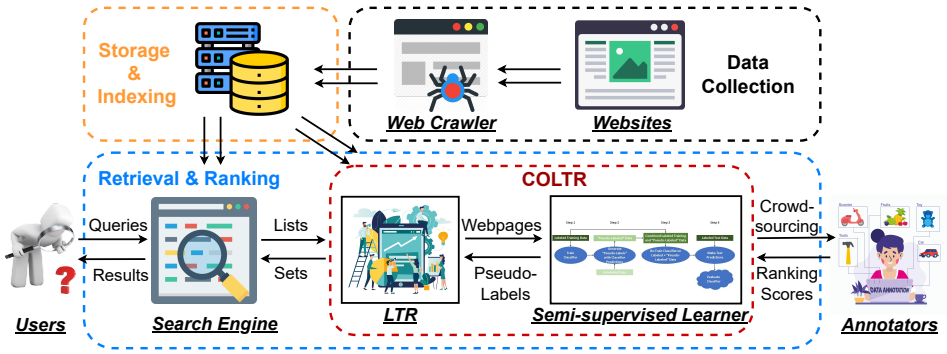


Fig. 3. The Overview of Baidu Search with **COLTR** Deployed.

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* $\mathcal{Z}^C$.

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 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 uses the pointwise model LTR^{Po} to serve online ranking.

3.3 Discussion on COLTR

In this work, we design LTR algorithms and **COLTR** on top of a pre-trained language models-based retrieval method [26], which supplies the candidates for webpage ranking and the features of LTR. The retrieval models together with ERNIE transformers are end-to-end trained incorporating both self-supervision of language models and some supervision signals, including text-to-text matching, click-throughs and dwell time. **COLTR** however doesn't include LTR models in the end-to-end training with the language models, primarily due to the cost reason. In fact, the requests to re-training LTR models would be highly frequent due to the fast shift the internet interests, however, the language models would be updated infrequently as the essentials of languages would not change rapidly. In this way, **COLTR** can train and re-train LTR models in a low cost while enjoying the features extracted from language models.

4 DEPLOYMENT OF COLTR

In this section, we introduce the deployment settings of **COLTR** at Baidu Search. As shown in Figure 3, the industrial search engine is essentially with three stages as follows: (1) *Data collection*, (2) *Storage & Indexing* and (3) *Retrieval & ranking*.

4.1 Data Collection

As a fully-automated search engine, Baidu search uses software that explores the web on a regular basis to find sites to be added to the database. In fact, the vast majority of sites listed in the results are not manually submitted for inclusion but are found and added automatically when the web crawler crawls the web. As a fully automated search engine, Baidu utilizes software to continuously browse the web in search of new websites to add to its database. In actuality, the vast majority of the websites that appear in the results are not manually added. On the contrary, they are discovered and uploaded automatically as the web crawler browses the internet. 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 the real-time web traffics of the Baidu search engine. Note that once a new or updated webpage is fetched, the Web Crawler would first parse the contents, find all possible links, and add them into the list for potential screening.

4.2 Storage & Indexing

Given the massive webpages downloaded from the web, Baidu search stores these contents in distributed archival storage systems with Fatman [58] and builds efficient indices for high-performance search based on DirectLoad [59]. While Fatman [58] could significantly reduce 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 [59] balances loads of indexing over these data centers with superb I/O efficiency using novel key-value operations and in-memory computation.

4.3 Retrieval & Ranking

For all webpages in the storage, as was mentioned in Section 3.1, Baidu search leverages an ERNIE-based encoder to compute the embeddings and conducts an embedding database, which will be used in the retrieval stage. Please refer to Section 3.1 for details for webpages retrieval.

In terms of webpages ranking in online settings, as was mentioned in Section 3.2, **COLTR** totally obtains three models LTR^{Po}, LTR^{Li}, and LTR^{Pa} through C rounds of co-training via pointwise, listwise and pairwise losses, respectively. **COLTR** adopts LTR^{Po} to serve the online ranking tasks. More specifically, after the training procedure of LTR, **COLTR** restored the RFF-transformation of LTR features for either webpages or queries as the immediate results of online ranking. For online inference, given a query and a webpage for online ranking, **COLTR** pickups their RFF-transformed representations and passes them to the GBM for inference in a fast manner at Baidu Search.

5 EXPERIMENTS

To demonstrate the effectiveness of our proposed model, we present extensive experiments compared 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** at Baidu Search.

