MPGraf: a Modular and Pre-trained Graphformer for Learning to Rank at Web-Scale (Extended Abstract)*

Yuchen Li[1,2], **Haoyi Xiong**[1], **Linghe Kong**[2], **Zeyi Sun**[3], **Hongyang Chen**[3], **Shuaiqiang Wang**[1] and **Dawei Yin**[1]

[1]Baidu Inc., China
[2]Shanghai Jiao Tong University, China
[3]Zhejiang Lab, China
{yuchenli, linghe.kong}@sjtu.edu.cn, {haoyi.xiong.fr, dr.h.chen}@ieee.org, sunzeyi@zhejianglab.com, shqiang.wang@gmail.com, yindawei@acm.org

Abstract

Both Transformer and Graph Neural Networks (GNNs) have been used in learning to rank (LTR), however, they adhere to two distinct yet complementary problem formulations, i.e., ranking score regression based on query-webpage pairs and link prediction within query-webpage bipartite graphs, respectively. Though it is possible to pre-train GNNs or Transformers on source datasets and fine-tune them subject to sparsely annotated LTR datasets separately, the source-target distribution shifts across the pairs and bipartite graphs domains make it extremely difficult to integrate these diverse models into a single LTR framework at a web-scale. We introduce the novel MPGraf model, which utilizes a modular and capsule-based pre-training approach, aiming to incorporate regression capacities from Transformers and link prediction capabilities of GNNs cohesively. We conduct extensive offline and online experiments to evaluate the performance of MPGraf.

1 Introduction

The recent advancements in deep learning have notably ushered in a juxtaposition of numerous datasets and models to solve complex problems. In Learning to Rank (LTR), the use of both Transformers[Vaswani *et al.*, 2017; Li *et al.*, 2024] and Graph Neural Networks (GNNs) have taken center stage, each contributing its distinctive capabilities to the LTR problem formulations[Li *et al.*, 2022; Li *et al.*, 2023c]. While Transformers, such as context-aware self-attention model[Pobrotyn *et al.*, 2020], handle the ranking score regression based on *query-webpage pairs*, GNNs, e.g., LightGCN[He *et al.*, 2020], offer solutions for link prediction via query-webpage bipartite graphs. Although graphformer[Yang *et al.*, 2021] has been proposed to combine advantages from GNNs and Transformers for representation learning with textual graphs, there still lack of joint efforts from the two domains (i.e., query-webpage pairs and graphs) in LTR. In order to improve the performance of over-parameterized models like Transformers or GNNs, the paradigm of *pre-training* and *fine-tuning* has been extensively employed. This involves firstly training the models on large-scale source datasets in an unsupervised or self-supervised manner to develop their core representation learning capabilities [Qiang *et al.*, 2023]. Subsequently, the pre-trained models can be fine-tuned using a small number of annotated samples from the target datasets [Kirichenko *et al.*, 2022]. However, such paradigm could not be easily followed by the LTR models leveraging both query-webpage pairs and graphs together. Despite separate fine-tuning of GNN or Transformer models yielding results, the distribution shifts between source and target datasets across the pairs and bipartite graphs domains, coupled with the rich diversity of these models, present immense challenges when integrating them into a unified LTR framework applicable.

To solve this problem, we propose MPGraf—a modular and pre-trained graphformer for learning to rank at web-scale. Compared to the vanilla graphformers [Yang *et al.*, 2021], which parallelize GNN and Transformer modules for two-way feature extraction and predict with fused features, MPGraf can choose to either parallelize or stack these two modules for feature learning in a hybrid architectural design. Then, MPGraf leverages a three-step approach: (1) Graph Construction with Link Rippiling; (2) Representation Learning with Hybrid Graphformer; (3) Surgical Fine-tuning with Modular Composition, where the first step generates graph-based training data from sparsely annotated query-webpage pairs, then the second step pre-trains the MPGraf's hybrid graphformer model including both GNN and Transformer modules composited in either parallelizing or stacking ways, and finally MPGraf leverages a surgical fine-tuning strategy to adapt the target LTR dataset while overcoming cross-domain source-target distribution shifts. We carry out extensive offline experiments on a real-world dataset collected from a large-scale search engine. We also deploy MPGraf at the search engine and implement a series of online evaluations. The experiment results show that, compared to the state-of-the-art in webpage ranking, MPGraf could achieve the best performance on both offline datasets. Furthermore, MPGraf obtains significant improvements in online evaluations under fair comparisons.

