Scope-aware Re-ranking with Gated Attention in Feed

Hao Qian[1,†], Qintong Wu[1,†], Kai Zhang[2], Zhiqiang Zhang[1], Lihong Gu[1], Xiaodong Zeng[1], Jun Zhou[1,⋆], Jinjie Gu[1]
[1]Ant Group, Hangzhou, China
[2]School of Data Science, University of Science and Technology of China, Anhui, China
{qianhao.qh,qintong.wqt,lingyao.zzq,lihong.glh,xiaodong.zxd,jun.zhoujun,jinjie.gujj}@antgroup.com
kkzhang0808@mail.ustc.edu.cn

ABSTRACT

Modern recommendation systems introduce the re-ranking stage to optimize the entire list directly. This paper focuses on the design of re-ranking framework in feed to optimally model the mutual influence between items and further promote user engagement. On mobile devices, users browse the feed almost in a top-down manner and rarely compare items back and forth. Besides, users often compare item with its adjacency based on their partial observations. Given the distinct user behavior patterns, the modeling of mutual influence between items should be carefully designed. Existing re-ranking models encode the mutual influence between items with sequential encoding methods. However, previous works may be dissatisfactory due to the ignorance of connections between items on different scopes. In this paper, we first discuss *Unidirectivity* and *Locality* on the impacts and consequences, then report corresponding solutions in industrial applications. We propose a novel framework based on the empirical evidence from user analysis.

To address the above problems, we design a **S**cope-aware **R**e-ranking with **G**ated **A**ttention model (**SRGA**) to emulate the user behavior patterns from two aspects: 1) we emphasize the influence along the user's common browsing direction; 2) we strength the impacts of pivotal adjacent items within the user visual window. Specifically, we design a global scope attention to encode inter-item patterns unidirectionally from top to bottom. Besides, we devise a local scope attention sliding over the recommendation list to underline interactions among neighboring items. Furthermore, we design a learned gate mechanism to aggregating the information dynamically from local and global scope attention. Extensive offline experiments and online A/B testing demonstrate the benefits of our novel framework. The proposed SRGA model achieves the best performance in offline metrics compared with the state-of-the-art re-ranking methods. Further, empirical results on live traffic validate that our recommender system, equipped with SRGA in the re-ranking stage, improves significantly in user engagement.

† The first two authors contributed equally to the work.
⋆ Corresponding Author.

WSDM '22, February 21–25, 2022, Tempe, AZ, USA.

ACM ISBN 978-1-4503-9132-0/22/02…$15.00
https://doi.org/10.1145/3488560.3498403

CCS CONCEPTS

• **Information systems** → **Social recommendation**; **Learning to rank**; • **Computing methodologies** → **Learning to rank**.

KEYWORDS

Recommender System; Learing To Rank; Re-ranking

ACM Reference Format: Hao Qian, Qintong Wu, Kai Zhang, Zhiqiang Zhang, Lihong Gu, Xiaodong Zeng, Jun Zhou, Jinjie Gu. 2022. Scope-aware Re-ranking with Gated Attention in Feed. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22), Feb. 21–25, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3488560.3498403

1 INTRODUCTION

The feed has become a popular product in E-commerce apps for its efficiency to seamlessly recommend a variety of items, such as clothes and shoes, to meet the distinct interests of different users [32]. As shown in Figure 1a, the feed usually displays items in a compact grid form [14]. Examples include Amazon's and Taobao's apps. Though with exceptional business successes, the compact display fashion fundamentally changes user behavior patterns, where intrinsic inter-item correlations among the slate of items notably influence user's choices. For example, users could prefer an item with a lower price compared with surrounding items. To further optimize the recommendation quality, the re-ranking method considering the mutual influence on a slate of items has been widely studied and applied in industrial recommender system.

There are two lines of researches that have been applied in the industrial re-ranking problems. Traditional Learning-to-Rank (LTR) techniques, especially *listwise* methods [8–10, 24, 31, 37], propose to learn a parametrized scoring function by optimizing the well-designed loss function, which implicitly models the relative order between recommended items. In addition, deep models [1, 21, 22, 38] explicitly extract the feature representation of the initial ranking list, and then re-rank items based on such global information. However, these approaches still could be dissatisfactory due to the following two phenomena of user behavior patterns:

(P1) Unidirectivity. We observe that users browse the feed almost in a unidirectional top-down manner instead of back and forth [14]. Also, previous work [1, 33] shows that information encoded from the bottom to top is of little importance. It should be noted that users are hardly aware of items located at lower positions while glancing over items at the top [35]. On the contrary, the top recommendation brings the anchoring effect that heavily influences user's subsequent choices at the bottom [11]. For instance, items

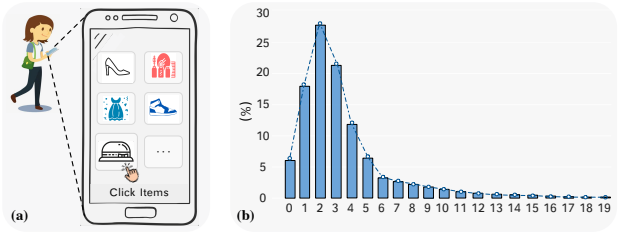


Figure 1: (a) shows that users check a few items before clicking in feed. In (b), X-axis is the margin between the index of the clicked and the maximum index of exposed items, Y-axis is the empirical probability (%) mass of the margin. The right-skewed distribution implies that most clicks happen within a relatively small range as users browsing the feed.

at the bottom are less relevant in general, and thus modeling the influence in the reverse direction may be harmful to overall performance. Previous approaches result in dissatisfactory performance since they ignore the unidirectional user browsing patterns but equally quantify the mutual influence from both directions.

(P2) Locality. When users stop at a specific item, they tend to compare it with adjacent ones based on their partial observation. We observe that most user behaviors, such as clicks or purchases, mainly occur in a narrow visual window as shown in Figure 1. In other words, the positive feedback reflects user's genuine preference from their local comparisons (i.e., prices, styles). Consequently, the local structure of feedback should be clearly underlined. In contrast, most literature mainly leverages the information of the entire list implicitly or explicitly but overlook the local structure.

To address aforementioned issues, we introduce a novel framework to underline scope-aware influence in a principled way and propose a gated attention based re-ranking model, named **S**cope-aware **R**e-ranking with **G**ated **A**ttention model (**SRGA**). Specifically, to characterize user's unidirectional browsing patterns (**P1**), a global scope attention is designed to encode the inter-item influence along the top-down direction featuring the significant impact of top items. A local scope attention, adjusted by a hyper-parameter window size, is adopted to explore the contrastive features (e.g., prices and styles) derived from the local structure among neighboring items (**P2**). As the global and local scope attention encode the representation on different scopes, the fusion is of great importance. To adaptively quantify contributions between the two, we further design a learned gate mechanism for each item based on the representation of user preferences. Our primary contributions can be summarized as follows:

- We discuss the impacts and consequences of user behavior patterns in the feed. To the best of our knowledge, we are the first to formulate the problems of *unidirectivity* and *locality* in the context of re-ranking. We also provide insights on modeling the item context based on the empirical evidence.
- We propose an innovative framework, called SRGA, for mitigating the problems by encoding the mutual influence from distinct scopes. The framework is designed to be modular and applicable at large-scale recommender system.
- We perform extensive experiments on public and real-world datasets to verify the superiority of our proposed framework. Also, SRGA gains the performance improvement of 4.22% in Click Through Rate (CTR) when deployed in recommendation feed, which is significant in industrial application. With detailed analysis, we demonstrates the effectiveness of global and local scope attention.

