Pre-training Graph Transformer with Multimodal Side Information for Recommendation

Yong Liu[1], Susen Yang[3], Chenyi Lei[3,4], Guoxin Wang[3,5], Haihong Tang[3], Juyong Zhang[4]
Aixin Sun[2], Chunyan Miao[1,2]
[1]Alibaba-NTU Singapore Joint Research Institute & LILY Research Centre, Nanyang Technological University, Singapore
[2]School of Computer Science and Engineering, Nanyang Technological University, Singapore
[3]Alibaba Group, China
[4]University of Science and Technology of China, China
[5]Zhejiang University, China
{stephenliu,axsun,ascymiao}@ntu.edu.sg,susen@mail.ustc.edu.cn,chenyi.lcy@alibaba-inc.com
{xiaogong.wgx,piaoxue}@taobao.com,juyong@ustc.edu.cn

ABSTRACT

Side information of items, *e.g.,* images and text description, has shown to be effective in contributing to accurate recommendations. Inspired by the recent success of pre-training models on natural language and images, we propose a pre-training strategy to learn item representations by considering both item side information and their relationships. We relate items by common user activities, *e.g.,* co-purchase, and construct a homogeneous item graph. This graph provides a unified view of item relations and their associated side information in multimodality. We develop a novel sampling algorithm named MCNSampling to select contextual neighbors for each item. The proposed Pre-trained Multimodal Graph Transformer (**PMGT**) learns item representations with two objectives: 1) graph structure reconstruction, and 2) masked node feature reconstruction. Experimental results on real datasets demonstrate that the proposed PMGT model effectively exploits the multimodality side information to achieve better accuracies in downstream tasks including item recommendation and click-through ratio prediction. In addition, we also report a case study of testing PMGT in an online setting with 600 thousand users.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Recommendation Systems, Pre-training Model, Graph Transformer

1 INTRODUCTION

In recent years, a good range of recommendation techniques have been proposed, from classic collaborative filtering techniques [30] to the recent deep learning models [40]. Besides the interactions between users and items, the multimodality side information of items has also been exploited and showed effectiveness in further improving recommendation accuracy [31]. Example side information of items include textual descriptions, images, and videos, as shown in Figure 1(a). Traditional methods exploit item side information by manual feature engineering [19], and then employ factorization machine [29] or gradient boosting machine [4] to predict users' preferences on items. These methods often require domain-specific knowledge, and are time-consuming. Deep learning-based methods leverage the strong representation learning ability of neural networks to exploit item side information, for learning the user and/or item representations. However, existing solutions only consider a specific type of side information of items for the dedicated recommendation applications [11, 20, 21]. The full multimodality side information of items are not fully exploited.

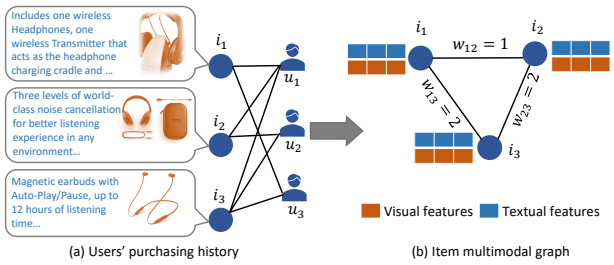


Figure 1: (a) Item side information and users' purchasing history, and (b) Item multimodal graph built on co-purchase relationship. In this graph, each node denotes an item with its visual and textual features extracted from the image and text description respectively. An edge between two items is weighed by the number of co-purchases.

Inspired by the successes of unsupervised pre-training strategies designed for natural language processing [7] and graph data [14, 15, 22, 39], we propose to develop an unsupervised pre-training framework to fully exploit the multimodality side information for item representations learning. Illustrated in Figure 1, we construct an *item multimodal graph* to provide a unified view of items with their associated multimodality side information. In this item multimodal graph, each node is an item and the edges model their relationships (*e.g.,* co-purchase or co-viewership, depending on the application domain). We then pre-train graph neural network (GNN) on this item multimodal graph to enable the GNN model to capture both item relationships and their multimodality side information.

The unified item multimodal graph well distinguishes our work from previous studies where side information of items and their relationships are studied separately. We argue that these two types of information complement each other in solving recommendation problems. We hence focus on effective pre-training on top of

Corresponding authors: Chenyi Lei, Chunyan Miao.

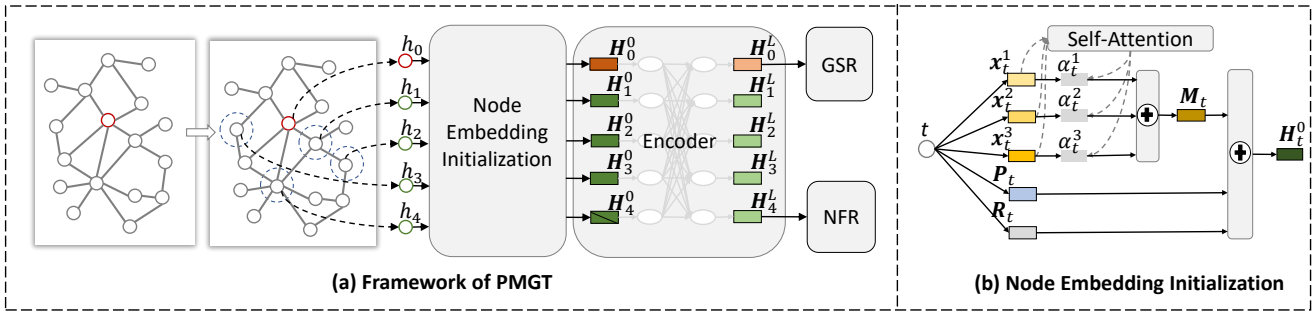


Figure 2: (a) An overview of the proposed PMGT framework. PMGT contains four components (illustrated from the left to the right): contextual neighbors sampling, node embedding initialization, transformer-based graph encoder, and graph reconstruction. GSR and NFR, in the last step, denote the graph structure reconstruction task and the masked node feature reconstruction task, respectively. (b) The node embedding is initialized by considering the node's multimodal features, position-id embedding, and role-label embedding.

this item multimodal graph to benefit item recommendation. Moreover, we also show that our pre-training benefit other E-commerce applications like click-through ratio (CTR) prediction.