5.1 Dataset

We evaluate the proposed model and baseline models on the dataset collected from Baidu search engine system. Due to privacy concerns, it should be noted that none of the data contains any user-related information. Specifically, the dataset consists of 15,000 queries and over 770,000 query-webpage pairs. 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 2 shows the statistics of the dataset.

5.2 Evaluation Methodology

To evaluate the performance of our proposed method, we use Normalized Discounted Cumulative Gain (NDCG) [40], which has been widely adopted to evaluate the relevance in the context of ad-hoc search engine. Before introducing NDCG, we first introduce the Discounted Cumulative 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:

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

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***

Table 2. 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

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 $NDCG@4$.

5.3 Competitor Systems and Baselines

For all experiments, the baseline model is a ***pointwise-based self-trained LTR model without RFF-based over-parameterization***. Specifically, the baseline model adopts a self-trained LightGBM [21]-based LTR model with an L2 loss function, which has been deployed at 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:

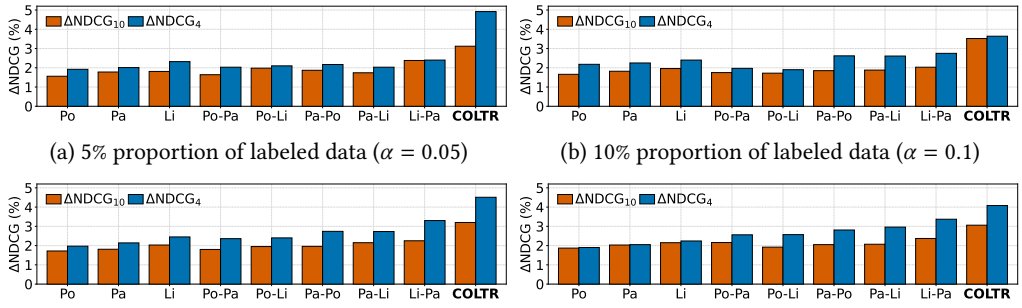
- **Base** is a *pointwise-based* self-trained LightGBM-based LTR model without *RFF-based over-parameterization*.
- **Po** refers to a *RFF-overparameterized* self-trained LTR model with the pointwise loss function.
- **Pa** refers to a *RFF-overparameterized* self-trained LTR model with the pairwise 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.

5.4 Experimental Settings

In this work, all the offline experiments are implemented on *PaddlePaddle*[2] cloud platform with 64G Memory, 4 NVIDIA Tesla V100 GPU, and 12T Disk. The online experiments are deployed in Baidu search engine system. More online A/B Test setting details are introduced in Section 5.6.1. We choose LightGBM, which is the most popular tree-based ranker, as the base ranking model with the number of trees as 2000 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 dimensions m as N/m.

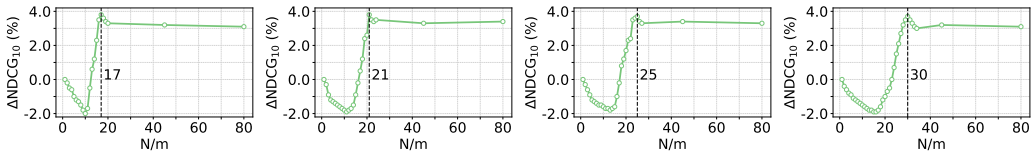
[2]https://www.paddlepaddle.org.cn/



(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. 4. 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.



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

Fig. 5. 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.

5.5 Offline Experimental Results

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

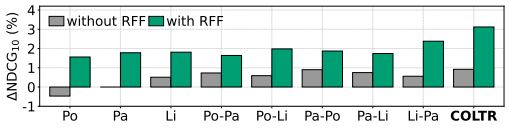
RQ1 How does **COLTR** perform compared with the baseline for LTR tasks?

RQ2 Which number of the transformed dimension can make LTR models gain the best performance under different ratios?

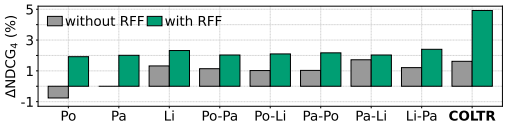
RQ3 Is the *RFF based over-parameterization* in **COLTR** necessary for improving performance?

RQ4 How does the number of rounds impact the performance of **COLTR**?