*This work was initially presented at IEEE ICDM2023.

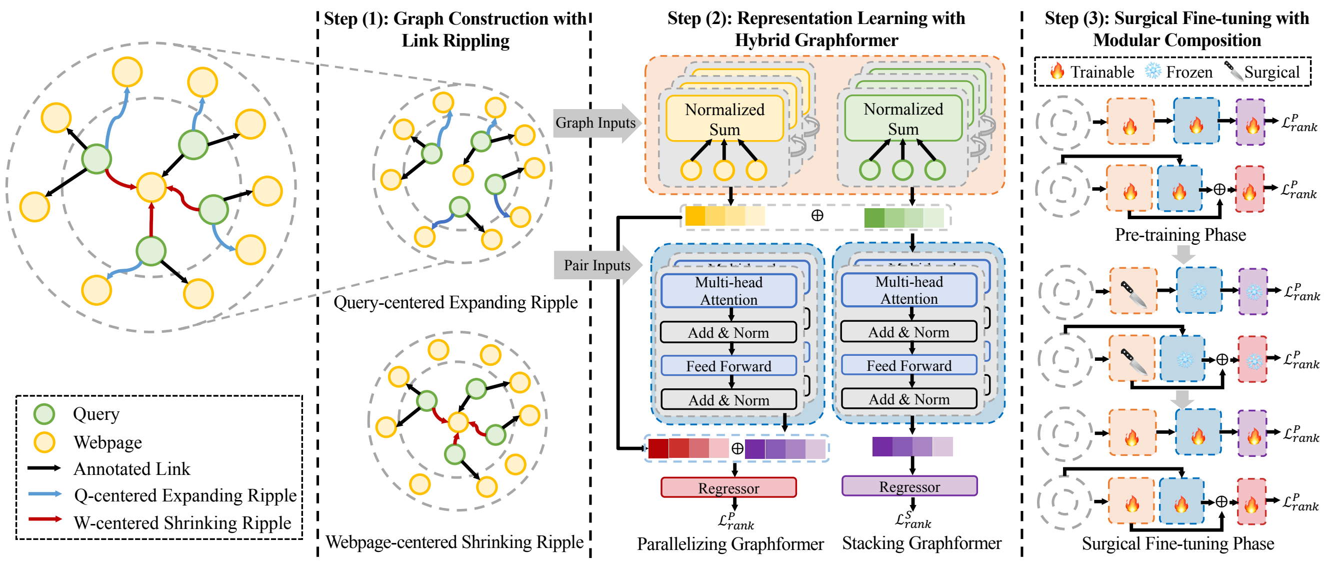


Figure 1: The framework of the proposed MPGraf.

2 The Proposed Model

Figure 1 sketches our proposed framework MPGraf. Specifically, MPGraf first conducts high-quality pseudo-label links for each unlabeled query-webpage pair by annotating all unlabeled pairs with pseudo-ranking scores, and then assigns every query webpages with high-ranking scores and also webpages with low scores to conduct *Query-centered Expanding Ripple* from training data. Next, MPGraf links every webpage to irrelevant queries with poor relevance scores to conduct *Webpage-centered Shrinking Ripple*. Given the query-webpage graph for every high-ranked query-webpage pair, MPGraf leverages a hybrid graphformer architecture to provide both Transformer and GNN modules with essential capacities of representation learning, where the graphformer consists of a GNNs module and a Transformer module. Eventually, MPGraf leverages a surgical fine-tuning strategy and transfers the pre-trained weights of both Transformer and GNN modules to adapt the target dataset while overcoming the source-target distribution shifts across graph and pair domains.

