Graph Neural Pre-training for Recommendation with Side Information

SIWEI LIU*, University of Glasgow
ZAIQIAO MENG*, University of Glasgow
CRAIG MACDONALD, University of Glasgow
IADH OUNIS, University of Glasgow

Leveraging the side information associated with entities (i.e. users and items) to enhance recommendation systems has been widely recognized as an essential modeling dimension. Most of the existing approaches address this task by the *integration-based scheme*, which incorporate the entity side information by combining the recommendation objective with an extra side information-aware objective. Despite of the growing progress made by the existing integration-based approaches, they are largely limited by the potential conflicts between the two objectives. Moreover, the heterogeneous side information among entities is still under-explored in these systems. In this paper, we propose a novel *pre-training scheme* to leverage the entity side information by pre-training entity embeddings using the multi-graph neural network. Instead of jointly training with two objectives, our *pre-training scheme* first pre-trains two representation models under the entity multi/single relational graphs constructed by their side information, and then fine-tunes their embeddings under an existing general representation-based recommendation model. Our proposed multi-graph and single-graph neural networks can generate within-entity knowledge-encapsulated embeddings, while capturing the heterogeneity from the entity side information simultaneously, thereby improving the performance of the underlying recommendation model. An extensive evaluation of our pre-training scheme fine-tuned under four general representation-based recommender models, namely, MF, NCF, NGCF and LightGCN, shows that effectively pre-training embeddings with both the user's and item's side information can significantly improve these original models in terms of both effectiveness and stability.

ACM Reference Format:
Siwei Liu, Zaiqiao Meng, Craig Macdonald, and Iadh Ounis. 2022. Graph Neural Pre-training for Recommendation with Side Information. *ACM Transactions on Information Systems* 1, 1, Article 1 (January 2022), 28 pages. https://doi.org/10.1145/3568953

1 INTRODUCTION

The goal of recommender systems is to assist users in filtering out non-relevant information and selecting a personalized set of interesting items to maximize the users' satisfaction. Modern recommendation models achieve this goal by learning representation vectors (i.e. embeddings) of the two entities (i.e. users and items) that capture the users' interests and items' attractiveness [79, 80], so that the learned embeddings can be used to accurately predict which items a user might choose in the future, e.g., by computing the dot product or a multilayer perceptron (MLP) [48] of the users' and items' embeddings. Typically, recommendation models are collaborative in nature, learning the users' interests and items' attractiveness through users' ratings, clicking or other interactive knowledge crossing the two entities, which we refer to as the *cross-entity knowledge*. For example,

* Both authors contributed equally to this work.

Authors' addresses: Siwei Liu, University of Glasgow, s.liu.4@research.gla.ac.uk; Zaiqiao Meng, University of Glasgow, zaiqiao.meng@glasgow.ac.uk; Craig Macdonald, University of Glasgow, craig.macdonald@glasgow.ac.uk; Iadh Ounis, University of Glasgow, iadh.ounis@glasgow.ac.uk.

the NGCF [62] model leverages the high-order connectivity between users and items (i.e. the interaction signal in the user-item interactive graph) to enhance the recommendation performance. However, these cross-entity interactions are typically sparse [47]. Therefore, a number of side information-aware recommendation models [6, 34, 40, 54] have been designed to alleviate the sparsity issue by integrating the rich side information of users and items such as the users' age groups and the items' textual descriptions. Such side information about entities can be used to learn the *within-entity knowledge* to further enhance the recommendation performance. For instance, movies with the same features (e.g. same genres and actors) may attract the same users, and such feature relations between movies are a type of knowledge within the side information of movies.

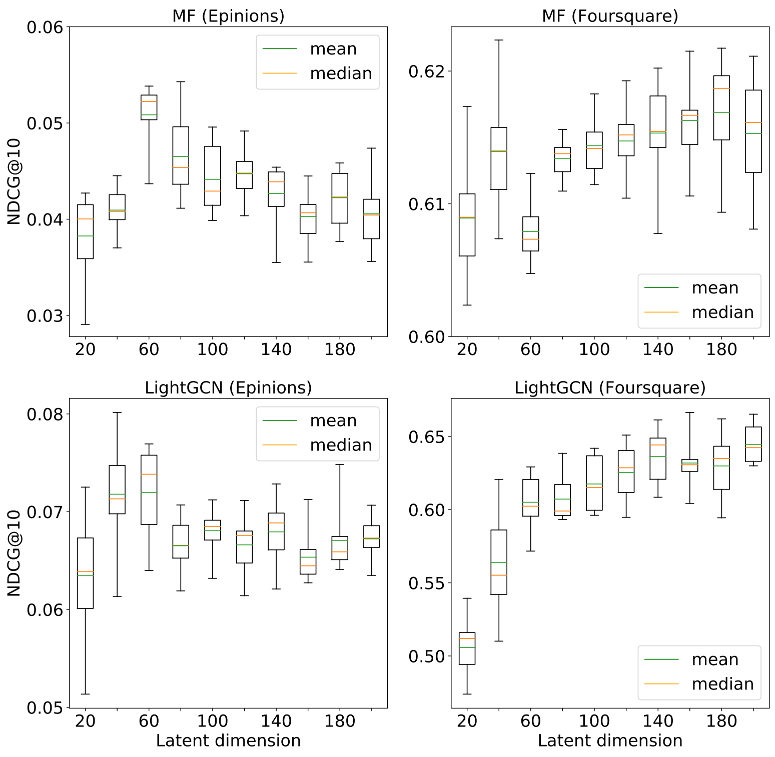

Fig. 1. Box-and-whisker diagrams for the NDCG performances of the MF [48] and LightGCN [14] models on the Epinions and Foursquare datasets.

To leverage the side information associated with users and items, many approaches have been proposed, most of which follow the conventional *integration scheme*, which encodes the side information simultaneously with the training of user-item interactions [6, 40, 43]. These integration-based approaches normally optimize a loss function consisting of two components, i.e. the recommendation loss and one (or even more) additional side information-aware loss(es), where a hyperparameter is usually used to control the importance of each loss component [6, 34, 61, 81]. It is often difficult to find one single adequate solution to optimise all of the loss components since different tasks might conflict with each others [31]. Intuitively, if two users share an interest in the same type of side information but without having a similar purchase behaviour, the two loss components may have different optimisation directions, making the combined loss hard to optimise and requiring a trade-off between the two objectives. However, it requires tremendous efforts to tune one (or more) hyperparameter(s) to find a good trade-off solution between the two objectives [34, 81]. Moreover, entity side information is typically heterogeneous, meaning that it may consist of many different feature types (e.g. age, gender and education level) and be represented by different data types (e.g. binary-values, categorical-values or real-values). These types of side information usually play different roles when contributing to the generation of each entity's representation and need to be jointly modeled to capture their heterogeneous semantics. However, existing side information-aware models usually use one or more fixed hyperparameter(s) [33, 34] to control the importance of all different types of side information. Alternatively, they follow the paradigm of multi-task learning by using a constant value to balance the sum of the main loss and the side information-aware loss [28, 59], thereby leading to a lack of generalization and/or reduced performance robustness.

Furthermore, while there have been many powerful neural network-based recommendation models proposed in recent years [14, 15, 48, 62], most of these models are unable to give stable recommendation results. Indeed, as we can see from Figure 1, with the different random initialisation of the model's parameters, high variances can be observed in the performances of both the conventional model (i.e. MF [48]) and the deep neural network-based model (i.e. LightGCN [14]) over different embedding dimensions in both the Epinions and Foursquare datasets, demonstrating the lack of stability of these models. In this paper, we argue that a better initialization of the model's parameters encapsulating the entity side information could alleviate this issue, and allows the underlying recommendation model to find a stable local optimal recommendation solution.

To address the aforementioned issues, we propose a *novel pre-training scheme* for leveraging the side information in recommender systems, namely, we first pre-train the embeddings of entities using their side information, and then fine-tune the pre-trained embeddings using an existing recommendation model. Specifically, we explore two types of graph-structured data to capture the interdependent relationships among the entities, and propose two pre-training models based on Graph Neural Networks (GNNs), namely, the **Single-P** model and the **Multi-P** model. The **Single-P** model learns the entity embeddings on the single-relational graphs using Graph Convolutional Networks [25], while the **Multi-P** model learns entity embeddings on multi-relational graphs using Composition-based Multi-Relational Graph Convolutional Networks [53]. With the expressive power of GNNs that recursively propagate messages and aggregate features over neighbours, our pre-training models are able to encode the within-entity context knowledge from the side information of users and items. Note that our proposed pre-training scheme is a general framework to pre-train entity embeddings with side information in recommender systems. Once these embeddings are obtained, they can be applied to existing general representation-based recommenders to enhance their effectiveness and stability. We deploy our pre-trained embeddings into four existing representative general recommender models, i.e., MF [48], NCF [15] NGCF [62] and LightGCN [14], to validate the effectiveness of our proposed pre-training scheme.