The contributions made in this paper are as follows. First, we propose a novel pre-training framework, namely Pre-trained Multimodal Graph Transformer (PMGT), to exploit items' multimodality information through unsupervised learning. To the best of our knowledge, this is the first deep pre-training method developed to exploit the multimodality side information of items for recommender systems. Second, we decompose the learning objective of PMGT into two sub-objectives: (i) graph structure reconstruction, and (ii) masked node feature reconstruction. To handle large-scale graph data, we develop an algorithm, named Mini-batch Contextual Neighbors Sampling (*i.e.,* MCNSampling), for effective and scalable training. Moreover, we employ the attention mechanism to aggregate the item's multimodality information, and a diversity-promoting Transformer framework to model the influences between an item and its contextual neighbors in the graph. Lastly, to demonstrate the effectiveness of the proposed PMGT model, we conduct extensive experiments on real datasets for different applications. The results on two downstream tasks, *i.e.,* item recommendation and CTR prediction, demonstrate that PMGT is more effective than existing graph-based pre-training methods in fully exploiting items' multimodality information. To further show the effectiveness of PMGT, we report a case study of applying this model in an online E-commerce platform.

2 RELATED WORK

Pre-training methods have been widely applied in computer vision (CV) and natural language processing (NLP) tasks [7, 10]. It has been shown that pre-training is effective in boosting performances of various downstream applications. For example, a general pre-training paradigm for CV tasks is firstly training a model on the ImageNet dataset [6], and then fine-tuning the pre-trained model for a specific task. NLP is another domain where pre-training is usually adopted. The shallow pre-training methods, *e.g.,* word2vec [23] and GloVe [25], learn word representations based on the word co-occurrence patterns in a corpus of documents. Recently, significant progress has been made in developing deep pre-training models for contextual word representations. For example, ELMo [27] employs a bidirectional language model to learn high-quality deep context-dependent word representations. The BERT [7] and XLNET [38] models use attention mechanisms to learn the word representations. Significant improvements are achieved when applying these pre-trained models on various NLP tasks.

On graph data, many embedding techniques have been developed in recent years [3]. Representative shallow graph embedding methods include TransE [2], DeepWalk [26], LINE [33], and Node2vec [8]. The recent popularity of GNNs motivates the development of pre-training strategies for GNN models. In general, these methods pre-train GNNs by solving the graph reconstruction problem. For example, Kipf *et al.* introduce the variational graph autoencoder (VGAE) framework for graph reconstruction [18]. Hamilton *et al.* propose a general inductive framework called GraphSAGE [9], which exploits node features to generate node embeddings by sampling and aggregating features from a node's local neighborhood. Velickovic *et al.* propose the Deep Graph Infomax model [35] that aims to maximize the mutual information between the node representations and the representation of the graph. Recently, self-supervised learning [16] is employed to simultaneously pre-train GNNs at both node and graph levels, and an example is the self-supervised learning method for graph neural networks [14]. Similarly, the generative framework GPT-GNN [15] employs a self-supervised attributed graph generation task to pre-train GNNs, by effectively capturing both semantic and structural properties of the graph. In [28], a self-supervised graph neural network pre-training model is proposed to capture the universal network topological properties across multiple networks. In [39], the proposed Graph-BERT model employs attention mechanism to aggregate the neighborhood information of a target node in the graph.

3 THE PROPOSED PRE-TRAINING MODEL

Figure 2 shows the proposed PMGT framework. Observe that PMGT contains four main components: 1) contextual neighbors sampling, 2) node embedding initialization, 3) transformer-based encoder, and 4) graph reconstruction. Before we detail each component in this section, we provide the preliminary background.

In this work, we construct a homogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to provide a uniform view of items' multimodality side information and their relationships. Here, $\mathcal{V}$ denotes the set of nodes (*i.e.,* items), and $\mathcal{E}$ denotes the set of edges between them.[1] Each node h has multiple types of side information. We denote the i-th modality feature of the node h by $\mathbf{x}_h^i$, and the number of modality by m.

In $\mathcal{G}$, we denote the one-hop neighbors of h by $\mathcal{N}_h$, and use ω_{ht} to denote the weight of the edge between two nodes h and t, where $\omega_{ht} > 0$. For a node h, we use C_h to denote its contextual neighbors selected by a sampling algorithm, *e.g.,* the MCNSampling algorithm. Given the item graph $\mathcal{G}$ and the contextual neighbors of each node, PMGT aims to obtain the node representations that can capture the multimodality information of nodes and the graph structure. Then, the learned node representations can be applied in downstream tasks directly or with adjustments such as fine-tuning.

3.1 Contextual Neighbors Sampling

For each node h, there exist some relevant nodes in the graph that may help enrich its representation. These relevant nodes are referred as the *contextual neighbors* of h. To efficiently select contextual neighbors for a batch of nodes during the training of PMGT, we develop a sampling algorithm named MCNSampling. MCNSampling iteratively samples a list of nodes for a target node h with a predefined sampling depth K. Let $\mathcal{S}_h^{k-1}$ denote the bag of nodes sampled at the $(k-1)$-th step. For each node t in $\mathcal{S}_h^{k-1}$, we randomly sample n_k nodes with replacement from t's one-hop neighbors $\mathcal{N}_t$ at the k-th step. The probability that a node $t' \in \mathcal{N}_t$ being sampled is proportional to the weight $\omega_{tt'}$ of the edge between nodes t and t'. Note that a node may appear multiple times in $\mathcal{S}_h^{k-1}$. In the MCNSampling algorithm, we treat all node instances in $\mathcal{S}_h^{k-1}$ as "*different nodes*" and perform the sampling procedure.

In our sampling algorithm, we select contextual neighbors by considering 1) the sampled frequency of a node, and 2) the number of sampling steps between the target node h and a sampled node in the sampling process. For every node $t \in \mathcal{V} \setminus h$, we empirically define its importance to the target node h at the k-th sampling step ($k \le K$) as follows,

$$s_t^k = f_t^k \times (K - k + 1), \tag{1}$$

where f_t^k denotes the number of times t appearing in $\mathcal{S}_h^k$. That is, a node t is considered more relevant to the target node h, if t is sampled more frequently and it has a smaller sampling steps to h. The final importance score of a node t to h is defined as follows,

$$s_t = \sum_{k=1}^{K} s_t^k. \tag{2}$$

Then, we sort all nodes in $\mathcal{V} \setminus h$ according to their importance scores in descending order, and choose the S top-ranked nodes as the sampled contextual neighbors of h. The details of the MCNSamping algorithm are summarized in Algorithm 1.