2 RELATED WORK

Our work is related to the following area: 1) LTR in recommender system, which aims to learn an optimized ranking order of items in a list, and 2) re-ranking that improves the ranking order through explicitly modeling the interactions among a slate of items [2, 19].

2.1 LTR

The early LTR methods, mainly categorized as pointwise, pairwise, and listwise, focus on designing efficient loss function to learn an optimized recommended item list. The pointwise methods [12, 30] learn user preference towards items independently with a binary cross entropy loss. The pairwise methods [7, 8] predict the partial order of two items in the list through a classification loss. Moreover, the listwise methods [10, 20] incorporate the whole list of items and try to find an optimal list through the listwise ranking loss. Researchers are working on designing efficient loss function to reduce computation complexity over the permutations of the list [26, 34]. However, LTR methods are limited to the exploitation of labels but neglect the importance of mutual influence between items in feature space. Additionally, it is less effective in learning the scoring function with implicit feedback due to the ignorance of complicated user behavior patterns, though succeeding with explicit feedback. Our approach explicitly encodes the mutual influences from distinct scopes and handles implicit feedback in the context of re-ranking.

2.2 Re-ranking

Existing re-ranking methods focus on explicitly extracting relationship among a slate of items. DLCM [1] and PRM [22] are closely related to our approach. [1] employs Gate Recurrent Unit (GRU) [13] to sequentially encode the prior knowledge of item context to optimize the arrangement order of items. [22] introduces pre-trained embedding to learn personalized representation of users and employs Transformer [28] structure to learn the global interactions of any item-pair. Our approach differs from the two in that we decompose the encoding process into two distinct scopes. Similar to the methodology of [1], we characterize global mutual influence from top to bottom, which emulates the impact of items along the ranking order, and we replace GRU with Transformer in favor of speed. Different from [22], we only strengthen the mutual interaction of item-pair within a local window instead of spanning the entire list. With carefully designed gated attention layer, our approach can selectively emphasize on mutual influences from distinct scopes.

Certain methods need relatively large computation complexities and serving costs compared to our approach. For instance, Global Re-Rank [38] applies Recurrent Neural Network (RNN) [16] to capture the influence from the list and generate the best ranking order using the beam search algorithm. Seq2Slate [4] adopts the Seq2Seq framework [5] to capture rich dependencies between ranked items and develops a sequential decoding method during inference.

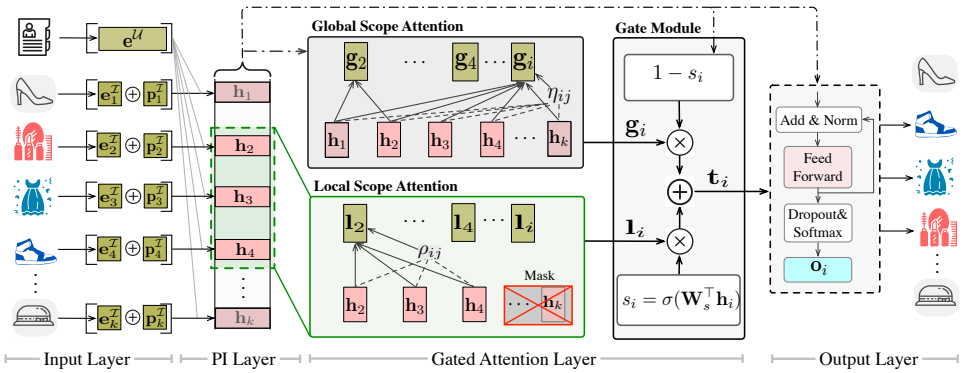


Figure 2: The overview architecture of the SRGA Model. The green box indicates the local attention with window size of 3.

3 PROBLEM DEFINITION

We first formally define the problem of re-ranking in recommender system as follows:

DEFINITION 1. **Re-ranking**. Let $\mathbf{x}^{\mathcal{U}} \in \mathbb{R}^{n_{\mathcal{U}}}$ denotes user profile and $\mathbf{x}^{\mathcal{I}} = [\mathbf{x}_1^{\mathcal{I}} \parallel ... \parallel \mathbf{x}_k^{\mathcal{I}}] \in \mathbb{R}^{kn_{\mathcal{I}}}$ represents the concatenation of a list of item features, where item i's feature $\mathbf{x}_i^{\mathcal{I}}$ is represented with one-hot encoding, k is the length of the list, $\parallel$ denotes the vertical concatenate operation, and $n_{\mathcal{U}}$ and $n_{\mathcal{I}}$ are the dimension of user and item features, respectively. The initial order of lists is generated by the preceding ranking algorithm. A training sample consists of input features $\mathbf{x} = [\mathbf{x}^{\mathcal{U}} \parallel \mathbf{x}^{\mathcal{I}}]$ and the user feedback (i.e., the ground-truth label), such as clicks or purchases, of items on the list denoted as $\mathbf{y} = [y_1, ..., y_k]$, where $y_i \in \{0, 1\}$ is the user feedback of item i. The problem of re-ranking is to learn a scoring function, $\mathbf{f}(\mathbf{x}; \Theta)$, that generates an ordered list by scoring each item in the list, parameterized by Θ. The scoring function is usually optimized by minimizing the loss function as:

$$\mathcal{L} = \sum_{\mathcal{R}} \ell\{\mathbf{y}, \mathbf{f}(\mathbf{x}; \Theta)\}, \tag{1}$$

where $\mathcal{R}$ is the set of training samples, and ℓ is the loss computed from each sample.

4 ALGORITHM DESIGN

In this section, we describe the architecture of the SRGA model as depicted in Figure 2, which includes the input layer, the personalized interaction layer, the gated attention layer, and the output layer. To increase the representation capability of SRGA, we may stack a few gated attention layers together.

