Understanding and Modeling Job Marketplace with Pretrained Language Models

Yaochen Zhu
University of Virginia
Charlottesville, VA, USA
uqp4qh@virginia.edu

Liang Wu
LinkedIn Inc.
Sunnyvale, CA, USA
liawu@linkedin.com

Binchi Zhang
University of Virginia
Charlottesville, VA, USA
epb6gw@virginia.edu

Song Wang
University of Virginia
Charlottesville, VA, USA
sw3wv@virginia.edu

Qi Guo
LinkedIn Inc.
Sunnyvale, CA, USA
qguo@linkedin.com

Liangjie Hong
LinkedIn Inc.
Sunnyvale, CA, USA
liahong@linkedin.com

Luke Simon
LinkedIn Inc.
Sunnyvale, CA, USA
lsimon@linkedin.com

Jundong Li
University of Virginia
Charlottesville, VA, USA
jundong@virginia.edu

ABSTRACT

Job marketplace is a heterogeneous graph composed of interactions among members (job-seekers), companies, and jobs. Understanding and modeling job marketplace can benefit both job seekers and employers, ultimately contributing to the greater good of the society. However, existing graph neural network (GNN)-based methods have shallow understandings of the associated textual features and heterogeneous relations. To address the above challenges, we propose PLM4Job, a job marketplace foundation model that tightly couples pretrained language models (PLM) with job market graph, aiming to fully utilize the pretrained knowledge and reasoning ability to model member/job textual features as well as various member-job relations simultaneously. In the pretraining phase, we propose a heterogeneous ego-graph-based prompting strategy to model and aggregate member/job textual features based on the topological structure around the target member/job node, where entity type embeddings and graph positional embeddings are introduced accordingly to model different entities and their heterogeneous relations. Meanwhile, a proximity-aware attention alignment strategy is designed to dynamically adjust the attention of the PLM on ego-graph node tokens in the prompt, such that the attention can be better aligned with job marketplace semantics. Extensive experiments at LinkedIn demonstrate the effectiveness of PLM4Job.

CCS CONCEPTS

• **Information systems** → **Data mining**.

KEYWORDS

Large Language Model; Graph Mining; Job Marketplace

CIKM '24, October 21–25, 2024, Boise, ID, USA

ACM ISBN 979-8-4007-0436-9/24/10
https://doi.org/10.1145/3627673.3680036

ACM Reference Format:
Yaochen Zhu, Liang Wu, Binchi Zhang, Song Wang, Qi Guo, Liangjie Hong, Luke Simon, and Jundong Li. 2024. Understanding and Modeling Job Marketplace with Pretrained Language Models. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24), October 21–25, 2024, Boise, ID, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3627673.3680036

1 INTRODUCTION

Job marketplace is a pivotal component of our society [1, 4]. A job marketplace can be viewed as a heterogeneous graph of members (job seekers), jobs, and companies, where companies release job postings that members can apply for, and members establish social connections by following one another. In general, job postings contain job descriptions and requirements to recruit suitable talents, whereas members are typically associated with abundant self-provided textual features such as biographies, skills, experiences, etc., to increase the likelihood of securing good employment.

Various interesting tasks can be conducted on job marketplace to contribute to the welfare of its stakeholders. From the entity's perspective, since some members do not provide certain important attributes (e.g., skills) in their profile, member attribute prediction is essential to more precisely match them to potential job opportunities [11, 40]. Additionally, with the recent COVID epidemic, there is growing interest in predicting members' work mode preference (e.g., onsite, online, or hybrid), such that job postings can be pre-filtered to save limited recommendation budgets [37]. For the relational level task, it is beneficial to suggest members with other members to follow [6] or suitable jobs to apply for [38, 39, 41].

Graph neural networks (GNN) [31] can be used to model the job marketplace to tackle the aforementioned tasks [3, 7, 19]. For example, Zhu et al. [37] propose to model member-job interactions as a bipartite graph to predict members' work mode preference given their interacted jobs. In addition, Wang et al. [25] propose a heterogeneous GNN for job recommendations. Nevertheless, GNN-based approaches lack prior knowledge of the diverse member-member relations (e.g., follow, co-work) or member-job relations

(e.g., follow, apply, view) in the job marketplace, resulting in a limited understanding of the relationships among different entities. In addition, since most GNN-based methods adopt bag-of-word relations of the rich textual data associated with members and jobs, their understanding of textual information is unavoidably shallow.

Recently, more efforts have been devoted to using pretrained language models (PLMs) to tackle text-attributed graphs (TAG) [12, 18, 26, 29], where their encoded knowledge and reasoning ability can be fully utilized to understand the node textual features and their relations [15, 30]. The key challenge is to introduce graph structures to PLMs. Generally, there are two strategies to address the issue. One main strategy is to integrate auxiliary GNNs with PLMs, which either views the PLMs as node feature extractors [36], or uses GNN embeddings (projected into the PLM token embedding space) to represent the nodes in the PLM [21, 22]. However, these strategies unavoidably inherit the drawbacks of the auxiliary GNN and introduce extra computational overhead. Another strategy is to use natural language to describe the proximity relationship between nodes in the graph, e.g., using textual descriptions such as "`node_1 and node_2 are within` **`one-hop`**" to denote the connected relation between "`node_1`" and "`node_2`" [32]. However, since there is no evidence that such textual descriptions can properly guide the PLM to attend to the nodes based on the proximity relation in the graph, the graph structure is still loosely coupled with the PLM.

To address the above challenges, we propose a graph-oriented PLM, i.e., PLM4Job, to tightly couple the pretrained knowledge with the heterogeneous structure of the job market graph, which could serve as the foundation model for various downstream tasks on the job marketplace. Specifically, we first introduce member/job tokens to faithfully represent nodes in the job marketplace graph. Then, in the pretraining phase, we propose a novel heterogeneous ego-graph-based prompting strategy to model and aggregate member/job textual features based on the topological structure around the target member/job, where entity embeddings and graph positional embeddings are introduced accordingly to facilitate the PLM to understand various entities and their respective relationships in the job marketplace. In addition, a proximity-aware attention alignment strategy is introduced to dynamically adjust the attention of the backbone PLM on the ego-graph node tokens in the prompt, such that the attention of the PLM can be better aligned with job marketplace semantics. Finally, for node-level tasks, we introduce label tokens for efficient, hallucination-free predictions.