The contributions of this work can be summarized as follows[1]:

(1) We introduce a novel pre-training scheme for leveraging side information by first pre-training the entity embeddings using entity side information and then fine-tuning them using an existing recommender model.

(2) We propose two pre-training models using graph neural networks, namely, the **Single-P** and the **Multi-P** models, which learn entity embeddings based on the single-relational and the multi-relational graphs, respectively, where both types of graphs are constructed from the entity

[1] Our source code is available at https://github.com/pretrain/pretrain.

side information. Both of our models can be deployed to fine-tune and enhance existing general representation-based recommender systems.

(3) An extensive empirical evaluation of our pre-training model – through the fine-tuning of four existing recommender models on three real-world datasets – shows that our pre-training scheme can significantly enhance those four models in terms of both the recommendation performance and the model stability.

The remainder of this paper is organized as follows. In §2, we position our work in the literature. §3 introduces all relevant notions used in this paper and formally defines the task. §4 describes our pre-training scheme and details the two proposed pre-training models. The experimental setup and the results of our empirical experiments are presented in §5 and §6 respectively, followed by some concluding remarks in §7.

2 RELATED WORK

In this section, we give a brief introduction about four bodies of related works: *recommendation models, integrating side information for recommendation, graph neural networks* and *graph neural recommendation models.*

2.1 Recommendation Models

Recommender systems are essential tools that help resolve the information overload, which have attracted much interest in both academia and industry. Among the various methodologies in the evolution of recommender systems, the *representation-based* methods, which learn latent embeddings for users and items, have been shown to be the most effective and popular ones in the literature because of their capability of capturing complex user preferences and items' popularity [15, 48]. Moreover, with the recent development of various deep neural networks, such as Convolution Neural Networks (CNNs) [76], Recurrent Neural Networks (RNNs) [23, 37], Attention Networks [22, 77] and Graph Neural Networks (GNNs) [14, 38, 62], more and more deep neural network-based recommendation models are being proposed to cope with various recommendation scenarios, such as temporal-aware [37, 78], social-aware [33, 72, 74], knowledge-aware [17, 61] and entity-context-aware [64] recommendation, making the learned embeddings capture more contextual information. In particular, traditional collaborative filtering methods cannot deliver satisfactory recommendations in the presence of various critical issues such as the data sparsity issue in the datasets and the cold start problem when recommending items to new users. To address this data sparsity issue, existing works have leveraged the self-supervised learning method in order to generate enough augmented samples [18, 27, 68]. Another line of research focusing on exploiting the side information of users and items has been widely recognized as an important modeling dimension to address these issues [6, 67]. The main focus of our present work is to provide a general scheme to integrate such widely available heterogeneous side information of both users and items into general representation-based recommender system.

2.2 Integrating Side Information for Recommendation

Indeed, before the prevalence of deep neural network-based recommendation methods, there have been many variants of the Matrix Factorization-based methods, such as the sparse linear methods with side information (SSLIM) [40], the hierarchical Bayesian matrix factorization method [43] and factorization machine [11, 46], which adopt the *integration scheme* that incorporates the entity side information by combining the recommendation loss function with an extra side information-aware loss. Even in recent years, this line of research still pervades many works in the literature. For example, xLightFM [20] is proposed to tackle the high memory-consumption issue of factorization machine [46] and its variants [7, 11] by using the quantization-based [30] and neural architecture search [44] method. Moreover, the HIRE model proposed by Liu et al. [34] uses a weighted matrix factorization to encode both flat and hierarchical forms of side information into the users' and items' representations, while combining the recommendation loss and two side information-aware losses. Although the HIRE [34] model can effectively incorporate two types of features, it needs to explicitly design different objectives for different types of features. Another line of research examined the integration of side information using deep neural networks, such as the stacked denoising auto-encoder [58] and the marginalized denoising auto-encoder [29]. More recently, many recommendation models have explored using Variational auto-encoders (VAEs) [42, 66, 69], which jointly encode user ratings and side information during the training, in order to overcome the (often) high-dimensionality of side information. However, most of these methods only consider one type of relation from the entity features, namely they treat all the feature columns equally, thereby ignoring the variance in the feature types' importance to the recommendation performance. Moreover, most of the existing methods [19, 20, 34, 46] adopt the integration scheme, which needs a trade-off between the recommendation loss function and the side information-aware loss, thereby restricting the model design and making it hard to deploy into other more effective general recommender systems. Instead, in this paper, we propose a general scheme for pre-training entity embeddings using the entity side information, such that these embeddings can be fine-tuned by an existing representation-based recommender system.

2.3 Graph Neural Networks

GNNs are powerful frameworks for learning representations of graph-structured data. They have shown superior performances not only in the network analysis task [2, 25, 39, 55, 63, 70] but also in other domains, such as natural language processing [36, 71, 82], recommender systems [14, 62, 65] and molecular design [21]. Recently, various graph learning technique [2, 9, 41, 45] have been proposed to further boost the accuracy and efficiency of existing GNNs. For example, GRL [9] is more robust against the removal of vertexes by using the graph reconstruction technique so that it gains enhanced expressive power compared with existing GNNs. Moreover, contrastive learning and pre-training techniques are both incorporated in [41, 45] to generate additional views for each node in order to learn more generalized graph representations. Although effective, most of the existing GNNs focus on learning representations of nodes on *simple graphs* that contain nodes and relations of a single type. The Graph Convolutional Network (GCN) [25] model and its variants [12, 39, 55, 63] are probably the most popular models for handling this type of networks. However, real-world networks are normally organized with multiple types of relations (e.g. links in social networks can denote both friendships and co-worker relationships), which are commonly modeled through *multi-graphs* [49, 53]. Since these multi-graphs can capture more comprehensive information and richer semantics than the simple graphs, they have been widely used in many tasks that mine knowledge graphs, such as lexical word networks [52, 60] and biomedical knowledge graphs [3]. Relational-GCN [49] and Compositional-GCN [53] are two generalizations of the GCN model for handling multi-graphs. Inspired by the promising performance and generalizability of GNNs with multiple relations, we propose to use a multi-graph neural network to capture the within-entity knowledge based on the entity side information, where heterogeneous features of users and items are explored for pre-training the entity embeddings.

2.4 Graph Neural Recommendation Models

Inspired by the wide variety of applications in many fields, graph neural networks have also been widely applied in many recommendation models, showing promising results on various recommendation benchmarking datasets [14, 32, 61, 64]. Among which, NGCF [62] can be regarded as an early implementation of the GCN model for the recommendation task. Building on NGCF, LightGCN [14] obtains a better performance and a higher efficiency by removing redundant neural operations and networks from NGCF. Recently, GF-CF [50] has been proposed to further enhance the performance by incorporating the graph filtering method. However, GF-CF is not an embedding-based model, hence it cannot be adapted to our proposed scheme. Other variants of LightGCN including SGL [65] and UltraGCN [38] have also achieved a competitive performances, however they incorporate memory-consuming data augmentation methods, which will be more challenging if side information is also considered. Most of the existing graph-based recommenders [5, 14, 62, 65] consider the user-item interactions in a recommendation task as a user-item graph, and use the power of GNNs to capture the higher-order dependencies among the entities, resulting in an enhanced recommendation performance. Many prior works also exploited using GNNs to model various cross-entity knowledge (i.e. interactions across different types of entities) [32, 64] and within-entity knowledge (i.e. relations within a type of entity) [33, 73]. However, all of the existing recommendation models discussed above are either unable to effectively leverage the entity side information, or are unable to effectively encode the heterogeneous within-entity knowledge from the side information in a general manner. In this paper, we provide a general scheme for leveraging the heterogeneous entity side information using a novel graph neural pre-training scheme, as detailed in the following sections.

3 PRELIMINARIES

In this section, we introduce the notations used across the whole article and formally define our research task.

3.1 Notations

Throughout this paper, regular letters are used to denote scalars (e.g. n is the number of users), while calligraphy typeface alphabets are used to denote sets (e.g. $\mathcal{V}$ is the set of nodes in a graph). Matrices and vectors are denoted by bold letters with uppercase letters representing matrices and lowercase letters (or uppercase letters with a subscript) representing vectors (e.g. $\boldsymbol{F}_u$ is the user feature matrix and $\boldsymbol{H}_{u_i}$ is represented as the latent vector of a user i.).

Let $\boldsymbol{R} \in \mathbb{R}^{m \times n}$ be the user-item feedback matrix of n users and m items. In general, the entries of $\boldsymbol{R}$ can be either binary-valued (i.e. through implicit feedback such as the click and purchase behaviours) or real-valued (i.e. through explicit feedback such as the rating scores). However, not all of the values can be observed and the observed values may also contain noise. While the observed entries at least reflect the real users' interests on items, most of the entries are typically missing due to the natural sparsity of negative feedback.