[1] In the context of recommendation, the relationships between items can be defined by their interactions with users, *e.g.,* co-purchase or co-click. More details about the construction of item graph are presented in Section 4.1.1.

3.2 Node Embedding Initialization

After the neighborhood sampling, we concatenate the target node h and its ordered contextual neighbors C_h, denoted by $\mathcal{I}_h = [h, h_1, h_2, \cdots, h_S]$. h_j is the j-th node in C_h, and $1 \le j \le S$. For each node $t \in \mathcal{I}_h$, we apply the attention mechanism to obtain its multimodal representation $\mathbf{M}_t$ as follows,

$$\begin{aligned} \mathbf{X}_t^i &= \mathbf{x}_t^i \mathbf{W}_M^i + \mathbf{b}_M^i, 1 \le i \le m \\ \mathbf{X}_t &= \mathbf{X}_t^1 \oplus \mathbf{X}_t^2 \oplus \cdots \oplus \mathbf{X}_t^m, \\ \alpha_t = \text{softmax}\big[\tanh(\mathbf{X}_t)\mathbf{W}_s + \mathbf{b}_s\big],\ \mathbf{M}_t &= \sum_i^m \alpha_t^i \mathbf{X}_t^i, \end{aligned} \tag{3}$$

where $\mathbf{W}_M^i \in \mathbb{R}^{d_i \times d_0}$ and $\mathbf{b}_M^i \in \mathbb{R}^{1 \times d_0}$ denote weight matrix and bias term for the i-th modality, $\mathbf{W}_s \in \mathbb{R}^{(md_0) \times m}$ and $\mathbf{b}_s \in \mathbb{R}^{1 \times m}$ denote weight matrix and bias term for attention mechanism. $\oplus$ is the concatenation operation. That is, the multimodality side information of each item is concatenated to contribute to the comprehensive representation learning. α_t^i denotes the i-th element of α_t.

The position of a node in the input list $\mathcal{I}_h$ reflects its importance to the target node h. Thus, we argue that the order of nodes in $\mathcal{I}_h$ is important in learning node representations. The position-id embedding is used to identify the node order information of an input list,

$$\mathbf{P}_t = \textbf{P-Embedding}\big[p(t)\big], \tag{4}$$

where $p(t)$ denotes the position id of node t in $\mathcal{I}_h$, $\mathbf{P}_t \in \mathbb{R}^{1 \times d_0}$ denotes the position-based embedding for t. Our main objective is to obtain the representation of the target node h. Intuitively, the target node and its contextual neighbors should play different roles in the pre-training. To identify the role differences, we add the following role-based embedding to each node $t \in \mathcal{I}_h$,

$$\mathbf{R}_t = \textbf{R-Embedding}\big[r(t)\big], \tag{5}$$

where $r(t)$ and $\mathbf{R}_t \in \mathbb{R}^{1 \times d_0}$ denote the role label and role-based embedding of the node t, respectively. In practice, we set the role label of the target node as "`Target`" and the role labels of the contextual neighbors as "`Context`". Based on the embeddings stated above, we aggregate them together to define the initial input embedding for a node $t \in \mathcal{I}_h$ as,

$$\mathbf{H}_t^0 = \textbf{Aggregate}\big(\mathbf{M}_t, \mathbf{P}_t, \mathbf{R}_t\big). \tag{6}$$

In this work, we simply define the **Aggregate**$(\cdot)$ function as vector summation. The initial input embeddings for the nodes in the input list $\mathcal{I}_h$ can be stacked into a matrix $\mathbf{H}^0 = [\mathbf{H}_0^0; \mathbf{H}_1^0; \cdots; \mathbf{H}_S^0] \in \mathbb{R}^{(S+1) \times d_0}$, where $\mathbf{H}_0^0$ corresponds to the target node h.

3.3 Transformer-based Graph Encoder

We use the Transformer framework [34] to model the mutual influences between a node and its contextual neighbors. Given the node representations $\mathbf{H}^{\ell-1}$ at the $(\ell-1)$-th layer, the output at the ℓ-th layer of original Transformer model is defined as follows,

$$\begin{aligned} \mathbf{H}^\ell &= \text{FFN}\left[\text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_h}}\right)\mathbf{V}\right], \\ \mathbf{Q} = \mathbf{H}^{\ell-1}\mathbf{W}_Q^\ell, \mathbf{K} &= \mathbf{H}^{\ell-1}\mathbf{W}_K^\ell, \mathbf{V} = \mathbf{H}^{\ell-1}\mathbf{W}_V^\ell, \end{aligned} \tag{7}$$

Algorithm 1 MCNSampling Algorithm

Input: Graph $\mathcal{G}$, batch of nodes $\mathcal{B}$, sampling depth K, sampling size $\{n_k\}_{k=1}^{K}$, number of contextual neighbors S;
Output: Sampled contextual neighbors C_B;

```
C_B ← [ ];
for h ∈ B do
    S_0 ← [h], S_1, S_2, ··· S_K ← [ ], s_t ← 0 ∀t ∈ V \ h;
    for k = 1, 2, ··· , K do
        for t ∈ S_{k−1} do
            Sample n_k nodes with replacement from N_t and append them to S_k;
        end for
        Count the frequency f_t^k of each distinct node t in S_k and update its score s_t ← s_t + f_t^k * (K − k + 1);
    end for
    Sort the nodes in V \ h according to the importance scores in descending order;
    Choose S top-ranked nodes as the contextual neighbors C_h of h and append it to C_B;
end for
return C_B
```

where $\mathbf{W}_Q^\ell, \mathbf{W}_K^\ell, \mathbf{W}_V^\ell \in \mathbb{R}^{d_0 \times d_0}$ denote the weight matrices, FFN$(\cdot)$ is the feed forward network. Here, we omit the residual network in the formula for convenience.