2 METHODOLOGY

In this section, we introduce the problem setting of treating the job marketplace as a heterogeneous text-attributed graph (TAG) and the proposed PLM4Job as a foundation model to tackle various entity-level and link-level downstream tasks.

2.1 Problem Formulation

Suppose we have a job marketplace with a set of members $\mathcal{U} = \{1, 2, \ldots, N_U\}$ and jobs $\mathcal{I} = \{N_U+1, N_U+2, \ldots, N_U+N_I\}$. Generally, each member or job is associated with rich textual features, such as biography, skills, résumé from the member side, and job descriptions (JD) from the job side. In addition, various relationships can be formed among members and jobs. For example, members can apply, click, and view job postings, where the observed relations can be recorded as $\mathcal{R}_{UI} \subseteq \{u \rightarrow i \,|u \in \mathcal{U}, i \in \mathcal{I}\}$. In addition, members can follow each other to form a professional social network, i.e., $\mathcal{R}_{UU} \subseteq \{u \rightarrow u' | u, u' \in \mathcal{U}\}$. Finally, member i is also associated with certain attributes of interests, which we denoted as y_i. Here, if we use $\mathcal{N} = \mathcal{U} \cup \mathcal{I}$ and $\mathcal{E} = \mathcal{R}_{UI} \cup \mathcal{R}_{UU}$ to denote the node and edge sets, and use $\mathcal{A} = \{U, I\}$ and $\mathcal{R} = \{R_{UI}, R_{UU}\}$ to denote the entity and relation sets, we can find that the job marketplace can be viewed as a text-attributed heterogeneous graphs $G = (\mathcal{N}, \mathcal{E}, \mathcal{A}, \mathcal{R})$.

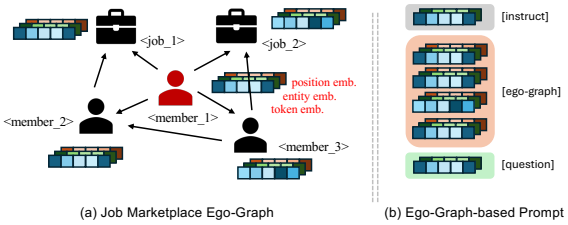

Figure 1: The job marketplace heterogeneous ego-graph and the corresponding ego-graph-based prompt.

Given that both node attributes and links can be missing from job market heterogeneous graph G, the objective of this paper is to understand and model G with graph-oriented pretrained language models (PLM), fully utilizing both graph structure and textual information to make strategic decisions that benefit all the stakeholders.

2.2 Ego-Graph-based Prompting

In this sub-section, we introduce an ego-graph-based prompting strategy to tightly couple the PLM with the heterogeneous ego-graph G_k of target node k. An overview is illustrated in Fig. 1.

2.2.1 ***Node Token for Heterogeneous Ego-Graph.*** For PLMs to understand G_k, we need first tokenize it into a sequence. Since the vocabulary of PLMs may not be able to faithfully represent graph nodes [32], we extend its vocabulary with node tokens and introduce learnable feature embeddings $\mathbf{Z} \in \mathbb{R}^{N \times K}$ to encode node features. Here, we use special tokens with bracket, e.g., "`<member_u>`", "`<job_i>`" to denote the newly introduced node tokens for member u and job i, respectively. In addition, to faithfully represent the entity and structural information in G_k, we introduce entity embeddings $\mathbf{E} \in \mathbb{R}^{|\mathcal{A}| \times K}$ and ego-graph positional embeddings $\mathbf{P} \in \mathbb{R}^{(D+1) \times K}$, where D is the maximum depth of G_k. The final token embedding for node $i \in G_k$ can be formulated as follows:

$$\mathbf{h}_i^{(0)} = \mathbf{z}_i + \mathbf{e}_{A_i} + \mathbf{p}_{dist(i,k)}, \tag{1}$$

where $dist(i, k)$ denote the shortest distance between node i and center node k. By converting each node in the heterogeneous ego-graph G_k into a token sequence with Eq. (1), node features and heterogeneous topological relationships can be well captured.

2.2.2 ***Feature Modeling.*** With the ego-graph node tokens and embeddings, we introduce the ego-graph-based prompting strategy to effectively learn the member/job token embeddings via language modeling (LM). We first discuss the feature learning step, which aims to encode the member/job textual features (e.g., member biographies and job descriptions) into the token embeddings.

W.L.O.G., if member k is the center node and biography is the textual feature, we first establish the prompt-completion pair $\left(\mathbf{x}_k^f, \mathbf{y}_k^f\right)$ as:

Feature Modeling

(a) Ego-graph-based prompt $\mathbf{x}_k^f$:

Given an ego-network in a job marketplace: ($\mathbf{x}^{ins}$) <member_k> (center)

<member_i> <member_j> (one-hop) <job_l> (one-hop) <job_m> (two-hop) <member_o> (two-hop) ($\mathbf{x}_k^g$),

the biography of the center member <member_i> (center) is : ($\mathbf{x}_k^{f,q}$)

(b) Completion $\mathbf{y}_k^f$:

A hard-working applied research scientist at LinkedIn.

Here, we use color **blue** to denote member tokens, yellow to denote job tokens, **green** to denote textual tokens, respectively. Ego-graph positional embeddings are denoted with sub-annotation. Specifically, the ego-graph-based prompt for member feature, i.e., $\mathbf{x}_k^f$, is composed of three parts: ***(i)*** instruction part $\mathbf{x}^{ins}$, which provides context regarding the job marketplace; ***(ii)*** ego-graph part $\mathbf{x}_k^g$, which includes the center node k and a randomly sub-sampled D-hop neighborhood as the job marketplace context; ***(iii)*** question part $\mathbf{x}_k^{f,q}$, which naturally leads to the completion $\mathbf{y}_k^f$.

