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

Yuchen Li¹,², Haoyi Xiong¹, Linghe Kong², Zeyi Sun³, Hongyang Chen³, Shuaiqiang Wang¹ and Dawei Yin¹
¹Baidu Inc., China
²Shanghai Jiao Tong University, China
³Zhejiang Lab, China
{yuchenli, linghe.kong}@sjtu.edu.cn, {haoyi.xiong, 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 compositcd 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.
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 webpage 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 $Q$ and the set of webpages $D$, MPGraf first obtains each possible query-webpage pair from both datasets, denoted as $(q_i, d_j^l)$ for $\forall q_i \in Q$ and $\forall d_j^l \in D_l \subset D$, i.e., the $j^{th}$ webpage retrieved for the $i^{th}$ query. For each query-webpage pair $(q_i, d_j^l)$, MPGraf further extracts an $m$-dimensional feature vector $x_{i,j}$ 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 $X^L = \{(x_{i,j}, y_j^i) \mid y_j^i \in Y \} \subset X$ and $\forall d_j^l \in D_l \}$ and $X^U = \{(x_{i,j} \mid y_j^i \in \mathcal{T}^U \}$. 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 $d_j^-\ell$ 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 $d_j$ is denoted as $x^{(n=0)}_{q_i}$ and $x^{(n=0)}_{d_j}$, 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 $x^{(n+1)}_{q_i} = \sum_{d_j \in \mathcal{N}_{q_i}} \frac{1}{2} x^{(n)}_{d_j}; x^{(n+1)}_{d_j} = \sum_{q_i \in \mathcal{N}_{d_j}} \frac{1}{2} x^{(n)}_{q_i}$, where $\mathcal{N}_{q_i}$ and $\mathcal{N}_{d_j}$ represent the set of webpages that are relevant to query $q_i$ and the set of queries that are relevant to webpage $d_j$, 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 $d_j$ as $\text{bodysymbol} x^{n}_{q_i} = \sum_{n=0}^{N} \alpha_n x^{(n)}_{q_i}; \text{bodysymbol} x^{n}_{d_j} = \sum_{n=0}^{N} \alpha_n x^{(n)}_{d_j}$, where $\alpha_n \in [0, 1]$ is a hyper-parameter to balance the weight of each layer representation. Then, MPGraf combines $x^{n}_{q_i}$ and $x^{n}_{d_j}$ to form the learned pair representation as $z_{i,j}^{n}$.

Given the learned vector $x^{n}_{i,j}$ of a query-webpage pair from the GNN module, MPGraf leverages a self-attention encoder of Transformer to learn a generalizable representation $z_{i,j}$. MPGraf first feeds $x^{n}_{i,j}$ 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 $z_{i,j}^{S}$ from the last encoder block. For each vector of each query-webpage pair, the whole training process can be formulated as $z_{i,j}^{S} = f_{\theta}(x^{(n=0)}_{q_i}, x^{(n=0)}_{d_j})$, where $\theta$ is the set of parameters of Stacking 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 $x^{G}_{i,j}$ from $x^{(n=0)}_{q_i}$ and $x^{(n=0)}_{d_j}$. Meanwhile, MPGraf first concatenates the feature of query $q_i$ and webpage $d_j$ to form the vector of query-webpage pair $x^{(n=0)}_{i,j}$. Then, MPGraf utilizes the self-attentive encoder of Transformer to generate the learned representation $x^{T}_{i,j}$. Given the learned representation $x^{G}_{i,j}$ and $x^{T}_{i,j}$, MPGraf concatenates two items as $x^{C}_{i,j}$ and performs a linear projection to transform $x^{C}_{i,j}$ into a low-dimensional vector space as $z^{P}_{i,j}$.

Given the learned generalizable representation $z^{S}_{i,j}$ or $z^{P}_{i,j}$, MPGraf adopts an MLP-based regressor to compute the ranking score $s_{i,j}$. 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, $\Delta_{AB}$ [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, MPGraf 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 $\Delta_{AB}$ and $\Delta_{GSB}$. We first observe that MP-
Graf constructs web-
References