For the target node h, there may exist some sampled nodes in C_h, whose representations are similar to the representation of h. Assume that all the sampled contextual neighbors are relevant to h. We hope the proposed model can capture the diversity of the sampled contextual neighbors, by concentrating on the nodes that are relevant but not very similar to the target node. To achieve this objective, we design a diversity-promoting attention mechanism and include it into the attention network of Transformer,

$$\begin{aligned} \mathbf{S} &= \mathbf{H}^{\ell-1}\mathbf{W}_S^\ell, \\ \mathbf{U}_1 &= \text{softmax}\left(\mathbf{E} - \frac{\mathbf{S}\mathbf{S}^\top}{||\mathbf{S}||_2||\mathbf{S}||_2^\top} + \mathbf{I}\right), \\ \mathbf{U}_2 &= \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_h}}), \\ \mathbf{H}^\ell &= \text{FFN}\Big[(\beta\mathbf{U}_1 + (1-\beta)\mathbf{U}_2)\mathbf{V}\Big], \end{aligned} \tag{8,9}$$

where $\mathbf{W}_S^l \in \mathbb{R}^{d_0 \times d_0}$ is the weight matrix, $\mathbf{E} \in \mathbb{R}^{(S+1)\times(S+1)}$ is a matrix where all its elements are 1, $||\mathbf{S}||_2 \in \mathbb{R}^{(S+1)\times 1}$ denotes the ℓ_2 row norm of $\mathbf{S}$, and $\mathbf{I} \in \mathbb{R}^{(S+1)\times(S+1)}$ denotes the identity matrix. In Eq. (8), "—" denotes the element-wise division of two matrices. Note that the larger the similarity between two different nodes, the smaller the attention weight between them in $\mathbf{U}_1$. The objective of adding $\mathbf{I}$ in the definition of $\mathbf{U}_1$ is to include the node's self information. β is a constant ($0 \le \beta \le 1$) balancing the contributions of the two attention weights. After obtaining the output $\mathbf{H}^L$ at the last layer of the encoder, we obtain $\mathbf{H}_0^L$ as the representation of target node h, denoted as $\mathbf{h}$ for simplicity. Then, $\mathbf{H}^L$ will be used in the following pre-training tasks.

3.4 Model Optimization

PMGT is pre-trained with the following two objectives: 1) graph structure reconstruction, and 2) masked node feature reconstruction. To ensure the learned node representations can capture the graph structure, we define the following loss function [9],

$$\begin{aligned} \mathcal{L}_{edge} = &\frac{1}{|\mathcal{V}|}\sum_{h\in\mathcal{V}}\frac{1}{|\mathcal{N}_h|}\sum_{t\in\mathcal{N}_h}\Bigg[-\log\left(\sigma(\frac{\mathbf{h}^\top\mathbf{t}}{||\mathbf{h}||_2||\mathbf{t}||_2})\right) \\ &- Q\cdot\mathbb{E}_{t_n\sim P_n(t)}\log\left(\sigma(-\frac{\mathbf{h}^\top\mathbf{t}_n}{||\mathbf{h}||_2||\mathbf{t}_n||_2})\right)\Bigg], \end{aligned} \tag{10}$$

where $\sigma(\cdot)$ is the sigmoid function, P_n and Q denote the negative sampling distribution and the number of negative samples.

The node feature reconstruction task focuses on capturing the multimodal features in the learned node representations. Previous methods, *e.g.,* GRAPH-BERT [39], design an attribute reconstruction task without masking operations. Thus, the models' abilities in aggregating the features of different nodes may be limited. In this work, we design a masked node feature reconstruction task, which aims to reconstruct the features of masked nodes by other non-masked nodes in $\mathcal{I}_h$. As the representation of the target node h is needed to reconstruct the graph structure in Eq. (10), we do not apply the masking operation to the target node h. Following [7], we randomly choose 20% of nodes in the list $\mathcal{I}_h \backslash h$ for masking. If the node t is chosen, we replace t with: 1) the [Mask] node 80% of the time, (2) a random node 10% of the time, and (3) the unchanged node t 10% of the time. Then, the masked item list will be input to the model, and the output $\mathbf{H}^L$ will be used to reconstruct the multimodal features of the masked nodes. We set the input features of the [Mask] node to $\mathbf{0}$, and define the feature reconstruction loss as follows,

$$\mathcal{L}_{feature} = \frac{1}{|\mathcal{V}|}\sum_{h\in\mathcal{V}}\frac{1}{|\mathcal{M}_h|}\sum_{t\in\mathcal{M}_h}\sum_{i}^{m}\left\|\mathbf{H}_t^L\mathbf{W}_r^i - \mathbf{x}_t^i\right\|_2^2, \tag{11}$$

where $\mathcal{M}_h$ denotes the set of masked nodes in $\mathcal{I}_h$, $\mathbf{H}_t^L$ denotes the representation of t in $\mathbf{H}^L$, and $\mathbf{W}_r^i$ is the weight matrix for the i-th modality information reconstruction.

The model parameters of PMGT can be learned by minimizing the combined objective function,

$$\mathcal{L}_{edge} + \lambda\mathcal{L}_{feature}. \tag{12}$$

The entire framework can be effectively trained by the end-to-end backpropagation algorithm. To make the training of the model more stable, a mini-batch of nodes are randomly sampled to update the model. When applying the pre-trained PMGT model in downstream tasks, the learned node representations can be either fed into the new tasks directly or with necessary adjustment (*e.g.,* fine-tuning).

4 EXPERIMENTS

4.1 Experimental Settings

4.1.1 Experimental Datasets. The experiments are performed on the Amazon review dataset [24] and Movielens-20M dataset[2].

Amazon Datasets. We choose the following 5-score review subsets of the Amazon review dataset for experiments, *i.e.,* "Video Games", "Toys and Games", and "Tools and Home Improvement"

[2]https://grouplens.org/datasets/movielens/20m/

(respectively denoted by VG, TG, and THI). The metadata of a product includes its text description and the URL of its image[3], which are used to extract the textual and visual features of the product respectively. In the experiments, we use the rating data generated before 2015-01-01 for building the product graph, and the rating data generated since 2015-01-01 for studying the performance of the item recommendation and CTR prediction tasks. In these two downstream tasks, we convert all the observed review ratings to be positive interactions and filter out the products that are not included in the product graph.

Moreover, we build the product graph based on users' review behaviors. Let r_{ht} denotes the number of users who have commonly reviewed the two products h and t. If $r_{ht} \geq 3$, we connect the two products h and t in the graph. In the graph, there inevitably exist some popular nodes with large degrees. This usually causes that the popular nodes with large degrees are more likely to be sampled in the contextual neighbors sampling procedure. To alleviate this problem, we define the edge weight based on the vertex degrees of an edge. Empirically, we define the weight ω_{ht} of the edge e_{ht} between the nodes h and t as follows,

$$\omega_{ht} = \frac{\log(r_{ht}) + 1}{\log(\sqrt{d_h * d_t}) + 1}, \tag{13}$$

where d_h and d_t denote the degrees of the nodes (*i.e.,* products) h and t in the graph, respectively. The operation of $\log(\cdot)$ alleviates the problem of large variance of edge weights in the product graph. By inversely proportional to the degrees of nodes h ad t, we reduce the weights of popular nodes.