4.1 Input Layer

The input layer defines the input features and the embedding procedure. As described in Section 3, input features are the composite of user profile $\mathbf{x}^{\mathcal{U}}$ and item features $\mathbf{x}^{\mathcal{I}}$. Besides, the position of items in the list is one of the crucial features in the re-ranking and is encoded with a trainable position embedding. Specifically, we use $\mathbf{p}_i^{\mathcal{I}} \in \mathbb{R}^{d_{\mathcal{I}}}$ to denote the position embedding of item i and $\mathbf{P}^{\mathcal{I}} \in \mathbb{R}^{d_{\mathcal{I}} \times k}$ to denote position embedding of the whole item list. For efficient storage and computation, the sparse one-hot encoded input features are projected into low-dimension dense representations through the embedding procedure as follows:

$$\mathbf{e}^{\mathcal{U}} = \mathbf{W}^{\mathcal{U}}\mathbf{x}^{\mathcal{U}}, \quad \mathbf{e}_i^{\mathcal{I}} = \mathbf{W}^{\mathcal{I}}\mathbf{x}_i^{\mathcal{I}} + \mathbf{p}_i^{\mathcal{I}}, \tag{2}$$

where $\mathbf{e}^{\mathcal{U}} \in \mathbb{R}^{d_{\mathcal{U}}}$ and $\mathbf{e}_i^{\mathcal{I}} \in \mathbb{R}^{d_{\mathcal{I}}}$ are the projected representations of user profile and item i's features, respectively. $\mathbf{W}^{\mathcal{U}}$ and $\mathbf{W}^{\mathcal{I}}$ are the trainable matrices. $d_{\mathcal{U}}$ and $d_{\mathcal{I}}$ is the embedding size of projection of user profile and item features.

4.2 Personalized Interaction Layer

As users may carry different interests towards items, we design a Personalized Interaction (PI) Layer to capture distinct user preference toward each item. Specifically, user profile representation $\mathbf{e}^{\mathcal{U}}$ is concatenated with item i's features $\mathbf{e}_i^{\mathcal{I}}$ before applying two fully connected layers along with ReLU activation function as follows:

$$\mathbf{h}_i = \text{ReLU}(\mathbf{W}_2\text{ReLU}(\mathbf{W}_1[\mathbf{e}^{\mathcal{U}} \parallel \mathbf{e}_i^{\mathcal{I}}] + \mathbf{b}_1) + \mathbf{b}_2), \tag{3}$$

where $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{b}_1$, and $\mathbf{b}_2$ are trainable matrices and bias. $\mathbf{h}_i \in \mathbb{R}^{d_2}$ is the personalized user preference representation toward item i. Without loss of generality, we use $\mathbf{H} \in \mathbb{R}^{d_2 \times k}$ to denote the personalized user preference representation toward the whole item list.

4.3 Gated Attention Layer

To explicitly encode the mutual influence among a slate of items, we design a scope-aware Gated Attention Layer. It includes a Global Scope Attention (GSA) that learns the relative ranking order among a slate of items unidirectionally in the global scope and a newly designed Local Scope Attention (LSA) that addresses the impact from neighboring items in the local scope, and gate module that adaptively leverages the contribution from both GSA and LSA.

Before applying high-level feature interactions in the gated attention layer, the personalized user preference representation $\mathbf{H}$ is first linearly projected to query $\mathbf{Q} \in \mathbb{R}^{d_g \times k}$, key $\mathbf{K} \in \mathbb{R}^{d_g \times k}$, and value $\mathbf{V} \in \mathbb{R}^{d_g \times k}$ as follows:

$$\mathbf{Q}, \mathbf{K}, \mathbf{V} = \mathbf{W}_{\mathbf{Q}}\mathbf{H}, \mathbf{W}_{\mathbf{K}}\mathbf{H}, \mathbf{W}_{\mathbf{V}}\mathbf{H}, \tag{4}$$

where $\mathbf{W}_{\mathbf{Q}}$, $\mathbf{W}_{\mathbf{K}}$, and $\mathbf{W}_{\mathbf{V}}$ are trainable weight matrices of query, key, and value, respectively.

The relationship between item i and j is measured with the scaled-dot self-attention weight ξ_{ij} in [28] as follows:

$$\xi_{ij} = \frac{\mathbf{q}_i \mathbf{k}_j}{\sqrt{d_g}}, \tag{5}$$

where $\mathbf{q}_i$ and $\mathbf{k}_i$ are the i-th column of $\mathbf{Q}$ and $\mathbf{K}$, representing the query and key of item i, respectively. The global and local influence is based on the attention weight ξ_{ij}.

Global Scope Attention (GSA). This module is responsible for explicitly modeling the interaction among items given an ordered list. As the discussion on Unidirectivity (**P1**), encoding mutual influence from bottom to top contributes little to overall performance [1], and such noisy information could be even harmful if the model is ill-fitted. Thus, we utilize the masked attention mechanism [6, 25] that strengthens the impact of items along the user's common browsing direction, which mitigates the mismatch between bidirectional modeling and nearly unidirectional influences. Then we apply softmax function to scale the masked attention weight before multiplying with the corresponding value $\mathbf{v}_j$. The operation incorporates the global influence among a slate of items and characterizes the importance of each item as follows:

$$\eta_{ij} = \begin{cases} \xi_{ij}, & i < j \\ -\infty, & \text{otherwise} \end{cases}, \tag{6}$$

$$\mathbf{g}_i = \sum_j \text{softmax}(\eta_{ij})\mathbf{v}_j, \tag{7}$$

where $\mathbf{g}_i \in \mathbb{R}^{d_g}$ is the global attention representation of item i in the global scope. $\mathbf{v}_i$ is the i-th column of $\mathbf{V}$, representing the value of the item i.

Local Scope Attention (LSA). This module reinforces the local interactions between adjacency items. We motivate the necessity, which is justified through empirical evidence from user behavior analysis, of promoting the influence between item-item comparison in the Locality (**P2**) discussion. Different from GSA, the mutual influence is addressed from both directions in LSA. By restricting the item interactions with a local window, LSA emulates the strong influence in the local scope. Meanwhile, as user sight shifts, e.g., scrolling down items in the feed or skimming through the website from top to bottom, the intensity of local interactions varies among different items on the fly. By sliding the local window along with the item list, LSA captures the evolving patterns between adjacency items. Specifically, this process is formulated as:

$$\rho_{ij} = \begin{cases} \xi_{ij}, & |i-j| < \frac{m}{2} \\ -\infty, & \text{otherwise} \end{cases}, \tag{8}$$

$$\mathbf{l}_i = \sum_j \text{softmax}\left(\rho_{ij}\right)\mathbf{v}_j, \tag{9}$$

where window size m is a hyperparameter and ρ_{ij} is the local scope attention weight of item i influenced by item j. For a local window, there are at most m items involved in the interaction process, which emulates the scenario that users compare m items in the local scope. The local scope attention weight ρ_{ij} is then normalized by a softmax function before multiplying with the corresponding value $\mathbf{v}_j$. $\mathbf{l}_i \in \mathbb{R}^{d_g}$ denotes the local attention representation of item i in the local scope.

Gate Module. This module highlights the local interaction and preserves the unidirectional global influence at the same time. Users' preferences determine the attention allocated to different items decisively, and thus the contribution should be quantified given users' characterizations. Besides, user representations are strong signals and generalize well in large-scale applications. To conclude, we design a gate module to dynamically adjust the flow of information from GSA and LSA, in which a gating weight s_i is learned from personalized user preference representation of i-th item $\mathbf{h}_i$ as:

$$\text{s}_i = \sigma(\mathbf{w}_s^\top \mathbf{h}_i), \tag{10}$$

where $\mathbf{w}_s$ is a trainable vector of the gate module, and σ is the sigmoid activation function.