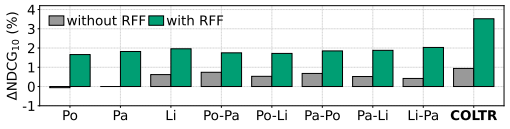
5.5.1 ***Comparative Results: RQ1.*** In Figure 4, we report the offline performance of **COLTR** compared with other baselines under four different ratios of labeled data on $\Delta NDCG_{10}$ and $\Delta NDCG_4$. For each semi-supervised learning model, we choose the model with the best performance 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



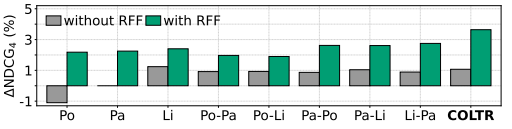
(a) 5% proportion of labeled data ($\alpha = 0.05$) on $\Delta NDCG_{10}$



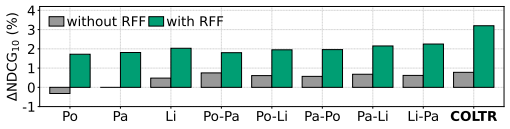
(b) 5% proportion of labeled data ($\alpha = 0.05$) on $\Delta NDCG_4$



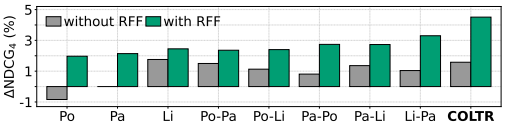
(c) 10% proportion of labeled data ($\alpha = 0.1$) on $\Delta NDCG_{10}$



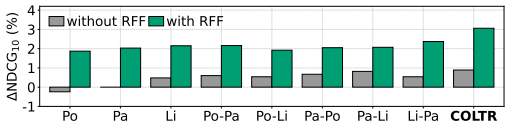
(d) 10% proportion of labeled data ($\alpha = 0.1$) on $\Delta NDCG_4$



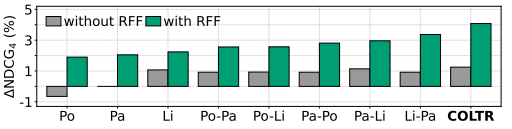
(e) 15% proportion of labeled data ($\alpha = 0.15$) on $\Delta NDCG_{10}$



(f) 15% proportion of labeled data ($\alpha = 0.15$) on $\Delta NDCG_4$



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

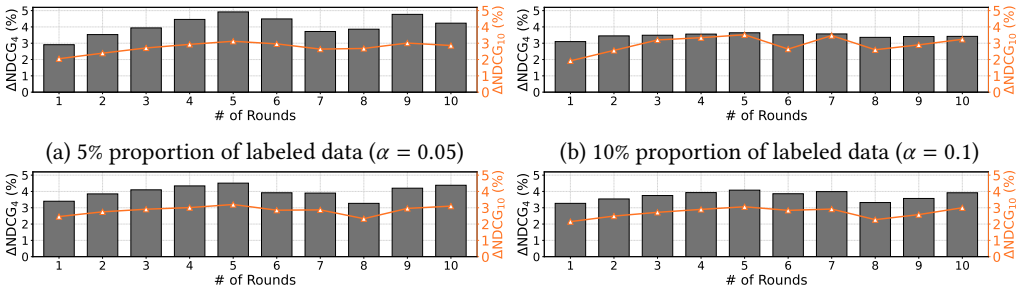


(h) 20% proportion of labeled data ($\alpha = 0.2$) on $\Delta NDCG_4$

Fig. 6. Ablation studies of *RFF-based Over-parameterization* for **COLTR** and baselines under different ratios of labeled data.

for pointwise model. Then the weaker model, pointwise model, generates relatively inaccurate but diverse pseudo-labels to 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-training 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 $\Delta NDCG_{10}$.

5.5.2 ***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 5 represents the margin curves for the pointwise model under four different ratios of labeled data on validation set. In Figure 5 (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 "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 5 (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



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

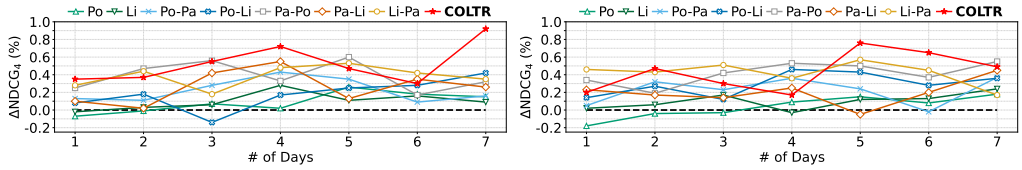
could improve the performance of downstream LTR models in the newly-fashioned "interpolating regime".