To facilitate the description of graph neural networks, we use $\mathcal{V}$ to denote the set of nodes in a graph. If the graph contains only one type of edges, then we call it a *single-graph*, or simply a *graph*; while a graph containing multiple edge types is normally called a *multi-graph*. We use $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$ to denote a single-graph, where $\mathcal{E}$ is a tuple set with $(u, v) \in \mathcal{E}$ being an edge between nodes $u, v \in \mathcal{V}$ and $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$ is the d-dimensional feature metric of nodes. Then we denote a multi-graph $\mathcal{G}'$ by $\mathcal{G}' = (\mathcal{V}, \mathcal{E}', \mathcal{R}, X)$, where $\mathcal{R}$ is the set of edge types, each edge $(u, v, r) \in \mathcal{E}'$ represents that the relation $r \in \mathcal{R}$ exists from node u to v.

3.2 Task Definition

Given the feedback matrix $\boldsymbol{R}$, a typical recommendation task aims to predict the preference scores for all the user-item pairs. To address such a task, the *general representation-based recommendation models* typically learn latent embeddings for the users and items such that these latent entity embeddings can be used to reconstruct the given feedback matrix $\boldsymbol{R}$. Then, these learned embeddings can be in turn used to calculate the preference scores of the unseen user-item pairs. In particular,

Table 1. Main Notations Used in this Article

Notation	Description
$\boldsymbol{R}$	the matrix of implicit feedback data
$\boldsymbol{U}, \boldsymbol{V}$	the latent embeddings of users and items
$\boldsymbol{F}_u$	the feature matrix of users
$\boldsymbol{F}_v$	the feature matrix of items
$\boldsymbol{X}$	the feature matrix of all nodes
$\mathcal{L}$	the loss
$\mathcal{V}$	the set of nodes
$\mathcal{E}$	the set of edges
$\mathcal{R}$	the set of edge types
$\mathcal{G}_u, \mathcal{G}_u^{'}$	the single/multi-graph of users
$\mathcal{G}_v, \mathcal{G}_v^{'}$	the single/multi-graph of items
$\mathcal{G}$	a graph (or multi-graph)
$\mathcal{R}^{inv}$	the inverse relation
$\top$	the self-loop relation
$\boldsymbol{Z}_r$	the embedding of relation r
$\boldsymbol{B}_b$	the basis vector
ϕ	the composition operator
$\boldsymbol{W}_{\lambda(r)}$	the relation-type specific parameter
$\boldsymbol{W}_O, \boldsymbol{W}_I, \boldsymbol{W}_S$	the trainable weights
l	the number of neural layers
d	the dimension of embeddings
b	the number of basis vectors

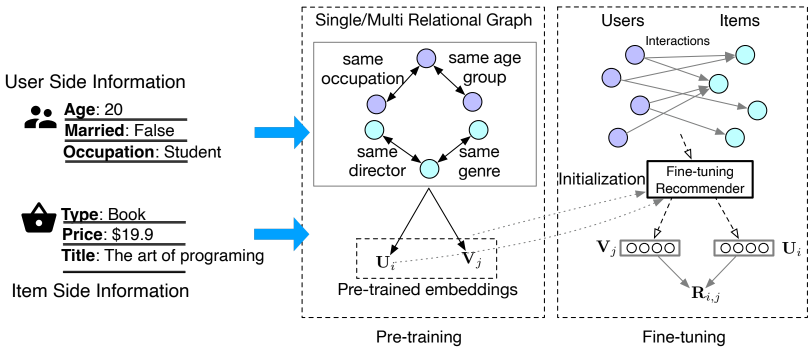

Fig. 2. An overview of our graph neural pre-training/fine-tuning scheme. Our pre-training model constructs relational graphs based on the feature of entities, and pre-trains the embeddings of entities by using **Multi-P** or **Single-P**.

such methods try to learn a model θ to obtain the embeddings of users and items:

$$\boldsymbol{R} \xrightarrow{\theta} \boldsymbol{U}, \boldsymbol{V}, \tag{1}$$

where $\boldsymbol{U} \in \mathbb{R}^{n \times d}$ and $\boldsymbol{V} \in \mathbb{R}^{m \times d}$ are the embeddings of users and items with d being the embedding dimension.

Usually, recommender systems use randomly initialised users and items' embeddings, i.e. $\boldsymbol{U}$ and $\boldsymbol{V}$ [62], respectively. As we have shown in the introduction section, these random initialised embedding have a strong impact on the model effectiveness and robustness. In real-world recommender systems, users and items are often associated with features, also called *side information*, such as the age groups of users and the textual descriptions/names of items. The different values of side information across different entities characterise the differences between entities, which is a type of within-entity knowledge. We use $\boldsymbol{F}_u$ and $\boldsymbol{F}_v$ to denote the feature matrices of the users and items, respectively. We later show that these types of features can be transformed into feature edges in multi-graphs. Then, the corresponding recommendation task is to learn a model θ:

$$\boldsymbol{F}_u, \boldsymbol{F}_v, \boldsymbol{R} \xrightarrow{\theta} \boldsymbol{U}, \boldsymbol{V}. \tag{2}$$

4 METHODOLOGY

We first introduce our pre-training scheme for recommender systems (§4.1), and detail two pre-training models, namely **Single-P** for single-graph pre-training (§4.2) and **Multi-P** for multi-graph pre-training (§4.3). We then describe the fine-tuning process using the existing recommender models (§4.6) and discuss connections between our scheme and the existing works in §4.7.

4.1 The Pre-training Scheme for Recommendation

In this paper, we argue that a suitable initialization of the entity embeddings encapsulating the entity side information is critical to help recommenders learn a stable and enhanced local optimal solution for the existing recommender systems. We propose to learn such an initialization of entity embeddings by exploiting knowledge from the entity side information. To this end, we propose a general pre-training scheme for leveraging the entity side information using graph neural networks. The overall scheme is illustrated in Figure 2. Our pre-training scheme consists of two processes: *pre-training* and *fine-tuning*. During *pre-training*, a graph neural network is used to learn an initialization of the entity embeddings based on both the side information of users and items (i.e. $\boldsymbol{F}_u$ and $\boldsymbol{F}_v$) and the feedback matrix $\boldsymbol{R}$. On the other hand, in the *fine-tuning* process, an existing recommendation model leverages the pre-trained embeddings as an embedding initialization and fine-tunes these embeddings by using the feedback matrix $\boldsymbol{R}$ only.

In order to learn the user-item preferences from the cross-entity contextual interactions between users and items, many recent models, such as NGCF [62] and LightGCN [14], have explored encoding the collaborative signals from the graph-structure interactions, showing promising performances. However, these methods only investigate the mutual interactions between users and items, ignoring the within-entity contextual knowledge, i.e. the collaborative signal within each type of entities, which can be acquired from the entity side information. To capture such within-entity contextual knowledge, we propose to use GNNs to pre-train the entity representations using the side information of both the users and items, so that the independent knowledge of each type of entities can be captured from the entity relations. Hence, extracting the relations from the entity side information is a crucial step for the pre-training process, since it determines how much information we can obtain from the entity features and how important such information can help to improve a recommendation model. To extract user-user and item-item relationships from their entity features (i.e. the users' and items' respective features), we propose to build two different types of feature graphs, i.e. the single-relational graphs and the multi-relational graphs, by constructing both the *homogeneous* and *heterogeneous* links between the entity pairs, respectively. We then explore how entity relations from various entity features affect the recommendation performance.

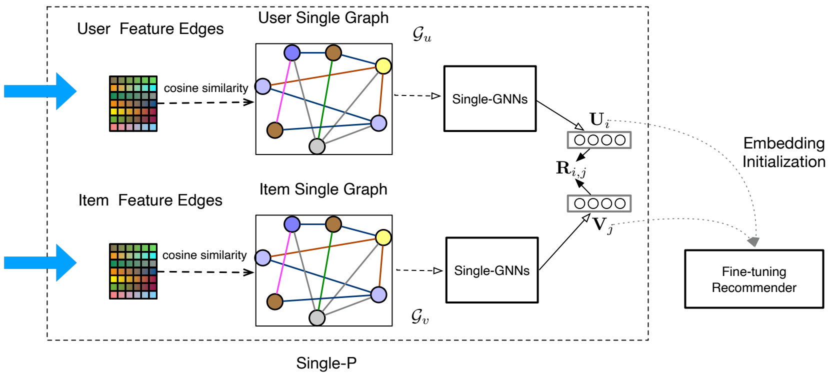

Fig. 3. An illustration describing the input features, single-relational graphs and output entity embeddings of **Single-P**.