2.1 Graph Construction with Link Rippling

Query-centered Expanding Ripple. Given the set of queries $\mathbb{Q}$ and the set of webpages $\mathbb{D}$, MPGraf first obtains each possible query-webpage pair from both datasets, denoted as (q_i, d_i^j) for $\forall q_i \in \mathbb{Q}$ and $\forall d_i^j \in \mathcal{D}_i \subset \mathbb{D}$, i.e., the j^{th} webpage retrieved for the i^{th} query. For each query-webpage pair (q_i, d_i^j), MPGraf further extracts an m-dimensional feature vector $\boldsymbol{x}_{i,j}$ representing the features of the j^{th} webpage under the i^{th} query. Then, the labeled and unlabeled sets of feature vectors can be presented as $\mathcal{X}^L = \{(\boldsymbol{x}_{i,j}, y_j^i) | \forall (q_i, \mathcal{D}_i, \mathcal{Y}) \in \mathcal{X}^L \text{ and } \forall d_j^i \in \mathcal{D}_i\}$ and $\mathcal{X}^U = \{\boldsymbol{x}_{i,j} | \forall (q_i, \mathcal{D}_i) \in \mathcal{T}^U\}$. MPGraf further takes a self-tuning approach [Li *et al.*, 2023d; Li *et al.*, 2023a] to propagate labels from annotated query-webpage pairs to unlabeled ones.

Webpage-centered Shrinking Ripple. Though *Query-centered Expanding Ripple* algorithm could generate ranking scores for every query-webpage pair in training data, it is still difficult to construct webpage-centered graphs using predicted scores at full-scale. While every query connects to the webpages with high/low *pseudo* ranking scores, a webpage usually only connects to one or very limited highly-relevant queries and the number of webpages is much larger than that of effective queries from the perspective of webpages. Therefore, there needs to find irrelevant queries for every webpage. To conduct webpage-centered graphs for a webpage, MPGraf leverages a *Webpage-centered Shrinking Ripple* approach. Given a webpage, MPGraf retrieves all query-webpage pairs and builds a webpage-centered graph for every query-webpage with relevance scores higher than *1-fair* [Li *et al.*, 2023b]. Specifically, MPGraf randomly picks up a query that does not connect to the webpage as the irrelevant query, then forms the three (i.e., the webpage, a query where the webpage is highly ranked, and an irrelevant query) into a webpage-centered graph. Specifically, for a query q_i, MPGraf randomly chooses the webpage from the other query to conduct the negative samples d_j^{i-} and assigns the relevant score (i.e., 0 or 1) to represent poor relevance. Through this negative sampling method, MPGraf could build webpage-centered graphs for the webpage.

2.2 Representation Learning with Hybrid Graphformer

Given the query-webpage graphs for every high-ranked query-webpage pair, in this step, MPGraf leverages a Graph-Transformer (i.e., graphformer) architecture to extract the generalizable representation and enables LTR in an end-to-end manner. Specifically, graphformer consists of two modules: a GNN module and a Transformer module. According to the relative position between the two modules, graphformer could be categorized into two types: *Stacking Graphformer* and *Parallelizing Graphformer*.