Movielens-20M Dataset. For the Movielens-20M dataset (denoted by ML), we construct the movie graph based on the tags of movies. The tags of a movie are obtained from the MovieLens Tag Genome Dataset[4], which includes 11 million computed tag-movie relevance scores from a pool of 1,100 tags applied to 10,000 movies. For each movie, we only keep the tags with relevance scores larger than 0.9. Then, we use r_{ht} to denote the number of common tags two movies h and t have. If $r_{ht} \geq 3$, we construct an edge between the two movies h and t, and the weight ω_{ht} of the edge e_{ht} between two nodes h and t is defined following Eq. (13). We collect the movie trailers from Youtube and extract keyframes for each movie trailer. These keyframes are used as the movie images for extracting the visual modality of a movie. The movie descriptions are collected from TMDB[5]. In addition, the rating data generated since 2008-01-01 are used to evaluate the performance of item recommendation and CTR prediction tasks, where we keep ratings larger than 3 as positive interactions.

Multimodal Feature Extraction. In the experiments, we use the pre-trained Inception-v4 network [32] to extract the visual features of each image. Then, we average the visual features of all the images of an item (*i.e.,* product or movie) to obtain its visual modality. For the text description of an item, we utilize the pre-trained BERT model [7] to extract the features of each sentence. Then, we average the features of all the sentences in the description of an item to obtain its textual modality. Empirically, we set the length of the sentence to 128. For the ML dataset, we separate the audio track from the movie trailer with FFmpeg[6] and adopt VGGish [13] to obtain the acoustic modality of the movie. The dimensionality of the visual, textual, and acoustic modalities are 1,536, 768, and 128, respectively. Table 1 summarizes the statistics of these experimental datasets used for item recommendation and CTR prediction tasks.

[3]https://nijianmo.github.io/amazon/index.html
[4]https://grouplens.org/datasets/movielens/tag-genome/
[5]https://www.themoviedb.org/

Table 1: Statistics of the experimental datasets.

Datasets	Data for Downstream Tasks			Item Graph	
	# Users	# Items	# Interact.	# Nodes	# Edges
VG	4,525	3,921	27,780	5,032	83,981
TG	31,109	13,870	182,744	17,388	232,720
THI	20,082	11,170	109,717	15,619	178,834
ML	27,715	4,253	2,179,386	4,271	249,498

4.1.2 Setup and Metrics. After the data pre-processing in Section 4.1.1, an item multimodal graph $\mathcal{G}$ is built, and a set of user-item interactions $\mathcal{D}_{down}$ is prepared for studying downstream tasks.

In the pre-training task, we randomly keep 90% of nodes and their relationships in $\mathcal{G}$ to pre-train the item representations. The remaining 10% of nodes (denoted by $\mathcal{V}_{val}$) are used to construct the validation data for choosing the hyper-parameters of different pre-training models. For each node t in $\mathcal{V}_{val}$, we randomly sample a node t_+ from its one-hop neighbors in $\mathcal{G}$, and use the pair (t, t_+) as a positive example in validation data. And we also randomly sample a node t_- that is not connected with t in $\mathcal{G}$, and use the pair (t, t_-) as a negative example in validation data. The model parameters are chosen based on the AUC (*i.e.,* Area under the ROC Curve) computed on all the validation data.

For downstream tasks, we choose Neural Collaborative Filtering (NCF) [12] as the base model for item recommendation task, and Deep & Cross Network (DCN) [36] as the base model for CTR prediction task. For item recommendation task, we randomly select 80% of the interactions in $\mathcal{D}_{down}$ as training data to update the NCF model, and the remaining 20% of interactions in $\mathcal{D}_{down}$ are used for testing. Moreover, we also randomly hold out 10% of the training data for tuning the hyper-parameters of NCF. In the training of NCF, for each positive interaction pair, we randomly sample one item that has no interactions with the user as negative feedback to update the model. The item recommendation performances achieved by NCF are evaluated by Recall@10, Recall@20, NDCG@10, and NDCG@20 (respectively denote by REC-R@10, REC-R@20, REC-N@10, and REC-N@20). To improve the evaluation efficiency, we randomly sample 1000 items that the testing user has not interacted with to compute the recall and NDCG. For CTR prediction task, we use $\mathcal{D}_{down}$ to simulate the CTR data. For each possible user-item pair in $\mathcal{D}_{down}$, we randomly sample 5 items that have no interactions with the user to construct the negative samples of the CTR data. All the interaction pairs in $\mathcal{D}_{down}$ are used as the positive examples of the CTR data. Then, we randomly sample 80% of the CTR data to train the DCN model, and use the remaining 20% of CTR data to testing the CTR prediction performances. Moreover, 10% of the training data are also held out for tuning the hyper-parameters of DCN. The CTR prediction performances are evaluated by AUC (denoted by CTR-AUC).

[6]http://ffmpeg.org/

Table 2: Performances of item recommendation (REC) and CTR prediction by using different pre-training methods. Best results are in boldface and second best underlined.