4.2 Pre-training on Single-Graphs

We construct two single-graphs, i.e. $\mathcal{G}_u$ and $\mathcal{G}_v$, from the features of users and items respectively, by considering the similarities between users and items as the homogeneous edges, and calculate the edge weights using the cosine similarities between each pair of entities. For example, we construct a *user single-graph* $\mathcal{G}_u = (\mathcal{V}_u, \mathcal{E}_u, \boldsymbol{A}_u, \boldsymbol{X}_u)$ by taking all the users as the set of nodes $\mathcal{V}_u$ and all the user pairs as the set of edges $\mathcal{E}_u$ in the graph. For each $(i, j) \in \mathcal{E}_u$, we calculate the edge weight $\boldsymbol{A}_{ij}$ using the cosine similarity of their feature vectors (i.e. $\boldsymbol{F}_{u_i}$ and $\boldsymbol{F}_{u_j}$): $\boldsymbol{A}_{ij} = \frac{\boldsymbol{F}_{u_i} \cdot \boldsymbol{F}_{u_j}}{\|\boldsymbol{F}_{u_i}\| \|\boldsymbol{F}_{u_j}\|}$. $\boldsymbol{X}_u \in \mathbb{R}^{n \times d}$ is an initial node feature matrix of the graph, the values of which are initialized from the uniform distribution $\mathcal{U}(-0.01, 0.01)$. For brevity, in the following, we only describe the encoding process for the user single-graph $\mathcal{G}_u$, since the *item single-graph* $\mathcal{G}_v$ is constructed and processed in a similar fashion.

The Single-P model. To obtain the pre-trained embeddings of entities (i.e. users and items) and to exploit the potential correlation among entities based on their single-graphs, three GCN layers [25] are applied to encode the entity embeddings according to their relations. The key point of GCN is to propagate the feature information through neighbourhoods of nodes in each iteration during training. The detailed framework of the **Single-P** model is illustrated in Figure 3. Specifically, given a graph $\mathcal{G}_u$, the GCN model adopts the following propagation rule:

$$\boldsymbol{H}_u^{(l)} = \sigma\left(\hat{\boldsymbol{A}}_u \boldsymbol{H}_u^{(l-1)} \boldsymbol{W}_u^{(l-1)}\right), \tag{3}$$

where $\hat{\boldsymbol{A}}_u = \widetilde{\boldsymbol{D}}^{-\frac{1}{2}}(\boldsymbol{A}_u + \boldsymbol{I})\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}$ is the symmetric normalized adjacency matrix[2], $\boldsymbol{W}_u^{(l)}$ is the weight matrix of the l^{th} layer, and $\sigma(\cdot)$ denotes an activation function (e.g. the ReLU function). $\boldsymbol{H}_u^{(l)}$ is the hidden node representation in the l^{th} layer with $\boldsymbol{H}_u^{(0)} = \boldsymbol{X}_u$. As mentioned earlier in this section, we use the uniform distribution to initialize $\boldsymbol{X}_u$, which means that $\boldsymbol{H}_u^{(0)}$ also starts with this uniform distribution. In particular, we initialize the entity embeddings using the entity side information by training over the side information encapsulated in $\boldsymbol{A}$ instead of directly incorporating an initial embedding consisting of the entity side information.

[2] $\widetilde{\boldsymbol{D}}$ is defined as $\widetilde{\boldsymbol{D}}_{ii} = \sum_j (\boldsymbol{A}_u + \boldsymbol{I})_{ij}$, where $\boldsymbol{I}$ is the identity matrix.

To facilitate the later description of our proposed model, Eq. (3) can also be formulated in the message passing form [53]:

$$\boldsymbol{H}_{u_i}^{(l)} = \sigma\left(\sum_{(u_i,u_j)\in\mathcal{E}_u\cup(u_i,u_i)} \boldsymbol{W}_u^{(l-1)}\boldsymbol{H}_{u_j}^{(l-1)} \right), \tag{4}$$

where $\boldsymbol{W}_u^{(l-1)}$ is the layer-wise parameter and here we only consider the undirected relation and the self-loop relation. The final output embeddings of the maximum depth in the GCN layers, i.e. $\boldsymbol{U} = \boldsymbol{H}_u^{(l)}$, are the pre-training embeddings to be fed into the pre-training loss function. Similarly, the item embeddings are obtained by $\boldsymbol{V} = \boldsymbol{H}_v^{(l)}$, where $\boldsymbol{H}_{v_i}^{(l)} = \sigma\left(\sum_{(v_i,v_j)\in\mathcal{E}_v\cup(v_i,v_i)} \boldsymbol{W}_v^{(l-1)}\boldsymbol{H}_{v_j}^{(l-1)}\right)$, which is aligned with Eq. (4).

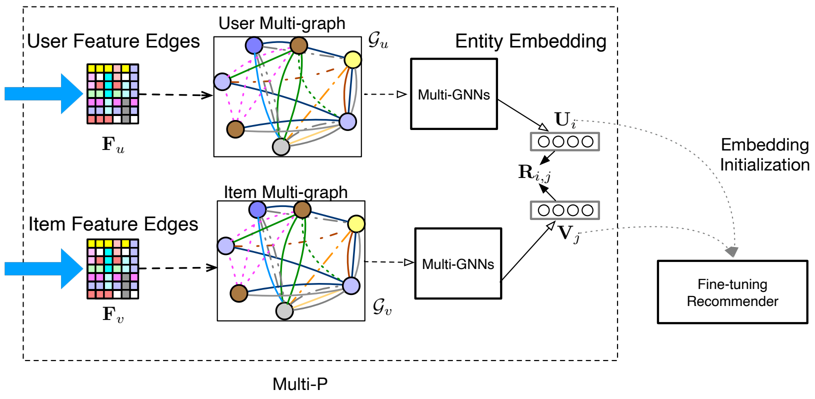

Fig. 4. An illustration describing the input features, multi-relational graphs and output entity embeddings of **Multi-P**.

4.3 Pre-training on Multi-Graphs

In the existing side information-aware models [6, 29, 58], different types of features associated with users and items contribute equally to the latent relationships between entities. However, in the real-world scenario, the features of users and items are typically heterogeneous, with different types of features having different usefulness for enhancing a recommender's performance. For example, users are normally associated with different types of features (e.g. age, gender and education level), which clearly characterize different aspects of the users' preferences [34]. To distinguish the importance of different feature types, we pre-train the entity embeddings through message propagation over the different feature types of the entities by using a multi-graph neural network [53].

Since the entity features can be real-valued, we first need to categorize such real-valued features into some groups, such that all the features of entities are sparsely categorized. Next, we can regard each feature category value as an edge type, and create an edge of this type between a pair of entities if they share the same feature value. In particular, to extract the heterogeneous relations between entities, we construct two multi-graphs for the users and items, i.e. $\mathcal{G}_u'$ and $\mathcal{G}_v'$, respectively. For example, to construct the *user multi-graph* $\mathcal{G}_u' = (\mathcal{V}_u, \mathcal{E}_u', \mathcal{R}_u, \boldsymbol{X}_u)$, we take all the users as the set of nodes $\mathcal{V}_u$, and the feature category values as the set of relations $\mathcal{R}_u$ in the graph. For any pairs of nodes $i, j \in \mathcal{V}_u$, we create an edge if the two users share the same feature category value (e.g. if both are in the age group 25-35). X_u is set to be the random initialized user embeddings. The *item multi-graph* $\mathcal{G}_v^{'}$ is constructed in a similar fashion. We use Figure 4 to illustrate the overall framework of the **Multi-P** model.

The Multi-P model. Given a multi-graph, e.g. the user multi-graph $\mathcal{G}_u^{'} = (\mathcal{V}_u, \mathcal{E}_u^{'}, \mathcal{R}_u, X_u)$, we first extend $\mathcal{E}_u^{'}$ and $\mathcal{R}_u$ with the corresponding inverse edges and relations:

$$\begin{aligned}\hat{\mathcal{E}}_u =& \mathcal{E}_u^{'} \cup \left\{\left(v, u, r^{-1}\right) \mid (u, v, r) \in \mathcal{E}_u^{'}\right\} \cup \{(u, u, \top) \mid u \in \mathcal{V}_u),\\ \hat{\mathcal{R}}_u =& \mathcal{R}_u \cup \mathcal{R}_u^{inv} \cup \{\top\},\end{aligned} \tag{5}$$

where $\mathcal{R}_u^{inv} = \left\{r^{-1} \mid r \in \mathcal{R}_u\right\}$ denotes the inverse relations (i.e. $(v, u, r^{-1}) = (u, v, r)$) and $\top$ indicates the self-loop. Then, inspired by Composition-GCN [53], the node embeddings are propagated through edges based on the following propagation rule:

$$H_{u_i}^{(l)} = \sigma\left(\sum_{(u_i,u_j,r)\in\hat{\mathcal{E}}_u} W_{\lambda(r)}^{(l-1)} \phi\left(H_{u_j}^{(l-1)}, Z_r^{(l-1)}\right)\right), \tag{6}$$

where l is the number of layers, $Z_r = \sum_{k=1}^{b} \alpha_{kr} B_k$ is the relation embedding with $\{B_1, B_2, \ldots, B_b\}$ being a set of learnable basis vectors and α_{kr} is the basis-specific learnable scalar weight. b is a hyperparameter corresponding to the number of basis vectors. $\phi(\cdot)$ is the composition operator defined as: $\phi\left(e_s, e_r\right) = e_s - e_r$, which is inspired by the TransE model [1]. $W_{\lambda(r)}^{(l-1)} \in \mathbb{R}^{d_1\times d_0}$ is a relation-type specific parameter, where $W_{\lambda(r)}$ are given below:

$$W_{\lambda(r)} = \begin{cases} W_O, & r \in \mathcal{R}_u \\ W_I, & r \in \mathcal{R}_u^{inv} \\ W_S, & r = \top \quad (\mathrm{self-loop}) \end{cases} \tag{7}$$

where W_O, W_I and W_S are all trainable weights.