The gating weight s_i is then used to dynamically aggregate the contribution from GSA and LSA as follows:

$$\mathbf{t}_i = (1 - \text{s}_i) * \mathbf{g}_i + \text{s}_i * \mathbf{l}_i, \tag{11}$$

where $\mathbf{t}_i \in \mathbb{R}^{d_g}$ is the output of the gate module, which carries integrated information from both GSA and LSA.

Subsequently, $\mathbf{t}_i$ is fed to residual connection [15], layer normalization [3], and position-wise feed-forward layer as in [28], which generates a final hidden representation of item i as $\mathbf{o}_i \in \mathbb{R}^{d_g}$.

4.4 Output Layer

The representation $\mathbf{o}_i$ is feed into a single layer perceptron network to produce a user preference probability over item i as follows:

$$p_i = \sigma(\mathbf{w}^\top \mathbf{o}_i + \mathbf{b}), \tag{12}$$

where $\mathbf{w}$ and $\mathbf{b}$ are trainable weight and bias parameters, σ is the sigmoid activation function.

To optimize the parameters, the cross entropy loss is defined as:

$$\mathcal{L} = -\sum_{\mathcal{R}} \{\frac{1}{k}\sum_i^k y_i \log p_i + (1 - y_i)\log(1 - p_i)\}. \tag{13}$$

where $\mathcal{R}$ is the set of training samples.

5 EXPERIMENTS

In this section, we conduct extensive offline experiments on SRGA against benchmark methods to demonstrate its effectiveness and robustness. Besides, we perform the ablation study to analyze the effects of sub-modules and hyper-parameters in SRGA.

5.1 Datasets

Currently, the Yahoo! Webscope v2.0 set 1 [*] and Microsoft 10K [23] are the commonly used public datasets in LTR area. In addition, PRM [22] released an E-commerce re-ranking dataset [†]. Besides, we collect re-ranking dataset from a real-world recommender system. The statistics of each dataset are listed in Table 1.

Table 1: Statistics of datasets used in this paper.

Datasets	Data Attributes		
	# Users / Queries	# Items / Docs	# Samples
Yahoo	29,921	709,877	29,921
Microsoft	10,000	1,200,189	10,000
PRM Public	743,720	7,246,323	14,350,968
Real-world	4,988,030	416,473	6,908,592

We categorize datasets into two distinct groups based on their feedback type. Yahoo and Microsoft datasets contain relevance judgments from experts as explicit feedback. While PRM public, as well as the real-world dataset, collect implicit feedback (i.e., clicks) from recommendation logs.

[*] https://webscope.sandbox.yahoo.com
[†] https://github.com/rank2rec/rerank

Yahoo dataset includes search queries and documents collected from Yahoo search engine. The relevance of a document to a corresponding query is given by one of 5 relevance labels from 0 (bad) to 4 (perfect). It is converted to binary label with a threshold of 2.

Microsoft dataset Similarly, this dataset, collected from bing search engine, also presents query-doc pairs that contain query id, relevance judgements, and text features.

PRM public dataset contains rich records of user profile, click-through labels and raw features for ranking from Taobao's recommendation system. Items are aligned from top to bottom in feed.

Real-World dataset Similar to PRM public dataset, we collect impression and click logs in feed from a real-world recommendation system. A data sample consists of user profile, item information, and user feedback. The maximum length of the list is truncated to 20 for online throughput and latency concerns.

5.2 Benchmark Methods

To show the performance of our proposed model, we compare SRGA with the state-of-the-art (SOTA) re-ranking models on top of the LTR methods. The benchmarks are:

- **SVMrank** [18] is trained with pairwise loss to model the score function. We use the implementation of SVMrank from [18] to train the LTR model.
- **LambdaMART** [9] is the SOTA LTR algorithm trained with listwise loss. The implementation of LambdaMART is from the Ranklib package [‡].
- **DNN-based LTR** such as Deep & Cross [30] structure, is trained with pointwise loss and used as the LTR model for PRM public and real-world dataset.
- **DLCM** [1] encodes mutual influence of items with unidirectional GRU [13] and proposes an effective attention-based AttRank loss that outperforms most listwise methods.
- **PRM** [22] applies transformer to capture item correlations in a bidirectional way. Furthermore, it incorporates a pre-trained personalized vector to strengthen personalized representation power of users.

Besides, methods like Global Re-Rank [38] and Seq2Slate [4] require a relatively large computation complexity and serving cost on the recommender system due to a sequential decoding procedure. To reduce the burden and risk, we do not include them as benchmark methods and deploy them for inference.

5.3 Implementation Details

Hyperparamters. For fair comparison, we set embedding size $d_{\mathcal{U}}$ and $d_{\mathcal{I}}$ = 16, learning rate = 0.001, batch size = 256, l2 regularizer = 0.001, dropout rate = 0.2, and select Adam as optimizer for all models. For DNN-based LTR, we used [64, 32] as hidden units in the deep component. In SRGA, we set hidden units of PI layer as [64, 32], one gated attention layer, window size m of 4 in LSA, and initialize parameters with truncated normal with a standard deviation of 0.02. For other methods, we follow the hyperparameter configurations described in their paper. We train all the models on parameter server based distributed learning systems.

Evaluation Metrics. As re-ranking method refines the order of a slate of items, we employ Mean Average Precision (MAP) [36] and Normalized Discounted Cumulative Gain (NDCG) [27] as rank-aware metrics to evaluate model performance [17]. **MAP@k** measures the mean average precision of the top-k items in the ordered list, in which precision represents the fraction of positive feedbacks in the list. **NDCG@k** further extends the MAP@k metric since items with higher relevance should be placed on top positions of the recommended list compared to those less relevant. For all results, we use "$*$" to indicate that SRGA is significantly different from the runner-up method at the significance level of 0.01.

[‡]https://sourceforge.net/p/lemur/wiki/RankLib/

5.4 Experimental Results

In this section, we evaluate the performance of our model on public and real-world datasets along with detailed analysis of results in Table 2. The major results are summarized as follows:

- SRGA consistently outperforms all benchmark methods in all datasets by a significant margin. Specifically, SRGA achieves remarkable improvement over the strongest baseline w.r.t NDCG@10 by 0.39 ∼ 3.33% and MAP@10 by 0.65 ∼ 3.26%, respectively. It demonstrates the superiority and versatility of SRGA in re-ranking applications. It is beneficial to employ LSA to capture the local relationship among the neighboring items in the local scope and adopt GSA to explicitly encode the unidirectional user behavior patterns in the global scope.
- The LTR methods (i.e., SVMrank, LambdaMART and DNN-based LTR) perform relatively poor across all datasets, which indicates that it is insufficient to rely on loss function to capture item interactions. Thus, it is critical to explicitly encode item context in practical applications.
- Compared with PRM, the improvements of SRGA are statistically significant in PRM public and real-world datasets containing implicit feedback. Although two models are both transformer-based, SRGA is better for modeling the item context in industrial recommendation settings. It is worth noting that experts assign the relevance score to each document independently in Yahoo and Microsoft datasets, which weakens our modeling assumptions. Nonetheless, the improvements still show the effectiveness of our design choices.