Stacking Graphformer. Given the query-webpage graphs, MPGraf extracts the feature vector of each query and webpage. Specifically, the feature of query q_i and webpage d^i_j is denoted as $\boldsymbol{x}^{(n=0)}_{q^i}$ and $\boldsymbol{x}^{(n=0)}_{d^i_j}$, where n indicates the feature output from the n^{th} GNN layer. Next, the GNN module utilizes the query-webpage interaction graph to propagate the representations as $\boldsymbol{x}^{(n+1)}_{q^i} = \sum_{d^i_j \in \mathcal{N}_{q^i}} \frac{1}{Z} \boldsymbol{x}^{(n)}_{d^i_j}$; $\boldsymbol{x}^{(n+1)}_{d^i_j} = \sum_{q_i \in \mathcal{N}_{d^i_j}} \frac{1}{Z} \boldsymbol{x}^{(n)}_{q^i}$, where $\mathcal{N}_{q^i}$ and $\mathcal{N}_{d_{j,i}}$ represent the set of webpages that are relevant to query q^i and the set of queries that are relevant to webpage d^i_j, respectively. Moreover, $Z = \sqrt{|\mathcal{N}_{q^i}|}\sqrt{|\mathcal{N}_{d_{j,i}}|}$ is the normalization term. After N layers graph convolution operations, MPGraf combines the representations generated from each layer to form the final representation of query q^i and webpage d^i_j as $boldsymbol x_{q^i} = \sum_{n=0}^{N} \alpha_n \boldsymbol{x}^{(n)}_{q^i}$; $\boldsymbol{x}_{d^i_j} = \sum_{n=0}^{N} \alpha_n \boldsymbol{x}^{(n)}_{d^i_j}$, where $\alpha_n \in [0, 1]$ is a hyper-parameter to balance the weight of each layer representation. Then, MPGraf combines $\boldsymbol{x}_{q^i}$ and $\boldsymbol{x}_{d^i_j}$ to form the learned pair representation as $\boldsymbol{x}^G_{i,j}$.

Given the learned vector $\boldsymbol{x}^G_{i,j}$ of a query-webpage pair from the GNN module, MPGraf leverages a self-attentive encoder of Transformer to learn a generalizable representation $\boldsymbol{z}_{i,j}$. MPGraf first feeds $\boldsymbol{x}^G_{i,j}$ into a fully connected layer and produces a hidden representation. Later, MPGraf feeds the hidden representation into a self-attentive autoencoder, which consists of E encoder blocks of Transformer. Specifically, each encoder block incorporates a multi-head attention layer and a feed-forward layer, both followed by layer normalization. Eventually, MPGraf generates the learned representation $\boldsymbol{z}^S_{i,j}$ from the last encoder block. For each vector of each query-webpage pair, the whole training process can be formulated as $\boldsymbol{z}^S_{i,j} = f_\theta(\boldsymbol{x}^{(n=0)}_{q_i}, \boldsymbol{x}^{(n=0)}_{d^i_j})$, where θ is the set of parameters of *Stackinig Graphformer*.

Parallelizing Graphformer. In contrast to the aforementioned model, MPGraf parallelizes the GNN module and Transformer module to conduct *Parallelizing Graphformer*. Specifically, given the extracted feature vector of every query and webpage, MPGraf simultaneously feeds the vectors into two modules in *Parallelizing Graphformer*. Similar to *Stacking Graphformer*, MPGraf employs the GNN module to learn the query-webpage pair representation $\boldsymbol{x}^G_{i,j}$ from $\boldsymbol{x}^{(n=0)}_{q^i}$ and $\boldsymbol{x}^{(n=0)}_{d^i_j}$. Meanwhile, MPGraf first concatenates the feature of query q_i and webpage d^i_j to form the vector of query-webpage pair $\boldsymbol{x}^{(n=0)}_{i,j}$. Then, MPGraf utilizes the self-attentive encoder of Transformer to generate the learned representation $\boldsymbol{x}^T_{i,j}$. Given the learned representation $\boldsymbol{x}^G_{i,j}$ and $\boldsymbol{x}^T_{i,j}$, MPGraf concatenates two items as $\boldsymbol{x}^C_{i,j}$ and performs a linear projection to transform $\boldsymbol{x}^C_{i,j}$ into a low-dimensional vector space as $\boldsymbol{z}^P_{i,j}$.

Given the learned generalizable representation $\boldsymbol{z}^S_{i,j}$ or $\boldsymbol{z}^P_{i,j}$, MPGraf adopts an MLP-based regressor to compute the ranking score $\boldsymbol{s}_{i,j}$. Against the ground truth, MPGraf leverages the ranking loss function.

2.3 Surgical Fine-tuning with Modular Composition

Pre-training Phase. We pre-train MPGraf on massive LTR datasets towards relevance ranking and obtain the pre-trained GNN, Transformer and MLP modules. MPGraf is pre-trained on various distribution shift datasets to learn the representative capability by cross-domain ranking-task learning. After pre-training MPGraf on three datasets, we could get the pre-trained GNN, Transformer and MLP modules, which have preserved information in a standard way.