We use causal language modeling [17] to learn the ego-graph token embeddings with prompt-completion pairs $\left(\mathbf{x}_k^f, \mathbf{y}_k^f\right)$. Specifically, we denote the backbone PLM with extended ego-graph tokens as $P_\Theta(x_t|\mathbf{x}_{<t})$, which generates the next token x_t based on the context token sequence $\mathbf{x}_{<t}$. The parameters $\Theta = \{\hat{\boldsymbol{\theta}}, \mathbf{Z}, \mathbf{E}, \mathbf{P}\}$ are composed of the pretrained PLM weights $\hat{\boldsymbol{\theta}}$ (which is frozen) and the newly introduced embeddings $\Theta_{hot} = \{\mathbf{Z}, \mathbf{E}, \mathbf{P}\}$. The loss of the feature modeling step for PLM4Job can be formulated as follows:

$$\mathcal{L}_k^f = \sum_{t=1} \log\left(P_\Theta\left(y_{k,t}^f \mid \mathbf{x}_k^f, \mathbf{y}_{k,<t}^f\right)\right). \tag{2}$$

Since the completion $\mathbf{y}_k^f$ contains only textual tokens, when optimizing the ego-graph token embeddings Θ_{hot} according to Eq. (2), we only calculate the softmax over all the textual tokens, where the stability of language modeling can be substantially enhanced [41].

2.2.3 ***Metapath-based Structural Modeling.*** After encoding the node textual features into the corresponding member and job token embeddings according to Eq. (2), we further aggregate the information based on the local job marketplace topology. Here, we define the metapath in a heterogeneous graph G as follows:

Definition 2.1. **Metapath [20].** A metapath ϕ is defined as a path in the form of $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \cdots \xrightarrow{R_l} A_{l+1}$, where $A_i \in \mathcal{A}$ and $R_j \in \mathcal{R}$ denote the entity and relation, respectively. The metapath can be abbreviated as $A_1A_2\cdots A_{l+1}$ with composite relationship $R_1 \circ R_2 \circ \cdots R_l$, where $\circ$ denotes composition operation on relations.

In the metapath-based structural modeling step, given a predefined set of candidate metapaths $\Phi = \{\phi_1, \phi_2, \cdots \phi_M\}$, for a center node k, we aim to transform each compatible metapath $\phi \in \Phi$ (compatible means $A_1^\phi = \text{type}(k)$) into an ego-graph-based prompt $\mathbf{x}_k^\phi$, with completion $\mathbf{y}_k^\phi$ constructed from a randomly shuffled sequence of the end entity A_{l+1}^ϕ. Then $\mathbf{y}_k^\phi$ is predicted based on $\mathbf{x}_k^\phi$ via language modeling. Through this strategy, information in the job market graph can be aggregated along the selected metapaths. The simplest ϕ is one-hop metapath, i.e., $\phi \in \Phi_1 = \{UU, UI, IU\}$. Here, we take the metapath $\phi = UU$ as an example, where the prompt, completion pair $\left(\mathbf{x}_k^{UU}, \mathbf{y}_k^{UU}\right)$ can be formulated as follows:

First-order Structural Modeling

(a) Ego-graph-based prompt $\mathbf{x}_k^{UU}$:

Given an ego-network in a job marketplace: ($\mathbf{x}^{ins}$) <member_k>

<member_i> <member_j> <job_l> <job_m> <member_o> ($\mathbf{x}_k^g$),

the center member <member_k> follows these members: ($\mathbf{x}_k^{q,UU}$)

(b) Completion $\mathbf{y}_k^{UU}$:

<member_r> <member_s> <member_t>

Here, we note that the question part $\mathbf{x}_k^{q,UU}$ of the ego-graph-based prompt $\mathbf{x}_k^{UU}$ specify the last relation R_1^ϕ (i.e., "follows") in the metapath ϕ, such that the encoded knowledge of the PLM can be fully utilized to facilitate the understanding of the relation and predict y_k^{UU}. In addition, only nodes not selected in the ego-graph G_k (i.e., not in $\mathbf{x}_k^g$) will be sampled in the completion $\mathbf{y}_k^{UU}$, which avoids the short cut of direct repeating nodes in the prompt.

Higher-order metapaths are complicated but are also necessary as they provide shortcuts for message passing among member and job nodes. Here, we use two-hop metapath as an example, where $\Phi_2 = \{UUU, UIU, IUU, IUI\}$. Previous work such as [32] use triples (A_1, A_2, A_3) to represent two-hop neighbors as token sequences, but this creates lengthy and redundant prompt due to repetition of intermediate nodes. In this paper, we propose a faster approximation strategy to represent high-order metapaths. Specifically, for center node k, we first establish a triple $T_k^\phi = \left(k, \mathcal{A}_{k,1}^\phi, \mathcal{A}_{k,2}^\phi\right)$, where $\mathcal{A}_{k,1}^\phi$ is the set of randomly sampled intermediate nodes starts from k, and $\mathcal{A}_{k,2}^\phi$ is a set of end nodes sampled from the union list of the end nodes connected with the intermediate nodes in $\mathcal{A}_{k,1}^\phi$ *with repetition* (such that important end nodes can be selected with higher probabilities). Here, we take two-hop metapath $\phi = UIU$ as an example. Based on the triple T_k^{UIU}, the prompt, completion pair $\left(\mathbf{x}_k^{UIU}, \mathbf{y}_k^{UIU}\right)$ for ϕ can be formulated as follows:

Higher-order Structural Modeling

(a) Ego-graph-based prompt $\mathbf{x}_k^{UIU}$:

Given an ego-network in a job marketplace: ($\mathbf{x}^{ins}$) <member_k>

<member_i> <member_j> <job_l> <job_m> <member_o> ($\mathbf{x}_k^g$),

the center member <member_k> is interested in these jobs:

<job_u> <job_v> <job_w> ($\mathbf{x}_k^{q,UI}$) the following users are also interested in some of these jobs: ($\mathbf{x}_k^{q,UIU}$)

(b) Completion $\mathbf{y}_k^{UIU}$:

<member_x> <member_y> <member_z>

From the above example, we can find that the ego-graph-based prompt for the metapath UIU, i.e., $\mathbf{x}_k^{UIU}$, is composed of an extra component $\mathbf{x}_k^{q,UI}$ that describes the intermediate relationship UI and the sampled final-step entities in $\mathcal{A}_{k,1}^{UIU}$, whereas the final relationship IU is described in the question part $\mathbf{x}_k^{q,UIU}$ that begs for completion with $\mathbf{y}_k^{UIU}$. Similar prompts can be established based on higher-order metapaths. The language modeling loss of structural modeling for metapath ϕ can be formulated as follows:

$$\mathcal{L}_k^{\phi} = \sum_{t=1} \log \left(P_\Theta \left(y_{k,t}^{\phi} \mid \mathbf{x}_k^{\phi}, \mathbf{y}_{k,<t}^{\phi} \right) \right). \tag{3}$$

Since the completion $\mathbf{y}_k^{\phi}$ is composed of either homogeneous member tokens or job tokens, we only calculate the softmax over the member/job token space to stabilize the language modeling process. For PLM with symmetric structure, i.e., the weights of the prediction head are tied with token embeddings (e.g., GPT-2 [16]), we also tie the weights of the prediction head with the corresponding member/job embeddings, whereas for other non-symmetric PLMs (e.g., LLaMA [23]), another set of randomly initialized embeddings needs to be introduced as the weights for the prediction head.

2.3 Proximity-Aware Attention Alignment

Another issue that hinders good modeling of job marketplace with PLMs is the **misalignment** of attention of the PLM with job market graph topology: When optimizing Θ according to Eqs. (2), (3), the PLM needs to attend to the prompt $\mathbf{x}_k^{\{f,\phi\}}$ and the already-generated completions $\mathbf{y}_{k,<t}^{\{f,\phi\}}$. However, the attention of the backbone PLM may not be well aligned with the member/job ego-graph G_k, as it may pay more attention to the recent tokens as for natural language, rather than to the important member/job nodes in the ego-graph G_k. To address this issue, we propose a proximity-aware attention alignment strategy to dynamically adjust the attention weights calculated by the PLM with proximity relations in the heterogeneous ego-graph G_k for both feature and structural modeling.

Here, the key insight is to view the tokens in the completion for *feature modeling*, i.e., $\mathbf{y}_k^{f}$, as associated with the center node k (whereas each token in the completion for structural modeling, i.e., $\mathbf{y}_k^{\phi}$, is associated with the node itself), and adjust the weights when attending to *member/job nodes* in the prompt based on their heterogeneous proximity in G_k. Specifically, the (un-normalized) attention when generating after the t-th *token* (assumed to be associated with the j-th *node*) on the t'-th token in the prompt (assumed to be the j'-th *node*) can be adjusted as follows:

$$\alpha_{tt'} = \frac{\left(\mathbf{h}_t \mathbf{W}_Q\right)\left(\mathbf{h}_{t'} \mathbf{W}_K\right)^T}{\sqrt{d}} + \boldsymbol{\psi}_{jj'}^T \mathbf{b}, \tag{4}$$

where $\mathbf{h}_{\{t,t'\}} \in \mathbb{R}^d$ is the latent representation of the $\{t,t'\}$-th token, $\mathbf{W}_Q$, $\mathbf{W}_K$ are the pretrained query and key matrices of the backbone PLM, respectively, $\boldsymbol{\psi}_{jj'} \in \mathbb{R}^{M+1}$ encodes the heterogeneous proximity between the node j and the attended node j', and $\mathbf{b} \in \mathbb{R}^{M+1}$ is the newly introduced learnable parameters. Specifically, given an ordered set of metapaths $\Phi = \{\phi_0, \phi_1, \phi_2, \cdots \phi_M\}$, where ϕ_0 denotes the trivial self metapath, $\boldsymbol{\psi}_{jj'}$ is defined as follows:

$$\psi_{i,jj'} = \begin{cases} 1, & \text{if } \phi_i \text{ exists between node } j,\ j', \\ 0, & \text{else.} \end{cases} \tag{5}$$

With the attention of generating completions to the member-job ego-graph G_k in the prompt dynamically adjusted according to Eqs. (4), (5), the attention of PLM4Job to member/job nodes can be better aligned with the heterogeneous structure of the job marketplace.

2.4 Task-Specific Finetuning

The feature and structural modeling aims to encode and aggregate member/job textual features and proximity information in the job marketplace into the member/job token embeddings, such that PLM4Job can understand the member-job heterogeneous ego-graphs. In this part, we introduce the task-specific finetuning strategies for PLM4Job to generalize it for various downstream tasks.

2.4.1 ***Node-Level Tasks.*** When conducting node-level tasks on the job marketplace graph G (e.g., member skill/work mode preference prediction), we first form an ego-graph-based prompt $\mathbf{x}_k^e$ with the target node k as the center nodes, which includes the instruction part $\mathbf{x}^{ins}$, the ego-graph part $\mathbf{x}_k^g$, the textual features of the center node part $\mathbf{x}_k^{e,f}$, and the question part $\mathbf{x}_k^{e,q}$ as follows:

Node-Level Task - Ego-Graph-based Prompt $\mathbf{x}_k^e$:

Given an ego-network in a job marketplace: ($\mathbf{x}^{ins}$) <member_k>
<member_i> <member_j> <job_l> <job_m> <member_o> ($\mathbf{x}_k^g$),
the biography of the center member <member_k> is:
A hard-working applied research scientist at LinkedIn ($\mathbf{x}_k^{e,f}$)
The member could possess the following skills: ($\mathbf{x}_k^{e,q}$)

Here, since node-level prediction focuses more on the node feature itself, we include textual feature into the prompt $\mathbf{x}_k^e$, i.e., $\mathbf{x}_k^{e,f}$. We only use member biography as an example, where other features such as member educational experience can be easily included.

We first embed the prompt $\mathbf{x}_k^e$ with the PLM and obtain the last-layer last-step hidden representation $\mathbf{h}_{k,-1}^{e,(-1)}$. Since directly generating the target class in natural language based on $\mathbf{h}_{k,-1}^{e,(-1)}$ via autoregression may lead to hallucination [10], e.g., outputting skills that are not in LinkedIn's standardized skill set, we introduce class tokens with embeddings $\mathbf{C}^n \in \mathbb{R}^{N_C \times d}$, where N_C is the number of classes, and predict the label of center node k as follows:

$$\hat{y}_k \sim \text{Categorical}\left(softmax\left(\mathbf{C}^n \cdot \mathbf{h}_{k,-1}^{e,(-1)} \right) \right). \tag{6}$$

For binary classification tasks, we could change the question part of the ego-graph-based prompt $\mathbf{x}_k^e$, i.e., $\mathbf{x}_k^{e,q}$, to "`does the member possess the skill {name}`". In this case, we have two class embeddings in $\mathbf{C}^n$ denoting the positive and negative predictions. We directly optimize the node class embeddings $\mathbf{C}^n$ via Eq. (6) by maximizing the log probability of the true class.