5.5 Ablation Study

To understand the impact of the sub-modules of SRGA, an ablation study is carried out by removing one sub-module at a time. The results reported here are based on experiments conducted in real-world dataset. Similar results are also achieved in other datasets.

LSA vs. GSA. As described in Section 4.3, LSA and GSA characterize item interactions from different scopes. Hence, we analyze the impact of sub-module in each scope in experiments. As shown in Figure 3a and 3e, we can observe that the performance drops sharply as LSA removed. Specifically, without LSA, we report 2.10% decrease in NDCG@10 and 3.15% decrease in MAP@10 metrics. Clearly, the standalone GSA is less effective to capture the rich mutual influences between items, which confirms the necessity of underlining local interactions between items. Also, the result verifies the assumption of introduced Locality (**P2**).

Furthermore, we conclude that the overall performance drops slightly without GSA. In the absence of GSA, the gated attention layer is less representative, since the limited scope of LSA discards useful information from the top. Therefore, it is indispensable

Table 2: Results of benchmark methods on public and real-world dataset. We underline the best performed baseline for comparison. The last row in each dataset indicates the percentage of improvements gained by SRGA w.r.t the best performed baseline.

Dataset	Initial List	Model	NDCG@3	NDCG@5	NDCG@10	MAP@3	MAP@5	MAP@10
Yahoo	SVMRank	SVMRank	0.6513	0.6738	0.7263	0.7768	0.7371	0.6828
		DLCM	0.6834	0.6991	0.7459	0.8086	0.7676	0.7088
		PRM	0.6848	0.7070	0.7532	0.8189	0.7767	0.7205
		SRGA	**0.6894***	**0.7110***	**0.7588***	**0.8249***	**0.7818***	**0.7389***
			(+0.67%)	**(+0.57%)**	**(+0.74%)**	**(+0.73%)**	**(+0.66%)**	**(+2.55%)**
	LambdaMART	LambdaMART	0.6757	0.6963	0.7382	0.7832	0.7577	0.7049
		DLCM	0.6849	0.7067	0.7434	0.8078	0.7782	0.7190
		PRM	0.7089	0.7199	0.7516	0.8143	0.7865	0.7272
		SRGA	**0.7124***	**0.7228***	**0.7545***	**0.8173***	**0.7896***	**0.7319***
			(+0.49%)	**(+0.40%)**	**(+0.39%)**	**(+0.37%)**	**(+0.39%)**	**(+0.65%)**
Microsoft	SVMRank	SVMRank	0.3112	0.3459	0.3785	0.3813	0.3640	0.3531
		DLCM	0.3854	0.3929	0.4173	0.4995	0.4935	0.4600
		PRM	0.3958	0.4033	0.4204	0.5068	0.4980	0.4657
		SRGA	**0.3995***	**0.4075***	**0.4247***	**0.5172***	**0.5079***	**0.4743***
			(+0.93%)	**(+1.04%)**	**(+1.02%)**	**(+2.05%)**	**(+1.98%)**	**(+1.85%)**
	LambdaMART	LambdaMART	0.4133	0.4254	0.4466	0.5400	0.5274	0.4940
		DLCM	0.4248	0.4306	0.4554	0.5548	0.5371	0.5059
		PRM	0.4351	0.4420	0.4597	0.5640	0.5455	0.5148
		SRGA	**0.4384***	**0.4441***	**0.4664***	**0.5713***	**0.5516***	**0.5251***
			(+0.76%)	**(+0.48%)**	**(+1.46%)**	**(+1.29%)**	**(+1.12%)**	**(+2.00%)**
PRM public	DNN-based LTR	DLCM	0.2242	0.2673	0.3422	0.2709	0.2930	0.3025
		PRM	0.2290	0.2714	0.3424	0.2768	0.3114	0.3210
		SRGA	**0.2359***	**0.2790***	**0.3538***	**0.2847***	**0.3208***	**0.3304***
			(+3.01%)	**(+2.80%)**	**(+3.33%)**	**(+2.85%)**	**(+3.02%)**	**(+2.93%)**
Real-World	DNN-based LTR	DLCM	0.3725	0.4396	0.5151	0.3600	0.3922	0.4133
		PRM	0.3758	0.4399	0.5174	0.3647	0.3953	0.4146
		SRGA	**0.3879***	**0.4543***	**0.5285***	**0.3767***	**0.4084***	**0.4281***
			(+3.22%)	**(+3.27%)**	**(+2.15%)**	**(+3.29%)**	**(+3.31%)**	**(+3.26%)**

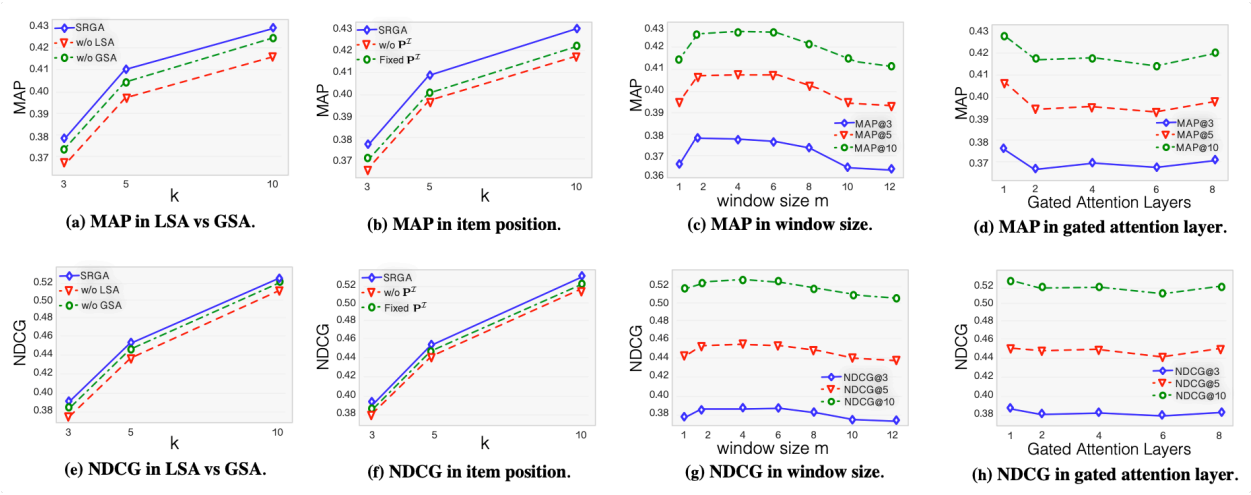
to encode the inter-item patterns for each item-pair with GSA unidirectionally in the global scope. It should be noted that standalone LSA still achieves better performance than standalone GSA, which further confirms the effectiveness of LSA. Finally, the results demonstrate the essential role of the gated attention layer: effective aggregation of both LSA and GSA.