Datasets	Metrics	Random	DeepWalk	LINE	TransAE	GraphSAGE	GRAPH-BERT	GPT-GNN	PMGT
VG	REC-R@10	0.1994	0.2236	0.2234	0.1989	0.2052	0.2380	0.2251	**0.2480**
	REC-N@10	0.1278	0.1441	0.1420	0.1217	0.1301	0.1491	0.1343	**0.1625**
	REC-R@20	0.2742	0.3179	0.3169	0.2903	0.2821	0.3330	0.3269	**0.3405**
	REC-N@20	0.1494	0.1711	0.1690	0.1480	0.1522	0.1767	0.1636	**0.1890**
	CTR-AUC	0.7311	0.768	0.7762	0.7675	0.7674	0.7746	0.7839	**0.7990**
TG	REC-R@10	0.2147	0.2787	0.2805	0.2137	0.2391	0.2858	0.2598	**0.3032**
	REC-N@10	0.1380	0.1847	0.1857	0.1358	0.1514	0.1942	0.1671	**0.2056**
	REC-R@20	0.3068	0.3807	0.3873	0.3051	0.3352	0.3808	0.3608	**0.4030**
	REC-N@20	0.1644	0.2141	0.2162	0.1620	0.1790	0.2215	0.1962	**0.2342**
	CTR-AUC	0.8047	0.8289	0.8326	0.8214	0.8266	0.8322	0.8328	**0.8370**
THI	REC-R@10	0.1957	0.2043	0.1626	0.1425	0.1921	0.2023	0.1680	**0.2358**
	REC-N@10	0.1360	0.1399	0.0996	0.0861	0.1320	0.1462	0.1047	**0.1707**
	REC-R@20	0.2555	0.2756	0.2403	0.2191	0.2577	0.2627	0.2451	**0.3025**
	REC-N@20	0.1529	0.1598	0.1214	0.1077	0.1505	0.1632	0.1264	**0.1895**
	CTR-AUC	0.7652	0.7850	0.7896	0.7815	0.7643	0.7878	0.7817	**0.7933**
ML	REC-R@10	0.3104	0.3144	0.3112	0.3096	0.3139	0.3136	**0.3162**	0.3161
	REC-N@10	0.4714	0.4793	0.4763	0.4703	0.4769	0.4781	**0.4823**	0.4814
	REC-R@20	0.4556	0.4618	0.4589	0.4549	0.4606	0.4581	0.4628	**0.4640**
	REC-N@20	0.4829	0.4911	0.4881	0.4822	0.4888	0.4883	**0.4928**	**0.4928**
	CTR-AUC	0.9109	**0.9218**	0.9200	0.9194	0.9145	0.9184	0.9191	0.9205

4.1.3 Baseline Methods. We compare the proposed PMGT model with the following pre-training methods: 1) **Random**: The item embeddings in the downstream tasks are randomly initialized; 2) **DeepWalk** [26]: This method learns node representations by sampling a large number of paths in the graph and maximizing the average logarithmic probability of all vertex context pairs in sampled paths; 3) **LINE** [33]: This graph embedding method is trained to preserve the first- and second-order proximities of nodes in the graph; 4) **GraphSAGE** [9]: This GNN model forces connected nodes to have similar embeddings by aggregating the information of neighboring nodes; 5) **TransAE** [37]: This method combines a multimodal encoder and the TransE [2] model to learn the node representations; 6) **GRAPH-BERT** [39]: This method applies Transformer to aggregate neighbors' information without masking operations on the nodes; 7) **GPT-GNN** [15]: This method employs the attribute generation and edge generation tasks to pre-train the GNN model. For a fair comparison, we use the same multimodal representation in Eq. (3) as the inputs for all pre-training methods, except Random.

4.1.4 Implementation Details. For the pre-training and downstream tasks, we set the dimensionality of latent space d_0 to 128. In the experiments, we empirically set the sampling depth K to 3, and the sampling sizes n_1, n_2, n_3 to 16, 8, 4 respectively. The number of contextual neighbors S is selected from {5, 10, 20, 30, 40}. The number of transformer layers L is chosen from {1, 2, 3, 4, 5}. The weight of diversity-promoting attention β is selected from {0, 0.2, 0.5, 0.8, 1.0}. λ is empirically set to 1. We implement PMGT based on TensorFlow [1] and AliGraph [41] frameworks. Adam [17] is used as the optimizer for learning model parameters, and the learning rate is chosen from $\{10^{-4}, 10^{-3}, 10^{-2}\}$.

4.2 Performance Comparison

After pre-training on the item graph, we use the pre-trained item representations to initialize the item embeddings in the downstream tasks. Then, we train the NCF and DCN models and fine-tune the item embeddings based on the user-item interaction data. Table 2 summarizes the performances of NCF and DCN initialized with item representations pre-trained by different methods. We make the following observations. Compared with the random initialization, initializing the base models with pre-trained item representations usually achieves better item recommendation and CTR prediction performances. This demonstrates the pre-training strategies can benefit downstream tasks in recommendation scenarios. The deep pre-training methods GRAPH-BERT, GPT-GNN, and PMGT usually outperform other shallow pre-training methods, by employing GNN to aggregate the neighbor information, and using node feature reconstruction and graph structure reconstruction tasks to pre-train the model. Moreover, PMGT usually achieves the best item recommendation and CTR prediction performances on all datasets. This demonstrates the effectiveness of PMGT in exploiting the item graph structure and item features. In addition, we can also note that PMGT achieves smaller improvements on the ML dataset. One potential reason is that the interaction data of the ML dataset are denser. Thus, with the random initialization, the base models can learn sufficiently good item representations for downstream tasks.

4.3 Ablation Study

We also study the effectiveness of PMGT in exploiting different modality information. As shown in Figure 3, we can note that the original methods considering multimodality information usually outperform the variants that only consider single modality information. This observation is as expected. It indicates that representing items with multimodality information can achieve better performance. PMGT is superior to GRAPH-BERT and GPT-GNN with considering single modality information in most scenarios. This observation again demonstrates that PMGT is more effective in capturing different types of modality information than baseline

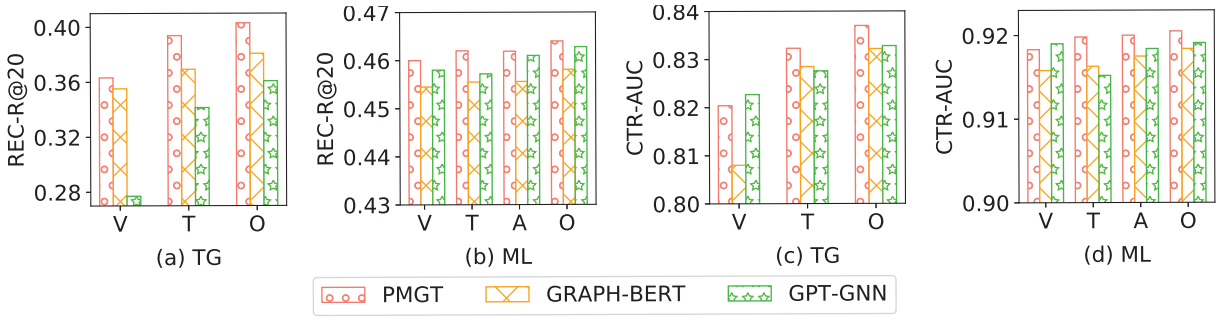


Figure 3: Performances of PMGT, GRAPH-BERT and GPT-GNN considering different modality information on TG and ML datasets. V, T, A denotes the visual, textual, and acoustic modality information respectively. O denotes original models considering all the modality information.