5.5.3 ***Ablation Results: RQ3.*** In order to demonstrate the effectiveness of *RFF-based Over-parameterization* part of **COLTR**, we conduct the ablation study 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 6 (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%** on 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.

5.5.4 ***Parameter Sensitivity: RQ4.*** In this section, we conduct a series of experiments to study the performance variation of **COLTR** with respect to the number of rounds for co-training. In Figure 7, 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 7 (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 similar phenomenons like Figure 7 (a). Therefore, we choose the trained **COLTR** at the 5th round for online experiments.

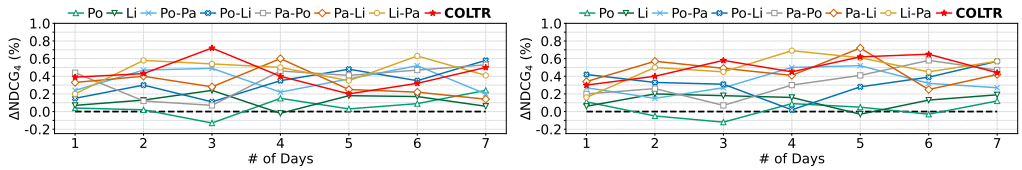
5.6 Online Experimental Results

To investigate the impact of **COLTR** at Baidu Search, we deploy the new system and conduct a series of online A/B Test with real-world web traffics compared with the base model in Baidu Search.



(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. 8. 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.

5.6.1 A/B Test Setups. In the online A/B Test, we conduct the experiment that compares the new ranking system, which deploys **COLTR**, with the old system for 7 days. For each day, we first remove pornographic and legally prohibited webpages. Then, we hire six common annotators to annotate the relevant score for each chosen query-webpage pair. Next, our professional annotators evaluate the quality of the annotations and guarantee that the accuracy is higher than 85%. Eventually, we leverage the weighted average of the annotations as the relevant score to train our proposed model. According to the offline experimental results, we choose the trained **COLTR** and other baseline models under four various ratios of labeled data in the best performance round. We conduct the online experiments with 0.6% real-world web traffics of Baidu search and concentrate on metrics that have a direct impact on user experience. From the perspective of business, we consider the NDCG of the top 4 ranking results and calculate $\Delta NDCG_4$ between the chosen model and the online base model.

5.6.2 Online Performance. Figure 8 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 in 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 of real-world online search engine. Basically, the online performance is consistent with our offline experiment result.

6 CONCLUSION AND DISCUSSION

In this paper, 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. **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** deployed at Baidu Search significantly boosts the online ranking performance in real-world applications, which is consistent with offline results.

In this work, we actually design LTR algorithms and **COLTR** on top of a pre-trained language models-based retrieval method, which supplies the candidates for webpage ranking and the features of LTR. The retrieval models are end-to-end trained incorporating both self-supervision and supervision signals, such as raw text-to-text matching, click-throughs, and dwell time that characterizes the relevance between queries and webpages. We however don't include LTR models in the end-to-end training with the language models, primarily due to two reasons: (1) the requests to re-training LTR models would be more frequent to adapt to the fast shift the internet interests, while language models would be updated infrequently in a low-cost fashion; (2) given the outputs of language models (i.e., features for LTR) for either queries or webpages, their RFF transformations are restored as immediate results for online ranking. In this way, given a query and a webpage for online ranking, **COLTR** pickups their RFF-transformed representations and passes them to the GBM for inference in a fast manner. In the future, we attempt to study the low-cost end-to-end neural LTR approaches and their practical deployment on real-world web-scale search systems.

REFERENCES

[1] Nurendra Choudhary, Nikhil Rao, Karthik Subbian, and Chandan K Reddy. Graph-based multilingual language model: Leveraging product relations for search relevance. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2789–2799, 2022.

[2] Xiaokai Chu, Jiashu Zhao, Lixin Zou, and Dawei Yin. H-ernie: A multi-granularity pre-trained language model for web search. In *Proceedings of the 45th International ACM SIGIR conference on research and development in information retrieval*, pages 1478–1489, 2022.

[3] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. Ernie 2.0: A continual pre-training framework for language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8968–8975, 2020.

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

[5] Ahmed El-Kishky, Thomas Markovich, Serim Park, Chetan Verma, Baekjin Kim, Ramy Eskander, Yury Malkov, Frank Portman, Sofía Samaniego, Ying Xiao, and Aria Haghighi. Twhin: Embedding the twitter heterogeneous information network for personalized recommendation. In Aidong Zhang and Huzefa Rangwala, editors, *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pages 2842–2850. ACM, 2022.