Modeling Choices of Direction In feed, users' browsing patterns raise the unique research question: Does properly designed model benefit the final performance? We meticulously design the following configurations:

- SRGA_GSA$_{bi}$. The bidirectional GSA introduces the extra mutual influence along the bottom-top direction in the global scope, while other modules remain the same.
- SRGA_LSA$_{uni}$. It applies a unidirectional mask on LSA, which assumes that items located ahead of the list of local items play the dominant role in the local scope.
- SRGA$_{rev}$. This is the composite of GSA$_{bi}$ and LSA$_{uni}$.

As shown in Table 3, we conduct experiments with these model configurations and report the relative improvement of SRGA w.r.t them, where SRGA performs best consistently. Comparing SRGA_GSA$_{bi}$ with SRGA, we report little information gain from the bidirectional representation in the global scope. Different from previous work [1], our experiment further confirms that information encoded from the reversed direction could be detrimental to re-ranking problems in feed, which accords with our observation (**P1**). Moreover, SRGA_LSA$_{uni}$ performs poorly due to the encoded representation fails to capture the distinct mutual influence (e.g., prices and styles) in the local scope, which we address the necessity in (**P2**). The performance of SRGA$_{rev}$ falls far behind the standard SRGA, which confirms the proper modeling of our proposed method. In conclusion, the main factor that determines the modeling choices is the unidirectivity and locality, which effectively characterizes the user behavior patterns in different scopes in feed apps.

Impact of Item Position. As we aim at ranking most relevant items on the top of list, the position of items is crucial in re-ranking scenario. We study the impact of position embedding of items by removing $\mathbf{P}^I$, fix $\mathbf{P}^I$ with sine and cosine function as [28], and learn $\mathbf{P}^I$ with parameters as described in Section 4.1. From Figure 3b and 3f, we observe the performance becomes worse as $\mathbf{P}^I$ removed. Since the attention mechanism in gated attention layer ditches the recurrence mechanism in favor of speed, the model itself have little sense of position or relative order for each item. Consequently, it is



(a) MAP in LSA vs GSA. (b) MAP in item position. (c) MAP in window size. (d) MAP in gated attention layer.

(e) NDCG in LSA vs GSA. (f) NDCG in item position. (g) NDCG in window size. (h) NDCG in gated attention layer.

Figure 3: Results of experiments conducted in ablation and hyper-parameter study. Subfigures (a) and (e) show the MAP and NDCG metrics in LSA vs. GSA ablation study. Subfigures (b) and (f) are results from impact of item position. Subfigures (c) and (g) are results from effect of window size. Subfigures (d) and (h) are results from effect of gated attention layers.

Table 3: Results of modeling choices of direction.

	SRGA_GSA$_{bi}$	SRGA_LSA$_{uni}$	SRGA$_{rev}$	SRGA
NDCG@3	0.3863	0.3854	0.3840	**0.3879***
	(+0.41%)	(+0.65%)	(+1.02%)	-
NDCG@5	0.4520	0.4501	0.4492	**0.4543***
	(+0.50%)	(+0.93%)	(+1.13%)	-
NDCG@10	0.5266	0.5231	0.5214	**0.5285***
	(+0.36%)	(+1.03%)	(+1.37%)	-
MAP@3	0.3763	0.3754	0.3750	**0.3767***
	(+0.11%)	(+0.34%)	(+0.45%)	-
MAP@5	0.4078	0.4071	0.4066	**0.4084***
	(+0.14%)	(+0.31%)	(+0.45%)	-
MAP@10	0.4274	0.4264	0.4259	**0.4281***
	(+0.17%)	(+0.41%)	(+0.51%)	-

necessary to incorporate the position information of items in the recommended list. Moreover, the fixed position embedding $\mathbf{P}^I$ is still insufficient to characterize the sequential nature of positions in the vector space. This is possibly because that the sinusodial position embedding expresses symmetry patterns without distinguishing the direction of items [29]. Therefore, its performance is limited in re-ranking scenario where the modeling direction is sensitive. Accordingly, SRGA with learnable position embedding is more suitable and effective to represent the position information of items in re-ranking scenario.

Effect of Gate Module. In our model, we enrich the information of items by utilizing a learned gating weight in Gate Module to integrate information from LSA and GSA as Equation 11. In order to explore the effect of different aggregation mechanisms, we design multiple variants of Gate Module in SRGA as follows:

- **SRGA**$_{sa}$: It utilizes self-attention mechanism as follows:

$$\alpha_i = \frac{\exp(\mathbf{l}_i)}{\exp(\mathbf{l}_i) + \exp(\mathbf{g}_i)}, \qquad \mathbf{t}_i = (1-\alpha_i) * \mathbf{g}_i + \alpha_i * \mathbf{l}_i. \tag{14}$$

- **SRGA**$_{concat}$: It applies concatenation.

$$\mathbf{t}_i = [\mathbf{g}_i \parallel \mathbf{l}_i], \tag{15}$$

- **SRGA**$_{sum}$: It applies sum pooling.

$$\mathbf{t}_i = \mathbf{g}_i + \mathbf{l}_i, \tag{16}$$

Different aggregation mechanisms are applied to the outputs of LSA and GSA simultaneously.

We report the performance of these variants of aggregation mechanisms and the relative improvement of SRGA w.r.t them in Table 4. Based on the results, we observe the following findings:

- Obviously, SRGA$_{concat}$ with concatenation performs worst, since it ignores distinct contributions of LSA and GSA.
- SRGA$_{sa}$ with complex attention mechanism still falls behind SRGA. As the contribution of the local and global scope influence of each item varies according to users' shifting attentions, it is insufficient to only rely on the output of LSA and GSA to decide the corresponding weights.
- In contrast, SRGA with sum pooling achieves relatively better performance among variants. However, it just assumes LSA and GSA contribute equally for each item, which overlooks the impact of various user interests towards items.
- SRGA consistently yields the best performance on all metrics. It demonstrates that our proposed gate attention mechanism is more capable of aggregating information from LSA and GSA by taking advantage of powerful user representations.

5.6 Hyper-parameter Study

Effect of Window Size. As window size m determines the effective scope of local interactions in LSA, we vary m from 1 to 12 to analyze its effect on the performance of SRGA. As shown in Figure 3c and 3g, the best performance is achieved at the window size between 4 and 6, which approximately matches the average number of items presented within the screen. However, the performance drops drastically as m decreases below 2. This is mainly because that there are no neighboring items involved so that the interactions are limited to the item itself, and thus hurting the final performance. Further, we increase the window size above 8, and thus more items are included. Limited by the screen size in practice, larger m generally weakens the local scope representation and

Table 4: Results of variants of gate module.