2.4.2 ***Link-Level Tasks.*** In this part, we focus on predicting one-hop relationships in the job marketplace G, i.e., predicting member-member following relations for *people you may know (PYMK)* recommendations, and predicting member-job interactions for *job you*

Table 1: Statistics of the LinkedIn job marketplace.

Dataset	#Mem	#Job	#Mem-Job	#Mem-Mem
LinkedIn	69,716	63,368	490, 768	827,844

may be interested in (JYMBII) recommendations, which form two most important business at LinkedIn. To predict the relationships, we first construct a similar ego-graph-based prompt $\mathbf{x}_k^l$ as follows:

Link-Level Task - Ego-Graph-based Prompt $\mathbf{x}_k^l$:

Given an ego-network in a job marketplace: ($\mathbf{x}^{ins}$) <member_k>
<member_i> <member_j> <job_l> <job_m> <member_o> ($\mathbf{x}_k^g$),
The center <member_k> currently follows:
<member_p> <member_q> <member_r> ($\mathbf{x}_k^{l,s}$)
The member may be interested following in these members: ($\mathbf{x}_k^{l,q}$)

From the above example, we can find that the difference between $\mathbf{x}_k^l$ and $\mathbf{x}_k^e$ is that, the observed end entities from the target relationship for the center node k, i.e., $\mathbf{x}_k^{l,s}$, is included in the prompt $\mathbf{x}_k^l$. During training, we randomly mask some observed entities to form $\mathbf{x}_k^{l,s}$ and stack all the hold-out neighbors as a multi-hot vector as the target $\mathbf{y}_k^l \in \{0,1\}^{\{U,I\}}$, which is generated as follows:

$$\hat{\mathbf{y}}_k^l \sim \text{Multinomial}\left(softmax\left(\mathbf{C}^{\{U,I\}} \cdot \mathbf{h}_{k,-1}^{l,(-1)}\right)\right). \tag{7}$$

Here, the weights $\mathbf{C}^{\{U,I\}}$ are the same as the weights of the LM prediction head from the first-order structural modeling for metapaths $\phi \in \{UU, UI\}$ in Eq. (3) (details see sub-section 2.2.3).

2.5 Pipeline Summary

In summary, when training PLM4Job, we first pre-heat the model by optimizing Eqs. (2), (3) for N_{pre} epochs. We then introduce the task-specific finetuning objective and train PLM4Job in an interleaving manner with Eq. (2), Eq. (3), and Eq. (6)/(7). Through this strategy, both member/job textual features and heterogeneous graph structure can be fully utilized to model the job marketplace.

2.6 Prediction

In the prediction phase of PLM4Job, we first randomly sample N_g ego-graphs to construct the prompt $\mathbf{x}_{k'}^{\{e,l\}}$ for the target node k'. We then calculate the categorical/multinomial probability according to Eq. (6)/(7) for node/link-level tasks and take the average. Finally, for node-level tasks, we use the class token with max probability as the prediction, whereas for link prediction tasks, we rank the multinomial likelihood and suggest the top M as the candidates.

3 EXPERIMENTS

3.1 Dataset Establishment

The studied job marketplace heterogeneous graph is established by sampling from one-day interactions between members and jobs from the United States at LinkedIn, where members' clicks, views, and applications of the job are recorded as the member-job edge in the graph under the relation "be interested in." In addition, members are connected if they work at the same company, representing the relation of "co-working." Textual attributes of the members include the headline (i.e., a brief intro. of the member under the name and photo of the member) and the biography. Textual features of the jobs include the title of the job, the company that posts the job, the job descriptions, and the skills required by the job. We collect the members' skills and work mode preferences to evaluate the node-level prediction ability of PLM4Job. Furthermore, for link-level tasks, we test the ability of PLM4Job to predict both member-job and member-member relations. The statistics of the established job marketplace heterogeneous graph are summarized in Table 1.

3.1.1 ***Implementation Details.*** Since the decisions on the job marketplace need to be fast at LinkedIn, we use a comparatively small PLM, i.e., GPT-2 [16] with 768-dimensional token embeddings and vocabulary size of 50,257, as the PLM backbone for PLM4Job. For the metapath-based structural modeling (see Section 2.2.3), we select six metapaths $\Phi = \{UI, UU, IU, UIU, UUI, IUI\}$, where in each epoch, we randomly select one of the one-hop meta-paths and one of the two-hop meta-paths for the structural information aggregation. During the training stage, we first optimize the newly introduced ego-graph token embeddings (see Eq. (1)) via self-supervised feature/structural modeling as with Eqs. (2) and (3) for ten epochs to warm up the model. Then, we add the task-specific finetuning objective to subsequent epochs, where we alternately train the PLM4Job model according to Eq. (2), Eq. (3), Eq. (6)/(7) for 100 epochs. For the node-level tasks, we randomly select 15% nodes with labels as the validation set and another 15% for testing, where accuracy and F1-score are used as the metrics. For the link-level tasks, we evaluate the PLM4Job on nodes with more than five target links, where for each of such nodes, 60% of the links are included for training, 20% are held out for validation, and another 20% for testing, where ranking-based metrics such as Recall@M and NDCG@M are used to measure the performance.

3.2 Baselines

We compare PLM4Job with various baselines on different downstream tasks on the job marketplace. Specifically, the baselines used in this paper can be categorized into three classes: ***(i)*** graph neural network (GNN)-based methods, such as GCN [14], GAT [24], as well as GNNs specifically designed for heterogeneous graphs, such as the heterogeneous GNN (HetGNN) [9] and heterogeneous graph attention network (HAN) [27]; ***(ii)*** graph transformer-based methods such as the graphormer (GT) [33], the ego-graph-based transformer, Gophormer [35] and the heterogeneous graph transformer (HGT) [9]; ***(iii)*** the PLM-based method, i.e., InstructGCL [32]. In addition, we introduce two more baselines, i.e., SGL-Text [28] and JMMFR (graph-based) [37] for node-level tasks, and LightGCN (graph-based) [8] and P5 (PLM-based) [5] for link-level tasks.