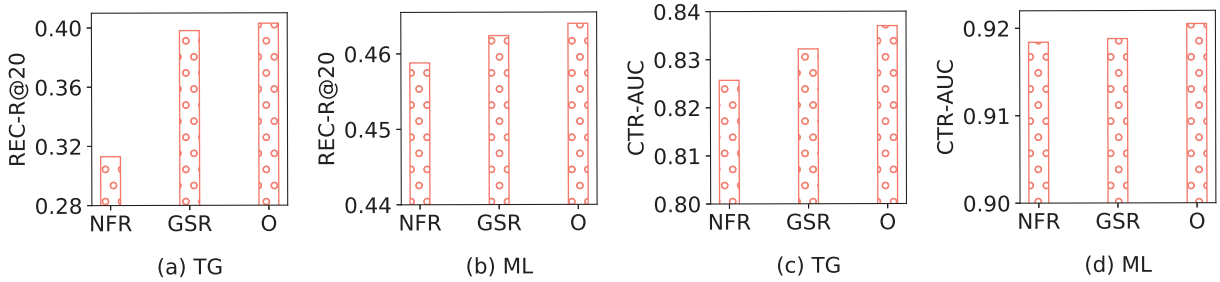


Figure 4: Performances of PMGT considering different graph reconstruction tasks on TG and ML datasets. NFR and GSR denote PMGT only considering the node feature reconstruction task and graph structure reconstruction task, respectively. O denotes the original PMGT considering both tasks.

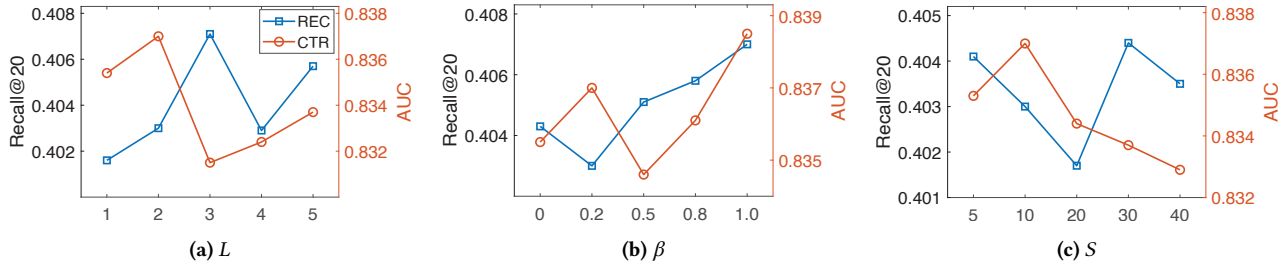
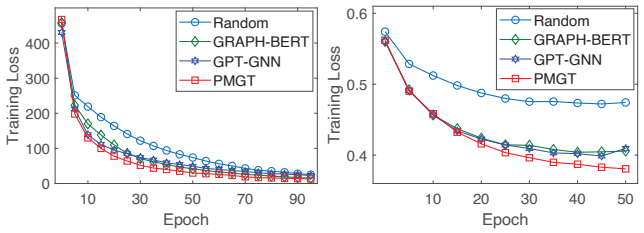


Figure 5: The performance trend of PMGT with respect to different settings of L, β, and S on TG dataset.

methods. Moreover, we also study the effectiveness of the two graph reconstruction tasks in learning node representations. Figure 4 summarizes the performances of PMGT variants on the TG and ML datasets. We can note that the original PMGT model with two tasks consistently outperforms the variants using a single task as the pre-training objective. This indicates that both the graph structure reconstruction task and the masked node feature reconstruction task are essential for learning useful node representations to benefit downstream tasks.

4.4 Parameter Sensitivity Study

In this section, we study the performances of PMGT with respect to (w.r.t.) different settings of three important hyper-parameters. Firstly, we vary the number of Transformer layers L from 1 to 5. As shown in Figure 5a, the best item recommendation and CTR prediction performances are achieved by setting L to 3 and 2, respectively. Further stacking more layers does not help improve the performances of downstream tasks. Moreover, we vary the weight of the diversity-promoting attention score β in {0, 0.2, 0.5, 0.8, 1.0}. As shown in Figure 5b, the recommendation accuracy can be improved by considering diversity-promoting attention in the Transformer-based encoder, when β is set to 0.5, 0.8, and 1.0. This indicates it is important to consider the diversity of contextual neighbors when learning the node representations. Moreover, we also note that the best performances of both downstream tasks are achieved by setting β to 1.0. However, more experiments are needed to study whether this is a general observation on different datasets. In the current implementation, we keep β to enable the flexibility of PMGT. Figure 5c summarizes the performances of PMGT w.r.t. different settings of the number of contextual neighbors S. We observe that PMGT usually achieves good performances by setting S



(a) NCF (b) DCN

Figure 6: The convergence speed of the training loss of (a) NCF model and (b) DCN model on the TG dataset.

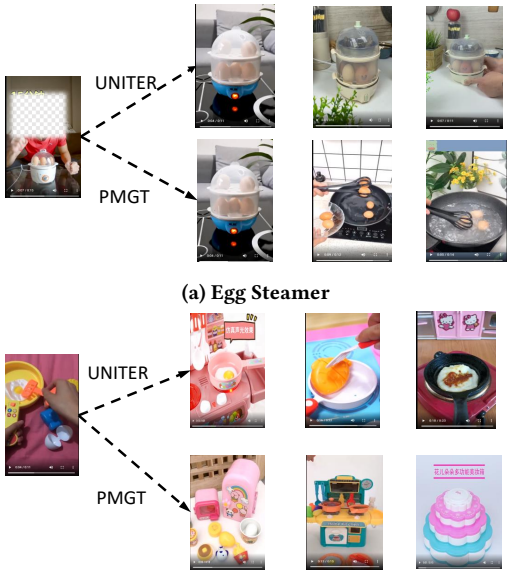
to a small value (*e.g.,* 5 and 10). This indicates that a small number of contextual neighbors can capture the important neighborhood information of a node. Further increase of S tends to include noise information, thus may not help improve the model performances.