	SRGA_{concat}	SRGA_{sum}	SRGA_{sa}	SRGA
NDCG@3	0.3861	0.3870	0.3868	**0.3879***
	(+0.47%)	(+0.23%)	(+0.28%)	-
NDCG@5	0.4527	0.4537	0.4534	**0.4543***
	(+0.35%)	(+0.13%)	(+0.20%)	-
NDCG@10	0.5267	0.5276	0.5272	**0.5285***
	(+0.34%)	(+0.17%)	(+0.25%)	-
MAP@3	0.3761	0.3764	0.3762	**0.3767***
	(+0.16%)	(+0.08%)	(+0.13%)	-
MAP@5	0.4074	0.4077	0.4075	**0.4084***
	(+0.25%)	(+0.17%)	(+0.22%)	-
MAP@10	0.4271	0.4276	0.4275	**0.4281***
	(+0.23%)	(+0.12%)	(+0.14%)	-

incurs noises. Coincided with the discussion of SRGA_GSA_{bi} in Section 5.5, a large bidirectional window could be detrimental to overall performance. Overall, it validates that LSA with proper window size is effective to capture the interactions among neighboring items in the local scope and benefits the performance of SRGA.

Table 5: Results of average gating weight as the number of gated attention layers.

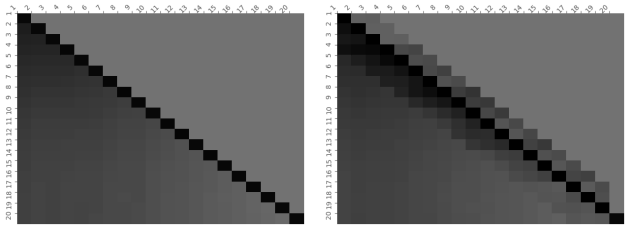
Gated Attention Layers	1	2	4	6	8
Average gating weight	**0.4929**	0.1204	0.0717	0.0069	0.0121
NDCG@10	**0.5285**	0.5181	0.5197	0.5133	0.5205
MAP@10	**0.4281**	0.4176	0.4186	0.4153	0.4208

Effect of Gated Attention Layer. At last, we study the performance of SRGA with the number of gated attention layers. As shown in Figure 3d and 3h, we report that the performance of SRGA declines with fluctuations as more gated attention layers stacked. Higher gated attention layer focuses on generating more representative global patterns by attending features from lower layers. Therefore, GSA, that characterizes interactions in the global scope, gradually dominates in higher gated attention layer of SRGA. Consequently, the locality effect of user behavior patterns is progressively ignored, which accords with the drop of the average gating weight that weighs the contribution of LSA as shown in Table 5. Additionally, more gated attention layers bring more model parameters and prone to overfitting, which could potentially hurt the overall performance. Based on the results, we conclude that a single gated attention layer can efficiently learn the powerful representation of user interests toward items in practice.

Model Complexity. As concluded from experimental results, SRGA is capable of achieving best performance with only one gated attention layer, which reduces a great number of parameters. On the contrary, PRM requires 4 transformer encoding layers with much larger model complexity and computation cost, and DLCM relys on RNN structure without parallel computation. Thus, SRGA is more efficient to train and much easier to deploy for inference than benchmark methods. In fact, we observe less inference time w.r.t PRM and DLCM after the deployment in live traffic.

5.7 Visualization

As discussed in Section 1, users leave feedback under the influence of adjacent items in a narrow visual window. To validate whether SRGA is capable of modeling locality of user behavior patterns, we visualize the attention weight with and without LSA in the real-world dataset. In Figure 4, both X- and Y-axis denote indexes of items, and each pixel represents the attention weight of an item-pair. The darker the pixel, the larger the attention weight thus the corresponding item-pair is more correlated.



(a) Without LSA. **(b) With LSA.**

Figure 4: Visualization of attention weight in SRGA.

In Figure 4a, we remove LSA and thus only GSA is in effect, resulting in that SRGA only captures the unidirectional influence among each item-pair across the entire list. From 4a, we conclude that items ranked at top have greater impacts on those at bottom than items ranked in the middle. In contrast, we observe noticeable clusters of darker pixels along the diagonal in Figure 4b as LSA included. Comparing Figure 4a with 4b, LSA benefits the overall performance as learned attention weights reflect the personalized representation of items in different scopes.

5.8 Online A/B Testing

To verify the effectiveness of SRGA model, we deployed some benchmark methods and carried out an online A/B testing. The accumulated data is enough to validate the results at a statistically significant level under the huge traffic. The initial ranking list is generated by a DNN-based LTR method, which is also the baseline. Our proposed SRGA model achieves 4.21% improvement of CTR w.r.t. the basline, which is aligned with the uplift achieved in offline experiments. In contrast, DLCM and PRM models marginally contribute to 0.99% and 1.03% improvement of CTR, respectively. It is worth noting that an increase like 0.1% in CTR value can bring huge benefits on a large scale recommendation system. We show that SRGA model significantly improves the primary metric by addressing *unidirectivity* and *locality* in the re-ranking context.

6 CONCLUSION

In this work, we formally define user behavior patterns through empirical analysis for the first time and propose a novel re-ranking framework for modeling the mutual influence in the large-scale recommendation system. Our approach adaptively integrates the mutual influences from distinct scopes: the GSA featuring the unidirectional impact of top items and the LSA strengthening the local comparison between neighboring items. We demonstrate the effectiveness and robustness of our proposed framework through extensive offline experiments and online A/B testing. The SRGA is deployed and serving a billion users in the recommendation feed, showing noteworthy lifts in user engagement. Furthermore, we conducted a detailed model analysis to justify the rationale and necessity behind the design. In the future, we would like to study SRGA with the appropriate loss functions for parameter estimation and improve the performance even further.

REFERENCES

[1] Qingyao Ai, Keping Bi, Jiafeng Guo, and W. Bruce Croft. 2018. Learning a Deep Listwise Context Model for Ranking Refinement. In *The 41st International ACM SIGIR Conference on Research Development in Information Retrieval (SIGIR '18)*. Association for Computing Machinery, New York, NY, USA, 135–144. https://doi.org/10.1145/3209978.3209985

[2] Qingyao Ai, Xuanhui Wang, Sebastian Bruch, Nadav Golbandi, Michael Bendersky, and Marc Najork. 2019. Learning Groupwise Multivariate Scoring Functions Using Deep Neural Networks. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR '19)*. Association for Computing Machinery, New York, NY, USA, 85–92. https://doi.org/10.1145/3341981.3344218

[3] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. *CoRR* abs/1607.06450 (2016). arXiv:1607.06450 http://arxiv.org/abs/1607.06450

[4] Irwan Bello, Sayali Kulkarni, Sagar Jain, Craig Boutilier, Ed Chi, Elad Eban, Xiyang Luo, Alan Mackey, and Ofer Meshi. 2019. Seq2Slate: Re-ranking and Slate Optimization with RNNs. *arXiv:1810.02019* (2019).