Then the output embeddings of the final layer are taken as pre-training embeddings (i.e. $U = H_u^{(l)}$). We use the Binary Cross-Entropy (BCE) loss [15] described below as the objective function. The item embeddings are obtained similarly to Eq. (6).

4.4 Pre-training Loss

For most graph neural networks, the embeddings can be learned by the reconstruction of the graph structure or the labels of the corresponding entities [25]. Therefore, for both our Single-P and Multi-P models, we pre-train the user and item embeddings with a rating/interaction-based loss, in order to encapsulate the potential information between entities for recommendation. Specifically, with the embeddings of both users and items learned from the multi-relational or single-relation graphs, we construct our pre-training loss using the Binary Cross-Entropy (BCE) loss function [15]:

$$\mathcal{L}_{PT} = -\sum_{R_{i,j}\in R} R_{i,j} \cdot \log\left(\hat{R}_{i,j}\right) + \left(1 - R_{i,j}\right) \cdot \log\left(1 - \hat{R}_{i,j}\right) + \eta \left\|\Theta\right\|^2, \tag{8}$$

where $\hat{R}_{i,j}$ is the predicted score calculated by the embedding dot product $\hat{R}_{i,j} = U_i^{\mathrm{tr}} \cdot V_j$, Θ denotes all parameter embeddings and η denotes the regularization weight. Two explicit entity biases are also used for calculating the scores, following [48].

4.5 Complexity Analysis

The complexity of our pre-training scheme highly depends on the complexity of the underlying graph neural models. For example, our **Single-P** holds a complexity of $O\left(ld|\mathcal{E}| + ld^2|\mathcal{V}|\right)$ [25], which is the same as the GCN model. The complexity of our **Multi-P** model is $O\left((ld^2 + bd + b|\mathcal{R}|)|\mathcal{E}|\right)$, which is similar to the complexity of the Composition-GCN model [53], where l denotes the number of layers, b is the number of learnable basis vectors, d is the embedding dimension, and $|\mathcal{R}|$ is the number of relation types. The number of edges $|\mathcal{E}|$ in the entity graphs (multi-graphs) typically accounts for the largest complexity in the **Multi-P** model, and if the entities contain dense features, the number of edges $|\mathcal{E}|$ could be much larger than the number of interactions (e.g. the generated user multi-relational feature graph of the Movielens-1M dataset contains around 2.7 million edges). Some variants of these neural models, such as FastGCN [4] and ClusterGCN [8], could enhance the efficiency of our scheme but at the possible cost of reducing effectiveness. Once the entity embeddings are pre-trained, they can be reused and fine-tuned by many existing recommenders to enhance their effectiveness and stability. We leave the investigation of the impact and added-value of these existing variants for future work.

4.6 Fine-tuning with Existing Recommenders

Most of the modern recommenders are trained based on gradient-based optimization methods, which usually obtain locally-optimal solutions. Due to the non-convexity of their objective functions, parameter initialization plays an important role for the convergence and performances of these recommendation models [10, 15]. Most of the existing recommendation models initialize their embeddings using a uniform distribution [57], a normal distribution [48] or the Xavier uniform distribution [14, 62]. Due to the randomness of the generated embeddings and the lack of prior knowledge, these models often fall into some poor locally-optimal solutions, resulting in high instabilities, as illustrated in Figure 1.

To address this issue, we propose to initialize the entity embeddings of an existing recommendation model from the output embeddings of our pre-training model, then we further fine-tune these embeddings with the recommendation model's own optimizer. Specifically, we first pre-train both the embeddings of users and items by our proposed **Multi-P** or **Single-P** models until convergence, then feed these pre-trained embeddings into an existing recommendation model as the parameter initialization to train the recommendation model with the interactions/ratings only. The training frameworks of the pre-training and fine-tuning processes are summarized in Algorithm 1 and Algorithm 2, respectively. During the fine-tuning stage, the training objective is chosen depending on the underlying base model. For example, in this paper, we integrate our pre-trained embeddings into four existing recommender models, i.e., MF [48], NCF [15] NGCF [62] and LightGCN [14], where two training objective functions are used in these models, i.e. the BCE loss and the Bayesian Personalized Ranking (BPR) loss. Specifically, the MF, NGCF and LightGCN models optimize the pairwise BPR loss, which is formulated as follows:

$$\mathcal{L}_{FT} = -\sum \ln \sigma(\hat{\boldsymbol{R}}_{i,j} - \hat{\boldsymbol{R}}_{i,z}) + \eta \, ||\Theta||^2 , \tag{9}$$

while the NCF model is trained based on the BCE loss:

$$\mathcal{L}_{FT} = -\sum \boldsymbol{R}_{i,j} \cdot \log\left(\hat{\boldsymbol{R}}_{i,j}\right) + \left(1 - \boldsymbol{R}_{i,j}\right) \cdot \log\left(1 - \hat{\boldsymbol{R}}_{i,j}\right) + \eta \, ||\Theta||^2 , \tag{10}$$

where $\hat{\boldsymbol{R}}_{i,j}$ denotes the predicted scores for the observed interactions, and $\hat{\boldsymbol{R}}_{i,z}$ denotes the predicted scores for the unobserved interactions.

4.7 Discussion

The idea of pre-training embeddings for recommender systems has already been investigated in the literature. For example, to avoid saddle points and poorly performing local minima, both NCF [15] and CMN [10] apply the Generalized Matrix Factorization (GMF) as a pre-training model

Algorithm 1: The pre-training process

```
Input: Rating matrix $R$; Feature matrices $F_u$ and $F_v$; Multi-graphs $\mathcal{G}_u^{'}$ and $\mathcal{G}_v^{'}$ or
  Single-graphs $\mathcal{G}_u$ and $\mathcal{G}_v$.
Output: Pre-trained embeddings $\hat{U}$, $\hat{V}$.
Initialise embeddings $U$, $V$ and other learnable parameters $\Theta$;
while not convergent do
    $\mathcal{L}_{PT} = 0$;
    for each training instance in $\mathcal{G}_u$ and $\mathcal{G}_v$ (or $\mathcal{G}_u^{'}$ and $\mathcal{G}_v^{'}$) do
        Propagate information according to Eq. (6);
    end
    for each training instance in $R$ do
        Compute epoch loss $\nabla\mathcal{L}$ according to Eq. (8);
    end
    $\mathcal{L}_{PT} \leftarrow \mathcal{L}_{PT} + \nabla\mathcal{L}$;
    Update $\Theta, U, V$;
end
```

Algorithm 2: The fine-tuning process

```
Input: Rating matrix $R$; Pre-trained embeddings $\hat{U}$, $\hat{V}$; A general recommender $q$.
Output: The recommended list $\mathcal{S}$ for each user.
Inherit $\hat{U}$ $\hat{V}$ to initialise $q$;
Initialise other learnable parameters $\hat{\Theta}$;
while not convergent do
    $\mathcal{L}_{FT} = 0$;
    for each training instance in $R$ do
        Compute epoch loss $\nabla\mathcal{L}$ according to Eq. (9) or Eq. (10);
    end
    $\mathcal{L}_{FT} \leftarrow \mathcal{L}_{FT} + \nabla\mathcal{L}$;
    Update $\hat{\Theta}, \hat{U}, \hat{V}$;
end
Do recommendation to find the recommended list $\mathcal{S}$ based on $\hat{U}$ and $\hat{V}$;
```

to initialize the embedding weights of users and items. In particular, the embedding vectors of users and items in GMF are simply obtained by training from the weighted output of the embedding dot product using the interaction matrix:

$$\hat{R}_{u,v} = W\sigma\left(U_u \odot V_V\right), \tag{11}$$

where $\odot$ denotes the element-wise product of vectors, σ is an activation function and W is the trainable parameter. However, these models are unable to leverage the within-entity knowledge from the heterogeneous entity features. We note that our pre-training model using the graph neural network can be seen as a generalization of the GMF model, since our **Multi-P** model can be reduced to the GMF model by removing all the links of the constructed multi-graphs and setting the maximum depth of the multi-graph neural network to be 1. More recently, Hao et al. [13] exploited how to use GNN to conduct pre-training for downstream tasks, inspired by the GNN pre-training [16]. However, their proposed models only leverage the graph structure, which lacks the ability to incorporate heterogeneous side information about the entities, compared with our proposed scheme.