Surgical Fine-tuning Phase. Given the pre-trained three modules from the pre-training phase, we first tune the parameters in the GNN module and freeze the remaining parameters in other modules. After tuning the GNN module for several epochs, we jointly fine-tune the whole modules in MPGraf on the target dataset. Contrary to the conventional fine-tuning strategy of directly fine-tuning the whole model, freezing certain layer parameters can be advantageous since, based on the interplay between the pre-training and target datasets, some parameters in these modules, which have been trained on the pre-training dataset, may already approximate a minimum for the target distribution. Consequently, by freezing these layers, it becomes easier to generalize the target distribution.

3 Experiments

3.1 Experimental Setup

We conduct offline experiments using three public collections (i.e., MSLR-Web30K [Qin and Liu, 2013], MQ2007 [Qin and Liu, 2013], and MQ2008 [Qin and Liu, 2013]), as well as a commercial dataset with 15,000 queries and over 770,000 query-webpage pairs collected from a large-scale commercial search engine. Moreover, we use three evaluation metrics to assess the performance of ranking models, i.e., NDCG, Δ_{AB} [Chuklin *et al.*, 2015] and GSB [Zhao *et al.*, 2011].

In this work, we adopt different state-of-the-art ranking losses as RMSE, RankNet [Burges *et al.*, 2006], ListNet [Cao *et al.*, 2007] and NeuralNDCG [Pobrotyn and Białobrzeski, 2021]. Regarding the ranking model, we compare MPGraf with the state-of-the-art ranking model as MLP, CR [Pobrotyn *et al.*, 2020], XGBoost [Chen and Guestrin, 2016] and LightGBM [Ke *et al.*, 2017].

3.2 Offline Experimental Results

Comparative Results. The offline evaluation results for commercial data are presented in Table 1. Intuitively, we could find that MPGraf gains the best performance compared with all competitors on two metrics under various ratios of labeled data. Specifically, $\mathrm{MPGraf}^{\mathrm{S}}$ with NeuralNDCG achieves the improvement with 1.64%, 1.65%, 1.43% and 1.74% than MLP with NeuralNDCG on NDCG@10 under four ratios of labeled data on commercial data. From the comparative results, we observe that MPGraf could learn better generalizable representations with the graphformer architecture for downstream ranking tasks compared with baselines.

3.3 Online Experimental Results

Table 2 illustrates the performance improvements of the proposed models on Δ_{AB} and ΔGSB. We first observe that MP-