[5] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc Le. 2017. Massive Exploration of Neural Machine Translation Architectures. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, 1442–1451. https://doi.org/10.18653/v1/D17-1151

[6] Tom Brown, Benjamin Mann, Nick Ryder, and etc. Subbiah. [n.d.].

[7] Christopher Burges, Robert Ragno, and Quoc Le. [n.d.]. Learning to Rank with Nonsmooth Cost Functions. In *Advances in Neural Information Processing Systems*, B. Schölkopf, J. Platt, and T. Hoffman (Eds.). MIT Press.

[8] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to Rank Using Gradient Descent. In *Proceedings of the 22nd International Conference on Machine Learning (ICML '05)*. Association for Computing Machinery, New York, NY, USA, 89–96. https://doi.org/10.1145/1102351.1102363

[9] Chris J.C. Burges. 2010. From RankNet to LambdaRank to LambdaMART: An Overview. https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambdarank-to-lambdamart-an-overview/

[10] 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 (ICML '07)*. Association for Computing Machinery, New York, NY, USA, 129–136. https://doi.org/10.1145/1273496.1273513

[11] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2020. Bias and Debias in Recommender System: A Survey and Future Directions. *arXiv:2010.03240* (2020).

[12] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS 2016)*. Association for Computing Machinery, New York, NY, USA, 7–10. https://doi.org/10.1145/2988450.2988454

[13] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv:1412.3555* (2014).

[14] Ruocheng Guo, Xiaoting Zhao, Adam Henderson, Liangjie Hong, and Huan Liu. 2020. Debiasing Grid-Based Product Search in E-Commerce. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining (KDD '20)*. Association for Computing Machinery, New York, NY, USA, 2852–2860. https://doi.org/10.1145/3394486.3403336

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778. https://doi.org/10.1109/CVPR.2016.90

[16] L. C. Jain and L. R. Medsker. 1999. *Recurrent Neural Networks: Design and Applications* (1st ed.). CRC Press, Inc., USA.

[17] Kalervo Järvelin and Jaana Kekäläinen. 2000. IR Evaluation Methods for Retrieving Highly Relevant Documents. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '00)*. Association for Computing Machinery, New York, NY, USA, 41–48. https://doi.org/10.1145/345508.345545

[18] Thorsten Joachims. 2006. Training Linear SVMs in Linear Time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '06)*. Association for Computing Machinery, New York, NY, USA, 217–226. https://doi.org/10.1145/1150402.1150429

[19] Alexandros Karatzoglou, Linas Baltrunas, and Yue Shi. 2013. Learning to Rank for Recommender Systems. In *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys '13)*. Association for Computing Machinery, New York, NY, USA, 493–494. https://doi.org/10.1145/2507157.2508063

[20] Tie-Yan Liu. 2009. Learning to Rank for Information Retrieval. *Found. Trends Inf. Retr.* 3, 3 (mar 2009), 225–331. https://doi.org/10.1561/1500000016

[21] Liang Pang, Jun Xu, Qingyao Ai, Yanyan Lan, Xueqi Cheng, and Jirong Wen. 2020. SetRank: Learning a Permutation-Invariant Ranking Model for Information Retrieval. (2020), 499–508. https://doi.org/10.1145/3397271.3401104

[22] Changhua Pei, Yi Zhang, Yongfeng Zhang, Fei Sun, Xiao Lin, Hanxiao Sun, Jian Wu, Peng Jiang, Junfeng Ge, Wenwu Ou, and Dan Pei. 2019. Personalized Re-Ranking for Recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys '19)*. Association for Computing Machinery, New York, NY, USA, 3–11. https://doi.org/10.1145/3298689.3347000

[23] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 Datasets. abs/1306.2597 (2013).

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

[25] Alec Radford and Karthik Narasimhan. 2018. Improving Language Understanding by Generative Pre-Training.

[26] 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 (WSDM '08)*. Association for Computing Machinery, New York, NY, USA, 77–86. https://doi.org/10.1145/1341531.1341544

[27] Hamed Valizadegan, Rong Jin, Ruofei Zhang, and Jianchang Mao. 2009. Learning to Rank by Optimizing NDCG Measure. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems (NIPS'09)*. Curran Associates Inc., Red Hook, NY, USA, 1883–1891.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. [n.d.]. Attention is All you Need. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc.

[29] Benyou Wang, Lifeng Shang, Christina Lioma, Xin Jiang, Hao Yang, Qun Liu, and Jakob Grue Simonsen. 2021. On Position Embeddings in Bert. In *International Conference on Learning Representations (ICLR)*.

[30] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & Cross Network for Ad Click Predictions. In *Proceedings of the ADKDD'17 (ADKDD'17)*. Association for Computing Machinery, New York, NY, USA, Article 12, 7 pages. https://doi.org/10.1145/3124749.3124754

[31] Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. 2018. The LambdaLoss Framework for Ranking Metric Optimization. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM '18)*. Association for Computing Machinery, New York, NY, USA, 1313–1322. https://doi.org/10.1145/3269206.3271784

[32] Mark Wilhelm, Ajith Ramanathan, Alexander Bonomo, Sagar Jain, Ed H. Chi, and Jennifer Gillenwater. 2018. Practical Diversified Recommendations on YouTube with Determinantal Point Processes. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM '18)*. Association for Computing Machinery, New York, NY, USA, 2165–2173. https://doi.org/10.1145/3269206.3272018

[33] Xiaohui Xie, Jiaxin Mao, Yiqun Liu, Maarten de Rijke, Yunqiu Shao, Zixin Ye, Min Zhang, and Shaoping Ma. 2019. Grid-Based Evaluation Metrics for Web Image Search. In *The World Wide Web Conference (WWW '19)*. Association for Computing Machinery, New York, NY, USA, 2103–2114. https://doi.org/10.1145/3308558.3313514

[34] Jun Xu and Hang Li. 2007. AdaRank: A Boosting Algorithm for Information Retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*. Association for Computing Machinery, New York, NY, USA, 391–398. https://doi.org/10.1145/1277741.1277809

[35] Jinyun Yan, Zhiyuan Xu, Birjodh Tiwana, and Shaunak Chatterjee. 2020. Ads Allocation in Feed via Constrained Optimization. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining (KDD '20)*. Association for Computing Machinery, New York, NY, USA, 3386–3394. https://doi.org/10.1145/3394486.3403391

[36] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. 2007. A Support Vector Method for Optimizing Average Precision. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*. Association for Computing Machinery, New York, NY, USA, 271–278. https://doi.org/10.1145/1277741.1277790

[37] 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 (SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 1649–1652. https://doi.org/10.1145/3397271.3401333

[38] Tao Zhuang, Wenwu Ou, and Zhirong Wang. 2018. Globally Optimized Mutual Influence Aware Ranking in E-Commerce Search. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*. AAAI Press, 3725–3731.