4.5 Convergence Speed Study

Figure 6 shows the convergence speed of the training losses of both NCF and DCN models on the TG dataset. We initialize the item embeddings in the NCF and DCN models by the following strategies: 1) random initialization, 2) initializing using the representations pre-trained by GRAPH-BERT, 3) initializing using the representations pre-trained by GPT-GNN, and 4) initializing using the representations pre-trained by PMGT. We can make the following observations. Compared with the random initialization, initializing item embeddings with the pre-trained representations achieves faster convergence speed. This once again demonstrates the effectiveness of the pre-training strategies. Moreover, both NCF and DCN models achieve the fastest convergence speed by using pre-trained representations by PMGT. The pre-trained PMGT model gives the downstream model a good initial state, thus can help the downstream model achieves faster and earlier convergence.

5 CASE STUDY IN ONLINE PLATFORM

A case study is conducted in the video recommendation scenario of one of the world's largest E-commerce platforms. The video graph is built based on users' watching behaviors. Let r_{ht} denote the number of users who have watched the videos h and t within one hour. If $r_{ht} \geq 10$, we build an edge between the nodes h and t in the video graph. The weight of the edge e_{ht} between h and t is defined following Eq. (13). Finally, there are about 4 million nodes and 500 millions of edges in the video graph used for this case study. Given the pre-trained video representations by PMGT, for a user, we retrieve 50 most similar videos for each video she has watched, based on the Cosine similarity between the video representations. Then, ItemKNN is used to rank and recommend the retrieved videos to the user. After three days of online testing, for 600 thousand users, the number of new video plays increases by 6.80%, compared with the online baseline method using the video representation learned by UNITER [5]. Figure 7 shows two video retrieval examples. We can note that the videos retrieved based on the representations pre-trained by PMGT are more diverse than those retrieved based on the representations pre-trained by UNITER.



(a) Egg Steamer

(b) Toy of Pot

Figure 7: Examples of video retrieval based on the video representations pre-trained by PMGT and UNITER.

6 CONCLUSION AND FUTURE WORK

This paper proposes a novel pre-training GNN framework, named PMGT (*i.e.,* Pre-trained Multimodal Graph Transformer), which exploits items' multimodal information guided by the unsupervised learning tasks on graph. Two graph reconstruction tasks, *i.e.,* graph structure reconstruction and masked node feature reconstruction, are used as learning objectives to pre-train the model. The learned representation of an item not only integrates the multimodal information of the item itself but also aggregates the information of its contextual neighbors in the graph. The superiority of PMGT has been validated by two downstream tasks (*i.e.,* item recommendation and CTR prediction) on real datasets. In this work, we focus on the homogeneous graph of items. For future work, we would like to investigate how to extend the proposed model to process the heterogeneous item graph.

ACKNOWLEDGMENTS

This research is supported, in part, by Alibaba Group through Alibaba Innovative Research (AIR) Program and Alibaba-NTU Singapore Joint Research Institute (JRI), Nanyang Technological University, Singapore. This research is also supported, in part, by the National Research Foundation, Prime Minister's Office, Singapore under its AI Singapore Programme (AISG Award No: AISG-GC-2019-003) and under its NRF Investigatorship Programme (NRFI Award No. NRF-NRFI05-2019-0002). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of National Research Foundation, Singapore.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al.

2016. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation*. 265–283.

[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*. 2787–2795.

[3] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. 2018. A comprehensive survey of graph embedding: problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering* 30, 9 (2018), 1616–1637.

[4] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 785–794.

[5] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. Uniter: Universal image-text representation learning. In *European Conference on Computer Vision*. Springer, 104–120.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on Computer Vision and Pattern Recognition*. IEEE, 248–255.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[8] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 855–864.

[9] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.

[10] Kaiming He, Ross Girshick, and Piotr Dollár. 2019. Rethinking imagenet pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4918–4927.

[11] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*. 507–517.

[12] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. 173–182.

[13] Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. 2017. CNN architectures for large-scale audio classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 131–135.

[14] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2020. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*.

[15] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. 2020. Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1857–1867.

[16] Longlong Jing and Yingli Tian. 2020. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[17] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[18] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[19] Guimei Liu, Tam T Nguyen, Gang Zhao, Wei Zha, Jianbo Yang, Jianneng Cao, Min Wu, Peilin Zhao, and Wei Chen. 2016. Repeat buyer prediction for e-commerce. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 155–164.

[20] Yong Liu, Susen Yang, Yonghui Xu, Chunyan Miao, Min Wu, and Juyong Zhang. 2021. Contextualized Graph Attention Network for Recommendation with Item Knowledge Graph. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[21] Yong Liu, Susen Yang, Yinan Zhang, Chunyan Miao, Zaiqing Nie, and Juyong Zhang. 2021. Learning Hierarchical Review Graph Representations for Recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[22] Yuanfu Lu, Xunqiang Jiang, Yuan Fang, and Chuan Shi. 2021. Learning to pre-train graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4276–4284.

[23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*. 3111–3119.

[24] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. 188–197.

[25] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. 1532–1543.

[26] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining*. 701–710.

[27] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* (2018).

[28] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1150–1160.

[29] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining*. IEEE, 995–1000.

[30] Yue Shi, Martha Larson, and Alan Hanjalic. 2014. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *Comput. Surveys* 47, 1 (2014), 1–45.

[31] Zhu Sun, Qing Guo, Jie Yang, Hui Fang, Guibing Guo, Jie Zhang, and Robin Burke. 2019. Research commentary on recommendations with side information: A survey and research directions. *Electronic Commerce Research and Applications* 37 (2019), 100879.

[32] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. 2017. Inception-v4, inception-ResNet and the impact of residual connections on learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. 4278–4284.

[33] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*. 1067–1077.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.

[35] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep graph infomax. In *International Conference on Learning Representations*.

[36] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*. 1–7.

[37] Zikang Wang, Linjing Li, Qiudan Li, and Daniel Zeng. 2019. Multimodal data enhanced representation learning for knowledge graphs. In *2019 International Joint Conference on Neural Networks*. IEEE, 1–8.

[38] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*. 5753–5763.

[39] Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. 2020. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140* (2020).

[40] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *Comput. Surveys* 52, 1 (2019), 1–38.

[41] Rong Zhu, Kun Zhao, Hongxia Yang, Wei Lin, Chang Zhou, Baole Ai, Yong Li, and Jingren Zhou. 2019. AliGraph: a comprehensive graph neural network platform. *Proceedings of the VLDB Endowment* 12, 12 (2019), 2094–2105.