Methods	NDCG@5				NDCG@10			
	5%	**10%**	**15%**	**20%**	**5%**	**10%**	**15%**	**20%**
XGBoost	50.70	54.91	58.16	61.43	53.19	58.36	61.75	64.75
LightGBM	51.53	55.74	58.87	62.15	53.94	59.05	62.28	65.98
MLP_{RMSE}	50.12	54.45	57.62	59.64	53.42	57.86	61.34	64.76
$\text{MLP}_{\text{RankNet}}$	49.76	54.08	57.41	59.38	53.07	57.37	60.92	64.25
$\text{MLP}_{\text{ListNet}}$	50.48	54.91	58.05	59.92	53.61	58.04	61.41	64.82
$\text{MLP}_{\text{NeuralNDCG}}$	51.05	55.19	58.24	61.21	53.89	58.31	61.82	64.97
CR_{RMSE}	51.24	55.42	58.16	61.43	53.71	58.78	62.08	65.42
$\text{CR}_{\text{RankNet}}$	51.36	55.49	58.33	61.49	53.82	58.81	62.15	65.58
$\text{CR}_{\text{ListNet}}$	51.68	55.85	58.84	61.75	54.14	59.24	62.27	65.92
$\text{CR}_{\text{NeuralNDCG}}$	51.98	56.02	59.17	62.04	54.38	59.43	62.39	66.12
$\text{MPGraf}^{\text{S}}_{\text{RMSE}}$	51.30	55.27	58.55	61.74	54.55	59.04	62.35	65.80
$\text{MPGraf}^{\text{S}}_{\text{RankNet}}$	51.51	55.43	58.69	61.89	54.62	59.07	62.40	65.87
$\text{MPGraf}^{\text{S}}_{\text{ListNet}}$	52.27	56.21	59.46	62.68	55.32	59.79	63.12	66.62
$\text{MPGraf}^{\text{S}}_{\text{NeuralNDCG}}$	**52.83**	**56.79**	**60.08**	**63.32**	**55.53**	**59.96**	**63.25**	**66.71**
$\text{MPGraf}^{\text{P}}_{\text{RMSE}}$	51.39	55.36	58.65	61.89	54.61	59.07	62.40	65.91
$\text{MPGraf}^{\text{P}}_{\text{RankNet}}$	51.52	55.54	58.84	62.09	54.65	59.18	62.51	66.03
$\text{MPGraf}^{\text{P}}_{\text{ListNet}}$	52.34	56.39	59.70	62.97	55.44	59.84	63.20	66.72
$\text{MPGraf}^{\text{P}}_{\text{NeuralNDCG}}$	**52.91**	**56.98**	**60.25**	**63.51**	**55.67**	**60.05**	**63.42**	**66.94**

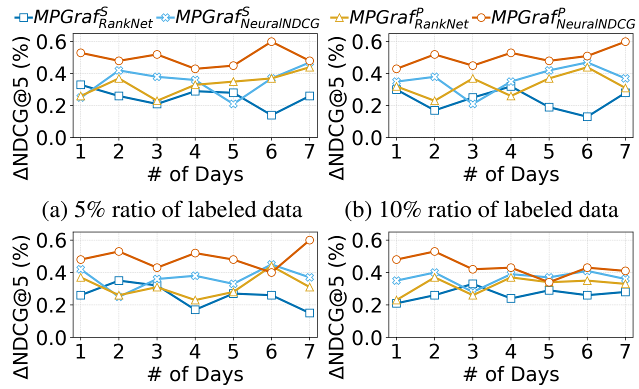
Table 1: Performance of MPGraf and baselines on commercial data.

Methods	Δ_{AB}		ΔGSB	
	Random	**Long Tail**	**Random**	**Long Tail**
Legacy System	-	-	-	-
$\text{MPGraf}^{\text{S}}_{\text{NeuralNDCG}}$	0.36%	0.45%	3.34%	5.50%
$\text{MPGraf}^{\text{P}}_{\text{NeuralNDCG}}$	**0.45%**	**0.58%**	**6.67%**	**7.50%**

Table 2: Performance improvements of online evaluation.

Graf with NeuralNDCG achieves substantial improvements for the online system on two metrics Specifically, our proposed models outperform the legacy system with 0.36% and 0.45% on Δ_{AB}, and achieve significant improvements with 3.34% and 6.67% on ΔGSB for random queries, respectively. Moreover, we could observe that MPGraf outperforms the legacy system for long-tail queries whose search frequencies are lower than 10 per week. In particular, under the long-tail scenario, *parallelizing graphformer*-based MPGraf with NeuralNDCG achieves the advantages of Δ_{AB} and ΔGSB are 0.58% and 7.50%.

Figure 2 presents the improvement of MPGraf with various losses compared with the *legacy system* on ΔNDCG@5. First, MPGraf could boost the performance compared with the online legacy system all day, which demonstrates that MPGraf is practical for improving the performance of the large-scale search engine. Moreover, we could observe that the trained MPGraf with NeuralNDCG under four ratios of labeled data achieves the largest improvements with 0.59%, 0.60%, 0.62% and 0.53%.



(a) 5% ratio of labeled data (b) 10% ratio of labeled data

(c) 15% ratio of labeled data (d) 20% ratio of labeled data

Figure 2: Online comparative performance (ΔNDCG@5) of MPGraf with various losses for 7 days (t-test with $p < 0.05$ over the baseline).