In relation to the graph neural-based recommender systems, the most relevant works to our **Multi-P** model is the Multi-GCCF model [51], which, similarly, considers the user-to-user and item-to-item relations as graphs and uses a graph convolution network to train the embeddings of the two entities. However, the entity graphs in Multi-GCCF are constructed from the rating/click matrix, rather than from the entity side information. Hence, the heterogeneous relations from the entity side information cannot be captured by the Multi-GCCF model.

5 EXPERIMENTAL SETUP

In this section, we first introduce the research questions that we aim to answer in this paper (§5.1). Next, we present the datasets used for conducting the experiments as well as the relevant pre-processing procedures to prepare the datasets including all interaction data and different types of side information (§5.2). Finally, we present the experimental settings and describe the used baselines (§5.3).

5.1 Research Questions

We aim to answer the following research questions:
RQ1. Do our pre-trained models help existing recommendation systems obtain better performances?
RQ2. Do our pre-trained models outperform the existing state-of-the-art recommenders that use side information?
RQ3. Are the performance improvements gained through our pre-training models due to the within-entity knowledge?
RQ4. Does the pre-training process help to improve the stability of the existing models?
RQ5. Does the pre-training process help to alleviate the classical *cold-start* problem?
RQ6. How do the embeddings dimension and different ranking cut-offs affect the recommendation performances of the pre-trained recommenders?

Table 2. Statistics of the datasets.

Dataset	# Users	# Items	# Interactions	# User/Item Features
Foursquare	2,060	2,876	27,149	2,108/47
Movielens-1M	6,040	3,704	1,000,209	21/18
Epinions	5,598	4,064	542,741	10/10

5.2 Datasets

To evaluate the effectiveness of our introduced pre-training scheme, we use three datasets, namely Foursquare[3], Movielens-1M[4], and Epinions[5]. Table 2 shows their statistics. For the Movielens-1M dataset, the users' features (i.e. side information) are "gender", "age" and "occupation", while the items' features are the 18 different genres. For the Foursquare dataset, we use the tags provided by online users as features for the restaurants, while we represent each user as a bag-of-words feature

[3] https://sites.google.com/site/yangdingqi/home/foursquare-dataset [4] https://grouplens.org/datasets/movielens/
[5] http://cseweb.ucsd.edu/~jmcauley/datasets.html

vector from his/her own reviews (stopwords are removed). Similarly, for the Epinions dataset, we represent both users and items with bag-of-words feature vectors from their associated reviews with stopwords removed, selecting the 10 most frequent words to represent both users and items. Among the three datasets, there is only one real-valued feature, namely ages in the Movielens-1M dataset, which needs to be pre-processed into one-hot representations by a categorization operation. Specifically, the users' age feature in the Movielens-1M dataset is categorized into 8 age groups, each with a step of 10 years. Then an one-hot vector is used to represent the users' age feature. Below, we summarise how to construct the single-relational graph and the multi-relational graph for our proposed **Multi-P** and **Single-P** models, respectively.

Single-relational graph construction. To construct a single-relation graph for **Single-P**, we first need to build the one-hot vectors for each user and item. Taking the Movielens-1M dataset as an example, we know that there are 18 different genres for all movies; therefore, each movie is represented as a binary vector of size 1×18. If one movie is labelled as a comedy movie, its vector will have 1 at the corresponding column of "comedy" and 0 elsewhere if no other labels are given. Hence, the feature matrix F_v of the Movielens-1M dataset has a size of 3704×18. Next, we compute cosine similarities between each pair of the users' one-hot vectors or each pair of the items' one-hot vectors so that we obtain cosine similarities between each user pair and each item pair. Such a single relational graph will have a size of 3704×3704 containing all similarity values. Finally, this single-relational graph can be used as an input for the proposed **Single-P** model. We follow a similar procedure to compute the single-relational graph for users.

Multi-relational graph construction. To construct a multi-relational graph for **Multi-P**, we also follow a similar procedure to build the one-hot vectors for each user and item so as to compute the feature matrices F_v and F_u. Instead of using the cosine similarity to capture the similarity values between each entity, the input of **Multi-P** is a series of graphs, where each graph contains all entities possessing one specific type of side information. Therefore, if we take again the Movielens-1M dataset as an example, the input of **Multi-P** for items will be 18 different graphs, where each graph has a size of $m \times m$. Here, the value of m depends on how many entities are involved in the multi-relational graph. Different with the graphs used by Compositional-GCN [53], our multi-relational graphs incorporate more types of edges and relations.

Table 3. Summary of all baselines and our proposed schemes.

Model	Side information	Pre-training	Graph-based
MF	✗	✗	✗
NCF	✗	✗	✗
NGCF	✗	✗	✓
LightGCN	✗	✗	✓
HIRE	✓	✗	✗
cVAE	✓	✗	✗
SSLIM	✓	✗	✗
SGL	✗	✗	✓
PT-GNN	✗	✓	✓
SimGCL	✗	✗	✓
Single-P	✓	✓	✓
Multi-P	✓	✓	✓

5.3 Experimental Settings

We use the leave-one-out splitting [15, 47, 48] method to split the interactions of each dataset into training, validation and testing sets. To speed up the evaluation, we adopt the sampled metrics [15, 48, 62], which randomly sample a small set of the non-interactive items as negative items (rather than take all the non-interactive items as negatives) of the validation and testing sets, and evaluate the metric performance on this smaller set. Here, we sample 100 negative items for each user in the testing sets for evaluation [15, 22, 48]. However, different from prior works [15, 48] that only use one oracle testing set per dataset with the sampled negative items, we construct 10 different testing sets with different sampled negative items for each dataset using different random seeds, in order to reduce the evaluation bias on some specific testing negatives [26]. Hence, the reported performance of each run is based on the average of the 10 testing sets. Three ranking evaluation metrics, namely the Normalised Discounted Cumulative Gain (NDCG), Recall and Mean Average Precision (MAP) metrics, are applied for evaluating the performances of our evaluated models.

We evaluate the effectiveness of our pre-training scheme by comparing it with ten existing state-of-the-art recommendation models. Among the baselines, four are general (representation-based) recommendation models, which are also used for the subsequent fine-tuning:

- **MF** [48]: This is the conventional matrix factorization model, which can be optimized by the Bayesian personalized ranking (BPR [47]) or the BCE losses.
- **NCF** [15]: This is a neural recommendation method, which learns the user and item embeddings while integrating the GMF & MLP models to capture their non-linear feature interactions.
- **NGCF** [62]: NGCF is devised to employ a multi-layer GCN on the user-item interaction graph to propagate the collaborative signals across multi-hops user-item neighbourhoods.
- **LightGCN** [14]: Building on NGCF, LightGCN has fewer redundant neural components compared with the NGCF model, which makes it more efficient and effective.

The other three baselines are recommenders that use an integration scheme to incorporate side information of both users and items:

- **HIRE** [34]: This is a side information-aware recommendation model, which combines the flat and hierarchical side information to alleviate the challenge brought by the heterogeneity of the side information.
- **cVAE** [6]: cVAE is a side information-aware recommendation model that uses the variational auto-encoder to encode the entity side information into entities for enhancing the performance.
- **SSLIM** [40]: This is a classical sparse linear recommender, which can use both the users and items' side information. In particular, we choose the binary representation to remain consistent with our feature representations.

In addition, we incorporate three other baselines, which use different methods to enhance their corresponding graph-based recommenders:

- SGL [65]: SGL uses self-supervised learning, including the node dropout, edge dropout and random walk augmentation techniques to generate multiple representations of users and items based on the structure of the graph. In addition, SGL has the ability to explore hard negative samples.
- PT-GNN [13][6]: PT-GNN uses the pre-trained GNN model to enhance the embeddings of the cold-start users or items. The pre-training task of PT-GNN consists in directly reconstructing the cold-start user/item embeddings by mimicking the meta-learning setting via episode-based training Vinyals et al. [56]. In [13], many state-of-the-art baseline models (including LightGCN [14], GraphSAGE [12] and GAT [55]) have been shown to be enhanced by PT-GNN. We choose the

[6] In [13], no explicit name has been given for the proposed graph pre-training model. For simplicity, we use PT-GNN to denote the model.

variant using GraphSAGE as one of our baselines due to its competitive overall performance. In the following, for simplicity, we use PT-GNN to denote the variant using GraphSAGE.

- SimGCL [75]: This is a recently proposed graph contrastive recommender with a high effectiveness and flexibility. SimGCL can enhance the contrastive representations of users and items by introducing a regulated noise sampled from the uniform distribution instead of relying on any graph augmentation techniques.