[6] Bochen Pang, Chaozhuo Li, Yuming Liu, Jianxun Lian, Jianan Zhao, Hao Sun, Weiwei Deng, Xing Xie, and Qi Zhang. Improving relevance modeling via heterogeneous behavior graph learning in bing ads. In Aidong Zhang and Huzefa Rangwala, editors, *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pages 3713–3721. ACM, 2022.

[7] Tan Yu, Yi Yang, Yi Li, Xiaodong Chen, Mingming Sun, and Ping Li. Combo-attention network for baidu video advertising. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2474–2482, 2020.

[8] Hao Liu, Qian Gao, Xiaochao Liao, Guangxing Chen, Hao Xiong, Silin Ren, Guobao Yang, and Zhiwei Zha. Lion: A gpu-accelerated online serving system for web-scale recommendation at baidu. In Aidong Zhang and Huzefa Rangwala, editors, *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pages 3388–3397. ACM, 2022.

[9] Jian Ouyang, Mijung Noh, Yong Wang, Wei Qi, Yin Ma, Canghai Gu, SoonGon Kim, Ki-il Hong, Wang-Keun Bae, Zhibiao Zhao, et al. Baidu kunlun an ai processor for diversified workloads. In *2020 IEEE Hot Chips 32 Symposium*, pages 1–18, 2020.

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

[11] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. 2005.

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

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

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

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

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

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

[18] Tao Qin and Tie-Yan Liu. Introducing letor 4.0 datasets. *arXiv preprint arXiv:1306.2597*, 2013.

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

[20] Zhaohui Zheng, Keke Chen, Gordon Sun, and Hongyuan Zha. 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*, pages 287–294, 2007.

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

[22] Claudio Lucchese, Cristina Ioana Muntean, Franco Maria Nardini, Raffaele Perego, and Salvatore Trani. 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*, pages 1281–1284, 2017.

[23] Zhen Qin, Le Yan, Honglei Zhuang, Yi Tay, Rama Kumar Pasumarthi, Xuanhui Wang, Michael Bendersky, and Marc Najork. Are neural rankers still outperformed by gradient boosted decision trees? In *International Conference on Learning Representations*, 2020.

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

[25] Zhaohui Zheng, Keke Chen, Gordon Sun, and Hongyuan Zha. 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*, pages 287–294, 2007.

[26] Yiding Liu, Weixue Lu, Suqi Cheng, Daiting Shi, Shuaiqiang Wang, Zhicong Cheng, and Dawei Yin. Pre-trained language model for web-scale retrieval in baidu search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3365–3375, 2021.

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

[28] Luca Didaci and Fabio Roli. 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)*, pages 522–530. Springer, 2006.

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

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

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

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

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

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

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

[36] Ping Li, Qiang Wu, and Christopher Burges. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Advances in Neural Information Processing Systems*, pages 65–72, 2008.

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

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

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

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

[41] Sebastian Bruch, Masrour Zoghi, Michael Bendersky, and Marc Najork. Revisiting approximate metric optimization in the age of deep neural networks. In Benjamin Piwowarski, Max Chevalier, Éric Gaussier, Yoelle Maarek, Jian-Yun Nie, and Falk Scholer, editors, *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 1241–1244. ACM, 2019.

[42] Gang Li Dixian Zhu Zhishuai Guo Quanqi Hu Bokun Wang Qi Qi Yongjian Zhong Tianbao Yang Zhuoning Yuan, Zi-Hao Qiu. Libauc: A deep learning library for x-risk optimization, 2022.

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

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

[45] Mehak Sheikh, Muhammad Adeel Asghar, Ruqia Bibi, Muhammad Noman Malik, Mohammad Shorfuzzaman, Raja Majid Mehmood, and Sun-Hee Kim. Dft-net: Deep feature transformation based network for object categorization and part segmentation in 3-dimensional point clouds. *Sensors*, 22(7):2512, 2022.

[46] Ali Rahimi, Benjamin Recht, et al. Random features for large-scale kernel machines. In *NIPS*, volume 3, page 5. Citeseer, 2007.

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

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

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

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

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

[52] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186, 2019.

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

[54] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

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

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

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

[59] An Qin, Mengbai Xiao, Jin Ma, Dai Tan, Rubao Lee, and Xiaodong Zhang. Directload: A fast web-scale index system across large regional centers. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1790–1801. IEEE, 2019.