3.3 Node-Level Tasks

In this subsection, we show the experiments of PLM4Job on node-level tasks on the LinkedIn job marketplace. Specifically, we are interested in two tasks, i.e., **member skill prediction**, which aims to predict whether a member has coding-related or management-related skills, and **work mode preference prediction**, which aims to predict whether a member is willing to take an online/onsite job.

Table 2: Comparison between PLM4Job and baselines on the node-level tasks on LinkedIn job marketplace modeling.

Skill	Coding-Related		Manage-Related	
Dataset	Accuracy	F1-score	Accuracy	F1-score
MLP	0.7578	0.6919	0.6271	0.5405
GCN [14]	0.7810	0.7263	0.6784	0.6026
GAT [24]	0.8053	0.7318	0.6828	0.5950
SGL-Text [28]	0.7900	0.7254	0.6933	0.6148
JMMFR [37]	0.8046	0.7469	0.7008	0.6196
HetGNN [34]	0.8129	0.7435	0.6952	0.6169
HAN [27]	0.8084	0.7280	0.6990	0.6284
GT [33]	0.7733	0.7264	0.6902	0.5997
HGT [9]	0.7859	0.7315	0.6914	0.6036
Gophormer [35]	0.7925	0.7328	0.6805	0.6273
InstructGCL [32]	0.8103	0.7490	0.7052	0.6344
PLM4Job	**0.8187**	**0.7568**	**0.7187**	**0.6459**

Pref.	Onsite Jobs		Online Jobs	
Dataset	Accuracy	F1-score	Accuracy	F1-score
MLP	0.5559	0.4368	0.5746	0.4871
GCN [14]	0.6048	0.5104	0.6802	0.5535
GAT [24]	0.5713	0.5091	0.6679	0.5460
SGL-Text [28]	0.5825	0.5078	0.6750	0.5589
JMMFR [37]	0.6094	0.5236	0.6696	0.5791
HetGNN [34]	0.6100	0.5287	0.6737	0.5641
HAN [27]	0.6084	0.5112	0.6784	0.5826
GT [33]	0.6035	0.5120	0.6203	0.5477
HGT [9]	0.6051	0.5139	0.6417	0.5582
Gophormer [35]	0.5996	0.5010	0.6629	0.5714
InstructGCL [32]	0.6187	0.5245	0.6821	0.5859
PLM4Job	**0.6312**	**0.5393**	**0.6937**	**0.5924**

Since a member can have multiple skills and prefer multiple types of work modes, we model them as different binary classification problems. Both of these can significantly benefit the member-job matching at LinkedIn for better job recommendation results.

3.3.1 ***Comparison with Baselines.*** We first compare the proposed PLM4Job with the baselines introduced in Section 3.2, where the results are summarized in Table 2. From Table 2, we can find that heterogeneous GNNs generally show better performance than the normal GNN models due to their explicit consideration of different relations in the job marketplace graph. However, since these models use bag-of-word representations to model member/job textual features, their shallow understanding of important textual features leads to overall unsatisfactory results. For the graph transformer (GT)-based methods, HGT can distinguish heterogeneous relationships in the job market graph, but as a global model, it may not be able to fully utilize the local information for predictions. Gophormer is specifically designed for ego-graphs, but it does not consider the heterogeneous structure in the job marketplace. Most importantly, although GT-based methods have a similar underlying transformer structure as the PLM, these models are not pre-trained on large datasets and do not contain prior knowledge of the natural language. Therefore, their understanding of the member/job textual features as well as their relationship in the job marketplace is also shallow. As a PLM-based graph mining algorithm, InstructGCL performs the best among all the baselines as it utilizes the pretrained knowledge of PLMs. However, it does not consider the heterogeneous relationships in the job market graph. In addition, the proximity information is described via natural language such as "one-hop," etc., which may not faithfully direct the attention of the PLM according to the proximity information in the heterogeneous job marketplace ego-graph. In contrast, by tightly coupling the heterogeneous local structure of job marketplace graph with the pre-trained knowledge of the PLM, the proposed PLM4Job achieves the best results on the four datasets across all the metrics.

Table 3: Ablation study for PLM4Job on node-level tasks.

Skill	Coding-Related		Manage-Related	
Dataset	Accuracy	F1-score	Accuracy	F1-score
PLM4Job-NE	0.8129	0.7501	0.7090	0.6422
PLM4Job-NA	0.8094	0.7335	0.7046	0.6368
PLM4Job-N2	0.8073	0.7274	0.6813	0.6237
PLM4Job	**0.8187**	**0.7568**	**0.7187**	**0.6459**

Pref.	Onsite Jobs		Online Jobs	
Dataset	Accuracy	F1-score	Accuracy	F1-score
PLM4Job-NE	0.6281	0.5336	0.6841	0.5809
PLM4Job-NA	0.6248	0.5312	0.6786	0.5781
PLM4Job-N2	0.6125	0.5237	0.6760	0.5754
PLM4Job	**0.6312**	**0.5393**	**0.6937**	**0.5924**

3.3.2 ***Ablation Studies.*** In this part, we conduct ablation study to show the effectiveness of the ego-graph-based prompt (see Section 2.2) and the proximity-aware attention alignment strategy (see Section 2.3). Specifically, three ablation models are introduced on PLM4Job, where *PLM4Job-NE* removes the entity and graph positional embeddings, *PLM4Job-NA* removes the proximity-aware attention alignment module, *PLM4Job-N2* removes the second-order meta-paths in structural modeling. The results are summarized in Table 3. For Table 3, we can find that proximity-based attention alignment contributes significantly to the superior performance of PLM4Job, which demonstrates the misalignment of the attention original PLM with the proximity relations in the heterogeneous graph structure. In addition, the combination of entity and ego-graph positional embeddings facilitates PLM4Job to well distinguish different nodes in the heterogeneous job marketplace ego-graph.