Furthermore, we compare our proposed pre-training scheme with variants, consisting of pre-training only and multi-task learning:

- Single-P (pre. only) & Multi-P (pre. only): These two models only use the users and items' embeddings learned during the **Single-P** or **Multi-P** pre-training stages to make recommendations. By comparing these two variants with the models pre-trained by our proposed scheme, we can directly observe the obtained improvements.
- $\text{MTL}_{+\textbf{Single-P}}$ & $\text{MTL}_{+\textbf{Multi-P}}$: These two variants use multi-task learning to train **Single-P** or **Multi-P** together with a baseline recommender, instead of following our proposed scheme. These two variants use $\mathcal{L}_{PT} + \beta * \mathcal{L}_{FT}$ as the overall training loss, where β is a constant and the detailed equations of $\mathcal{L}_{PT}$ and $\mathcal{L}_{FT}$ can be found in Section 4.3 and Section 4.6, respectively.

We apply our two pre-training models on the four existing widely used representation-based baselines, namely MF, NCF, NGCF and LightGCN. We use **Single-P** and **Multi-P** to denote the two pre-training models, respectively, while $\text{MF}_{+\textbf{Multi-P}}$ stands for a model, pre-trained by **Multi-P** and fine-tuned with MF. To summarise, we use Table 3 to indicate which model uses side information, pre-training and graph-based techniques, respectively.

We adopt the Adam [24] optimizer in both **Multi-P** and **Single-P** as well as the four fine-tuning baselines. To determine the values of all hyperparameters, we randomly sample one interaction for each user as the validation set and tune the hyperparameters on it for all of the models. In particular, we tune the pre-training models (i.e. **Multi-P** and **Single-P**) by varying the learning rate in $\left\{10^{-2}, 10^{-3}, 10^{-4}\right\}$ and the regularization weight η in $\left\{10^{-2}, ..., 10^{-5}\right\}$. The learning rates of the baseline models are also tuned according to the suggested ranges from the original papers. The depths for all GNNs of the graph-based recommenders (i.e. NGCF, LightGCN, SGL and SimGCL) and the pre-training models are kept to 3 with each layer having a size of 64, while the dropout ratios of all GNNs vary among $\{0.3, 0.4, ..., 0.8\}$ as suggested in the existing literature [14]. We set the maximum number of training epochs to 500; the batch size to 1000 and the latent dimension to 64 for all models. Moreover, we use an early stopping strategy, i.e., we apply a premature stopping if NDCG@10 on the validation data does not increase for 50 successive epochs. Note that the embedding dimension d is a hyperparameter for both the pre-training models and the fine-tuning models (i.e. the baseline models). For a fair comparison, we set this hyperparameter to 64, since most of our experimental models can almost achieve their best performances for this dimension size across our three datasets. To make a fair comparison between our proposed scheme and those three baselines, i.e., HIRE, cVAE and SSLIM, which also incorporate the side information of both users and items, we train the baselines using the same side information used by our proposed scheme for pre-training. For our **Multi-P** model, the number of learnable basis vectors b is set to 10, which is empirically tuned from the set {5, 7, 10, 20} by using the validation set for all datasets.[7] For those two multi-task learning model variants, i.e., $\text{MTL}_{+\textbf{Single-P}}$ & $\text{MTL}_{+\textbf{Multi-P}}$, we tune the hyperparameter β among {0.1, 0.5, 1} following [65].

[7] Note that a more thorough tuning of this parameter may further improve the recommendation performances, but we did not observe a clear performance trend over different b values in our experiments, and under this setting we have already obtained excellent performances that can be used to draw our conclusions.

6 RESULTS AND ANALYSIS

In this section, we report the results obtained from four main experiments aimed at answering the research questions listed in §5.1. In particular, we first address **RQ1** by analysing whether both **Single-P** and **Multi-P** can help to improve the four existing representative recommenders. Then, we further compare the performances of these models with three competitive recommenders, which incorporate different graph-based techniques and three recommenders that leverage both the users' and items' side information (**RQ2**). We provide an ablation study where we randomly drop {20%,40%,60%,80%} of the entity features during the pre-training process in order to seek an answer to **RQ3**. To answer **RQ4**, we conduct experiments over different random seeds, and analyze the standard deviations of these models' performances. To address **RQ5**, we conduct an analysis, where we compare the best performing pre-trained models (i.e. LightGCN$_{\textbf{+Single-P}}$ and LightGCN$_{\textbf{+Multi-P}}$) with PT-GNN and LightGCN across different groups of users to examine whether our proposed scheme can help to alleviate the *cold-start* problem. Finally, to answer **RQ6**, we provide a detailed analysis of the performances of the pre-training models on different embedding dimensions and different cut-offs for the recommended items.

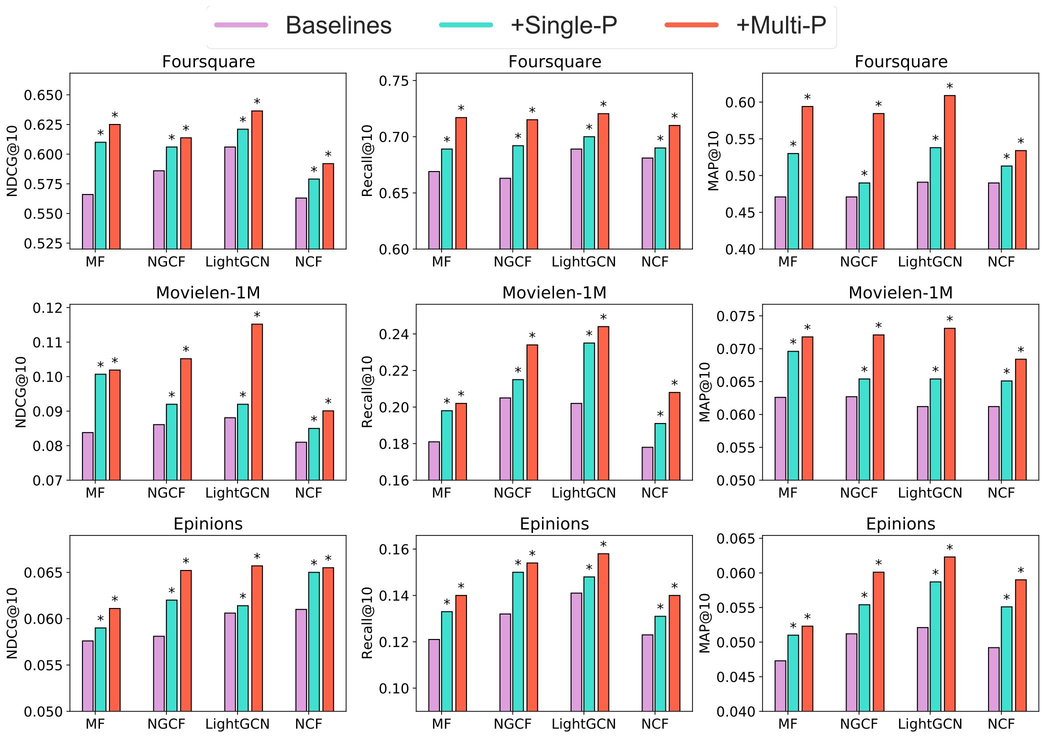

Fig. 5. Performance comparison of the four selected existing recommenders with **Single-P** and **Multi-P** pre-training processes. We use $*$ to denote a significant difference between the performances of the baselines and their pre-trained variants, according to the paired t-test with the Holm-Bonferroni correction for p<0.01.

Table 4. Performance comparison of recommenders with side information. The best and second best performances are marked in boldface or underlined, respectively. We use $*$ to denote a significant difference between the performances of the side information-aware baselines and the best proposed model, according to the paired t-test with the Holm-Bonferroni correction for $p < 0.01$.

Model	Foursquare		Movielens-1M		Epinions	
	NDCG	MAP	NDCG	MAP	NDCG	MAP
HIRE	0.5232*	0.4575*	0.0997*	<u>0.0692</u>*	0.0667*	0.0518*
cVAE	0.5326*	0.4438*	0.0678*	0.0453*	0.0532*	0.0410*
SSLIM	0.5894*	0.4691*	0.0655*	0.0441*	0.0533*	0.0401*
SGL	0.6051*	0.5691*	0.0739*	0.0485*	0.0710*	0.0484*
PT-GNN	0.6048*	0.5687*	0.0755*	0.0501*	0.0697*	0.0503*
SimGCL	0.6121*	0.5891*	0.0758*	0.0541*	<u>0.0723</u>*	0.0481*
$\text{MF}_{\textbf{+Single-P}}$	0.6206	0.5901	0.0979	0.0646	0.0527	0.0434
$\text{MF}_{\textbf{+Multi-P}}$	<u>0.6249</u>	<u>0.5944</u>	<u>0.1019</u>	**0.0718**	0.0587	0.0498
$\text{NGCF}_{\textbf{+Single-P}}$	0.6016	0.5683	0.0713	0.0450	0.0708	0.0454
$\text{NGCF}_{\textbf{+Multi-P}}$	0.6138	0.5844	0.0752	0.0461	0.0719	0.0485
$\text{LightGCN}_{\textbf{+Single-P}}$	0.6162	0.5940	0.0952	0.0631	0.0717	<u>0.0594</u>
$\text{LightGCN}_{\textbf{+Multi-P}}$	**0.6364**	**0.6089**	**0.1068**	0.0689	**0.0792**	**0.0623**
$\text{NCF}_{\textbf{+Single-P}}$	0.5677	0.4939	0.0870	0.0551	0.0620	0.0531
$\text{NCF}_{\textbf{+Multi-P}}$	0.6021	0.5340	0.0913	0.0584	0.0691	0.0583
Single-P (pre. only)	0.5758	0.5093	0.0700	0.0431	0.0517	0.0421
Multi-P (pre. only)	0.5912	0.5235	0.0812	0.0481	0.0601	0.0510
$\text{MTL}_{\textbf{+Single-P}}$	0.6012	0.5093	0.0725	0.0453	0.0522	0.0431
$\text{MTL}_{\textbf{+Multi-P}}$	0.6100	0.5337	0.0810	0.0482	0.0615	0.0530