4 Conclusion

In this work, we focus on the use of a *Graph-Transformer* architecture to handle LTR in link predictions over query-webpage bipartite graphs and ranking score regressions based on query-webpage pairs. We propose MPGraf, where Transformer and GNN modules can be composited in either parallelizing or stacking architectures. MPGraf constructs web-scale query-webpage bipartite graphs with ranking scores as edges from pre-training LTR datasets. These graphs, along with the sparsely annotated query-webpage pairs, are used to pre-train the graphformer. The pre-trained weights of both modules are then transferred using the surgical fine-tuning strategy to adapt to the target dataset, which addresses the source-target distribution shifts across the graph and pair domains. Furthermore, we performed comprehensive offline and online experiments. Experimental results show the superior performance of MPGraf compared to competitors.

Acknowledgments

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

References

[Burges *et al.*, 2006] Christopher J. C. Burges, Robert Ragno, and Quoc Viet Le. Learning to rank with nonsmooth cost functions. In *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems*, pages 193–200, 2006.

[Cao *et al.*, 2007] 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.

[Chen and Guestrin, 2016] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

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

[He *et al.*, 2020] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR*, pages 639–648, 2020.

[Ke *et al.*, 2017] 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. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, pages 3146–3154, 2017.

[Kirichenko *et al.*, 2022] Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Last layer re-training is sufficient for robustness to spurious correlations. *arXiv preprint arXiv:2204.02937*, 2022.

[Li *et al.*, 2022] Yuchen Li, Haoyi Xiong, Linghe Kong, Rui Zhang, Dejing Dou, and Guihai Chen. Meta hierarchical reinforced learning to rank for recommendation: A comprehensive study in moocs. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 302–317, 2022.

[Li *et al.*, 2023a] Yuchen Li, Haoyi Xiong, Linghe Kong, Qingzhong Wang, Shuaiqiang Wang, Guihai Chen, and Dawei Yin. S2phere: Semi-supervised pre-training for web search over heterogeneous learning to rank data. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4437–4448, 2023.

[Li *et al.*, 2023b] Yuchen Li, Haoyi Xiong, Linghe Kong, Shuaiqiang Wang, Zeyi Sun, Hongyang Chen, Guihai Chen, and Dawei Yin. Ltrgcn: Large-scale graph convolutional networks-based learning to rank for web search. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 635–651. Springer, 2023.

[Li *et al.*, 2023c] Yuchen Li, Haoyi Xiong, Linghe Kong, Rui Zhang, Fanqin Xu, Guihai Chen, and Minglu Li. Mhrr: Moocs recommender service with meta hierarchical reinforced ranking. *IEEE Transactions on Services Computing*, 2023.

[Li *et al.*, 2023d] Yuchen Li, Haoyi Xiong, Qingzhong Wang, Linghe Kong, Hao Liu, Haifang Li, Jiang Bian, Shuaiqiang Wang, Guihai Chen, Dejing Dou, et al. Coltr: Semi-supervised learning to rank with co-training and over-parameterization for web search. *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[Li *et al.*, 2024] Yuchen Li, Haoyi Xiong, Linghe Kong, Jiang Bian, Shuaiqiang Wang, Guihai Chen, and Dawei Yin. Gs2p: a generative pre-trained learning to rank model with over-parameterization for web-scale search. *Machine Learning*, pages 1–19, 2024.

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

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

[Qiang *et al.*, 2023] Jipeng Qiang, Feng Zhang, Yun Li, Yunhao Yuan, Yi Zhu, and Xindong Wu. Unsupervised statistical text simplification using pre-trained language modeling for initialization. *Frontiers Comput. Sci.*, 17(1):171303, 2023.

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

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

[Yang *et al.*, 2021] Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Defu Lian, Sanjay Agrawal, Amit Singh, Guangzhong Sun, and Xing Xie. Graphformers: Gnn-nested transformers for representation learning on textual graph. *Advances in Neural Information Processing Systems*, 34:28798–28810, 2021.

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