3.4 Link-Level Tasks

In this sub-section, we show the experimental results of link-level prediction tasks on the LinkedIn job marketplace. Specifically, we focus on predicting member-job interactions (i.e., JYMBII prediction) and member-member interactions (i.e., PYMK prediction).

3.4.1 ***Comparison Results.*** Similarly, we first compare PLM4Job with various GNN-based, GT-based, and PLM-based baselines, where the results are summarized in Table 4. From Table 4, we can find that, generally, PLM4Job outperforms most of the GNN/GT/PLM-based baselines, which demonstrates its ability to generalize to link prediction tasks on the job marketplace. In addition, ablation

Table 4: Comparison between PLM4Job and various baselines on the link-level tasks for job marketplace modeling.

Link	Member-Job Interaction		
Dataset	Recall@20	Recall@40	NDCG@100
Dual-MLP	0.0819	0.1318	0.0703
GCN [14]	0.1280	0.1905	0.0946
GAT [24]	0.1204	0.1828	0.0912
LightGCN [8]	0.1351	0.1985	0.1027
HetGNN [34]	0.1331	0.1957	0.1025
HAN [27]	0.1386	0.1971	0.1089
GT [33]	0.1013	0.1830	0.0976
HGT [9]	0.1206	0.1929	0.0970
Gophormer [35]	0.1173	0.1852	0.0947
P5 [5]	0.1412	0.2116	0.1049
InstructGCL [32]	0.1437	0.2158	0.1104
PLM4Job	**0.1545**	**0.2210**	**0.1153**

Link	Member-Member Interaction		
Dataset	Recall@20	Recall@40	NDCG@100
Dual-MLP	0.1199	0.1843	0.1085
GCN [14]	0.1484	0.2326	0.1421
GAT [24]	0.1577	0.2490	0.1578
LightGCN [8]	0.1584	0.2501	0.1597
HetGNN [34]	0.1462	0.2285	0.1402
HAN [27]	0.1513	0.2371	0.1456
GT [33]	0.1437	0.2312	0.1483
HGT [9]	0.1470	0.2433	0.1537
Gophormer [35]	0.1526	0.2498	0.1584
P5 [5]	0.1801	0.2575	0.1695
InstructGCL [32]	0.1795	0.2531	0.1762
PLM4Job	**0.1953**	**0.2684**	**0.1809**

studies are also conducted for the link prediction task, where the introduced ablation models are the same as the ones used in subsection 3.3.2. The results are summarized in Table 5. From the Table, we can find that all components of the proposed PLM4Job also contribute positively to its final superior results for link-level tasks.

3.5 Deployment of PLM4Job Embeddings

PLM4Job intends to serve as the foundation model for the LinkedIn job retrieval system. At LinkedIn, the L1 retrieval model is evaluated from the user feedback on the L2 ranking model. Since PLM4Job is expensive to deploy directly, we extract the member/job token embeddings from the trained PLM4Job model (takes O(1) complexity), reduce their dimension, and deploy them on two of the most important systems at LinkedIn: *JYMBII* and *PYMK*. We name the two-tower model with PLM4Job embeddings as PLM4Job-Emb.

We compare the PLM4Job-Emb model with another model that adds the embeddings of M6-Rec [2], i.e., a PLM-based matching method for recommendations (where we denote the model as M6-Rec-Emb), as well as the original two tower model. Specifically, we randomly split the members into three folds and evaluate the three models on users' feedback on the L2 ranking model accumulated in a week. From Table 6, we can find that, adding PLM4Job member/job embeddings can improve the performance of the existing two-tower

Table 5: Ablation Study for PLM4Job on link-level tasks.

Link	Member-Job Interaction		
Dataset	Recall@20	Recall@40	NDCG@100
PLM4Job-NE	0.1502	0.2188	0.1131
PLM4Job-NA	0.1477	0.2095	0.1046
PLM4Job-N2	0.1410	0.2163	0.1094
PLM4Job	**0.1545**	**0.2210**	**0.1153**

Link	Member-Member Interaction		
Dataset	Recall@20	Recall@40	NDCG@100
PLM4Job-NE	0.1927	0.2675	0.1798
PLM4Job-NA	0.1761	0.2503	0.1716
PLM4Job-N2	0.1809	0.2517	0.1753
PLM4Job	**0.1953**	**0.2684**	**0.1809**

Table 6: Deploy PLM4Job embeddings to the two-tower models on the LinkedIn JYMBII and PYMK systems.

Online	Member-Job Interaction (JYMBII)		
Dataset	Recall@20	Recall@40	NDCG@100
Two-Tower	0.1353	0.1937	0.0980
M6-Rec-Emb	0.1409	0.2062	0.1058
PLM4Job-Emb	**0.1426**	**0.2099**	**0.1093**

Online	Member-Member Interaction (PYMK)		
Dataset	Recall@20	Recall@40	NDCG@100
Two-Tower	0.1475	0.2360	0.1437
M6-Rec-Emb	0.1505	0.2374	0.1481
PLM4Job-Emb	**0.1593**	**0.2446**	**0.1528**

model at LinkedIn (which includes embeddings from internally trained BERT [13] and GNNs), which further demonstrates the ability of PLM to serve as foundation models for job marketplace and adapt to downstream tasks with effectiveness and efficiency.

4 CONCLUSION

In this paper, we proposed PLM4Job, a graph-oriented pre-trained language model, to serve as the foundation model for job marketplace modeling. Specifically, we first propose an ego-graph-based prompt to facilitate the PLM to understand the features, relations, and local structure of the job marketplace with the pretrained knowledge. In addition, a proximity-aware attention alignment strategy is proposed to align the attention of the PLM with the heterogeneous proximity relations among members and jobs in the job marketplace ego-graph. Extensive experiments on LinkedIn real-world data demonstrate the effectiveness of PLM4Job.

ACKNOWLEDGMENT

This work is supported in part by the National Science Foundation under grants (IIS-2006844, IIS-2144209, IIS-2223769, CNS-2154962, BCS-2228534, and CMMI-2411248), and the Commonwealth Cyber Initiative Awards under grant (VV-1Q24-011).

REFERENCES

[1] Fedor Borisyuk, Liang Zhang, and Krishnaram Kenthapadi. 2017. LiJAR: A system for job application redistribution towards efficient career marketplace. In *SIGKDD*. 1397–1406.

[2] Zeyu Cui, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. 2022. M6-Rec: Generative pretrained language models are open-ended recommender systems. *arXiv preprint arXiv:2205.08084* (2022).

[3] Corné De Ruijt and Sandjai Bhulai. 2021. Job recommender systems: A review. *arXiv preprint arXiv:2111.13576* (2021).

[4] Yingpeng Du, Di Luo, Rui Yan, Xiaopei Wang, Hongzhi Liu, Hengshu Zhu, Yang Song, and Jie Zhang. 2024. Enhancing job recommendation through llm-based generative adversarial networks. In *AAAI*, Vol. 38. 8363–8371.

[5] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as language processing (RLP): A unified pretrain, personalized prompt & predict paradigm (P5). In *RecSys*. 299–315.

[6] Sahin Cem Geyik, Stuart Ambler, and Krishnaram Kenthapadi. 2019. Fairness-aware ranking in search & recommendation systems with application to linkedin talent search. In *SIGKDD*. 2221–2231.

[7] Linxin Guo, Yaochen Zhu, Min Gao, Yinghui Tao, Junliang Yu, and Chen Chen. 2024. Consistency and Discrepancy-Based Contrastive Tripartite Graph Learning for Recommendations. *SIGKDD*.

[8] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and powering graph convolution network for recommendation. In *SIGIR*. 639–648.

[9] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *WWW*. 2704–2710.

[10] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232* (2023).

[11] Kameni Florentin Flambeau Jiechieu and Norbert Tsopze. 2021. Skills prediction based on multi-label resume classification using CNN with model predictions explanation. *Neural Computing and Applications* 33 (2021), 5069–5087.

[12] Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. 2023. Large Language Models on Graphs: A Comprehensive Survey. *arXiv preprint arXiv:2312.02783* (2023).

[13] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*. 4171–4186.

[14] Thomas N Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[15] Haochen Liu, Song Wang, Yaochen Zhu, Yushun Dong, and Jundong Li. 2024. Knowledge Graph-Enhanced Large Language Models via Path Selection. In *ACL*.

[16] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

[17] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[18] Xubin Ren, Jiabin Tang, Dawei Yin, Nitesh Chawla, and Chao Huang. 2024. A Survey of Large Language Models for Graphs. *SIGKDD*.

[19] Walid Shalaby, BahaaEddin AlAila, Mohammed Korayem, Layla Pournajaf, Khalifeh AlJadda, Shannon Quinn, and Wlodek Zadrozny. 2017. Help me find a job: A graph-based approach for job recommendation at scale. In *IEEE Big Data*. 1544–1553.

[20] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *PVLDB* 4, 11 (2011), 992–1003.

[21] Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2023. GraphGPT: Graph instruction tuning for large language models. *arXiv preprint arXiv:2310.13023* (2023).

[22] Yijun Tian, Huan Song, Zichen Wang, Haozhu Wang, Ziqing Hu, Fang Wang, Nitesh V Chawla, and Panpan Xu. 2023. Graph neural prompting with large language models. *arXiv preprint arXiv:2309.15427* (2023).

[23] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[24] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[25] Hao Wang, Wenchuan Yang, Jichao Li, Junwei Ou, Yanjie Song, and Yingwu Chen. 2023. An improved heterogeneous graph convolutional network for job recommendation. *Engineering Applications of Artificial Intelligence* 126 (2023), 107147.

[26] Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, et al. 2023. Knowledge editing for large language models: A survey. *arXiv preprint arXiv:2310.16218* (2023).

[27] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *WWW*. 2022–2032.

[28] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *SIGIR*. 726–735.

[29] Likang Wu, Zhaopeng Qiu, Zhi Zheng, Hengshu Zhu, and Enhong Chen. 2024. Exploring large language model for graph data understanding in online job recommendations. In *AAAI*, Vol. 38. 9178–9186.

[30] Xuansheng Wu, Haiyan Zhao, Yaochen Zhu, Yucheng Shi, Fan Yang, Tianming Liu, Xiaoming Zhai, Wenlin Yao, Jundong Li, Mengnan Du, et al. 2024. Usable XAI: 10 strategies towards exploiting explainability in the LLM era. *arXiv preprint arXiv:2403.08946* (2024).

[31] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE TNNLS* 32, 1 (2020), 4–24.

[32] Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. 2023. Natural language is all a graph needs. *arXiv preprint arXiv:2308.07134* (2023).

[33] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do transformers really perform badly for graph representation?. In *NeurIPS*, Vol. 34. 28877–28888.

[34] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *SIGKDD*. 793–803.

[35] Jianan Zhao, Chaozhuo Li, Qianlong Wen, Yiqi Wang, Yuming Liu, Hao Sun, Xing Xie, and Yanfang Ye. 2021. Gophormer: Ego-graph transformer for node classification. *arXiv preprint arXiv:2110.13094* (2021).

[36] Jason Zhu, Yanling Cui, Yuming Liu, Hao Sun, Xue Li, Markus Pelger, Tianqi Yang, Liangjie Zhang, Ruofei Zhang, and Huasha Zhao. 2021. Textgnn: Improving text encoder via graph neural network in sponsored search. In *WWW*. 2848–2857.

[37] Qinyi Zhu, Liang Wu, Qi Guo, and Liangjie Hong. 2022. Remote Work Optimization with Robust Multi-channel Graph Neural Networks. *arXiv preprint arXiv:2209.03150* (2022).

[38] Yaochen Zhu and Zhenzhong Chen. 2022. Mutually-regularized dual collaborative variational auto-encoder for recommendation systems. In *WWW*. 2379–2387.

[39] Yaochen Zhu and Zhenzhong Chen. 2022. Variational bandwidth auto-encoder for hybrid recommender systems. *IEEE TKDE* 35, 5 (2022), 5371–5385.

[40] Yaochen Zhu, Jing Ma, Liang Wu, Qi Guo, Liangjie Hong, and Jundong Li. 2023. Path-specific counterfactual fairness for recommender systems. In *SIGKDD*. 3638–3649.

[41] Yaochen Zhu, Liang Wu, Qi Guo, Liangjie Hong, and Jundong Li. 2024. Collaborative large language model for recommender systems. In *WWW*. 3162–3172.