6.1 Effectiveness of Pre-training

To validate the effectiveness of our pre-training scheme, we compare the performances of the four selected recommender models (i.e. MF [48], NCF [15], NGCF [62] and LightGCN [14]) with their pre-trained variants under the pre-training processes defined by our **Multi-P** and **Single-P**. Figure 5 reports the recommendation performances comparison in terms of the NDCG, Recall and MAP metrics at a rank cut-off of 10. From Figure 5, we can clearly observe that, over the three used datasets, all the selected 4 recommender models exhibit significantly improved performances when **Multi-P** is applied. Moreover, we can also see that a baseline pre-trained with our **Multi-P** can always outperform the baseline pre-trained with **Single-P**. This is somewhat expected, as **Multi-P** is trained using the multi-graphs constructed from the entities' side information. Therefore, **Multi-P** is capable of capturing the heterogeneous relations between entities within the side information, in contrast to the **Single-P**, which can only leverage the similarity between each feature vector.

To conclude on **RQ1**, we have shown that our proposed **Multi-P** model can effectively leverage various users and items' side information, thereby enhancing the existing representation-based recommenders with significant performance improvements, consistent across the three used datasets, three measures and four baselines.

6.2 Effectiveness of Integrating Side Information

Having shown that our proposed **Multi-P** is effective at enhancing the performances of the existing recommenders through the leveraging of side information, we next examine whether these recommenders with our pre-training scheme perform better than the existing state-of-the-art models. To

Table 5. Standard deviations (denoted std.) and means of the NDCG@10 performances over 50 random seeds. Lower standard deviation (in bold) means a better stability.

Model	Foursquare		Epinions	
	std.	mean	std.	mean
MF	0.0197	0.5163	0.0127	0.0501
MF$_{\textbf{+Single-P}}$	0.0046	0.6206	0.0049	0.0527
MF$_{\textbf{+Multi-P}}$	**0.0029**	0.6249	**0.0043**	0.0587
NGCF	0.0237	0.5162	0.0108	0.0681
NGCF$_{\textbf{+Single-P}}$	0.0053	0.6016	0.0042	0.0708
NGCF$_{\textbf{+Multi-P}}$	**0.0044**	0.6138	**0.0038**	0.0719
LightGCN	0.0209	0.5365	0.0092	0.0706
LightGCN$_{\textbf{+Single-P}}$	0.0050	0.6262	0.0039	0.0717
LightGCN$_{\textbf{+Multi-P}}$	**0.0039**	0.6264	**0.0031**	0.0792
NCF	0.0283	0.4621	0.0159	0.0591
NCF$_{\textbf{+Single-P}}$	0.0091	0.5777	0.0079	0.0620
NCF$_{\textbf{+Multi-P}}$	**0.0086**	0.6021	**0.0064**	0.0691

answer this, we further compare the pre-trained models (i.e. MF$_{\textbf{+Multi-P}}$, NCF$_{\textbf{+Multi-P}}$,NGCF$_{\textbf{+Multi-P}}$ and LightGCN$_{\textbf{+Multi-P}}$) with their corresponding baselines pre-trained with **Single-P** as well as three state-of-the-art recommenders where both side information of users and items are used. Table 4 reports the recommendation performances in terms of the NDCG and MAP metrics at rank cut-off 10 for each model across all three datasets. From Table 4, we observe that although the relative performance ranking of systems is different across the used datasets, the three baselines (i.e. HIRE, cVAE and SSLIM) do not achieve the highest performances on any of the used datasets. Specifically, the LightGCN$_{\textbf{+Multi-P}}$ model performs the best on both the Epinions and Foursquare datasets, outperforming the NDCG@10 score of the HIRE side information-aware baseline by 18.7% (0.0667 $\rightarrow$ 0.0792) and 21.6% (0.5232 $\rightarrow$ 0.6364), respectively. For the Movielens-1M dataset, MF$_{\textbf{+Multi-P}}$ achieves the best performance on the MAP metric, outperforming the HIRE model by 3.76% (0.0692 $\rightarrow$ 0.0718). Furthermore, we compare the pre-trained models with four variants that also use side information: Single-P (pre. only), Multi-P (pre. only), MTL$_{\textbf{+Single-P}}$ and MTL$_{\textbf{+Multi-P}}$. Recall that Single-P (pre. only) and Multi-P (pre. only) are two variants that only use the pre-training models without fine-tuning; MTL$_{\textbf{+Single-P}}$ and MTL$_{\textbf{+Multi-P}}$ are two variants that use multi-task learning to train the models. From Table 4, we find that the variants using multi-task learning always outperform the two pre-training only variants. However, none of these variants can reach the best or second best performances on all three used datasets, further demonstrating our proposed scheme's superiority. We also notice that models pre-trained by our proposed scheme cannot consistently outperform the pre-training only variant. For example, the NCF$_{\textbf{Single-P}}$ model is less effective than the pre-training only variant **Single-P** (pre. only) on the Foursquare dataset. This suggests that the performance of **Single-P** has decreased after being fine-tuned by the NCF model. Such an observation is related to the well-known issue of training a deep neural network with a warm restart [35] when a pre-trained model cannot even achieve the previous local optima during the fine-tuning process. We leave such an issue to future work, for example investigating how to at least maintain the previous local optima when fine-tuning a recommender system.

In addition to comparing our proposed scheme with side information-based baselines, we incorporate another three baselines, which use different graph-based methods to enhance the recommendation performance instead of relying on side information. Specifically, the SGL model uses the graph self-supervised learning method, PT-GNN uses the graph pre-training method, and SimGCL uses the graph contrastive learning method. By comparing the SGL, PT-GNN and SimGCL models with our proposed scheme in Table 4, we observe that our proposed scheme can achieve the best performances for all cases in terms of NDCG@10. This observation shows that using our proposed pre-training scheme to integrate side information outperforms baselines that use other competitive graph-based techniques.

Overall, to answer **RQ2**, we have shown that the four selected representative recommendation models with our **Multi-P** pre-training scheme can outperform the other three baseline models that also leverage entity side information using the integration scheme on all the three used datasets.

6.3 Ablation Study of Side Information

Having observed that all the evaluated existing baseline models are significantly improved by our pre-training scheme, we now check whether these improvements are actually the result of using the within-entity knowledge captured by our pre-training models (**RQ3**). To answer this question, we conduct an ablation study to examine the effect of randomly removing entity features, thereby revealing the connection between the performance improvements and the within-entity knowledge. Specifically, we randomly drop different proportions ({20%, 40%, 60%, 80%}) of entity features during the pre-training process, and evaluate the recommendation performances of the fine-tuned models given these pre-trained embeddings. Figure 6 reports the obtained results. From Figure 6, we can see that all the NDCG@10 performances of all fine-tuned models decrease as the features dropout ratio increases from 0% (i.e. no dropped features) to 80% (80% of features are dropped) in all the three datasets. This result suggests that randomly dropping entity features does hurt the overall recommendation performance. This result also suggests that the performance improvements are indeed gained from the entity features and our **Single-P** and **Multi-P** models are able to accurately capture the within-entity knowledge from the entities' side information.

6.4 Stability Analysis

In the previous sections, we have demonstrated the general applicability and effectiveness of our introduced pre-training scheme. To address **RQ4**, we calculate the standard deviations of the NDCG@10 performances of the baseline models (i.e. MF, NCF, NGCF and LightGCN) and their enhanced variants by our **Multi-P** and **Single-P** models. Table 5 presents the obtained results on the Foursquare and Epinions datasets. From the table, we observe that **Multi-P** markedly improves the performances and stabilities of all the used baselines, with much smaller observed standard deviations in each paired comparison of a baseline model with and without the use of the pre-training scheme. Noticeably, although less effective than our **Multi-P** model, the **Single-P** model can also bring a marked stability enhancement to all baselines, which demonstrates the superiority of the graph pre-training scheme for recommender systems. To conclude, our proposed pre-training scheme can enhance the performance of each of the representation-based recommender systems as well as their stability, thereby alleviating the high variances issue observed in Figure 1.

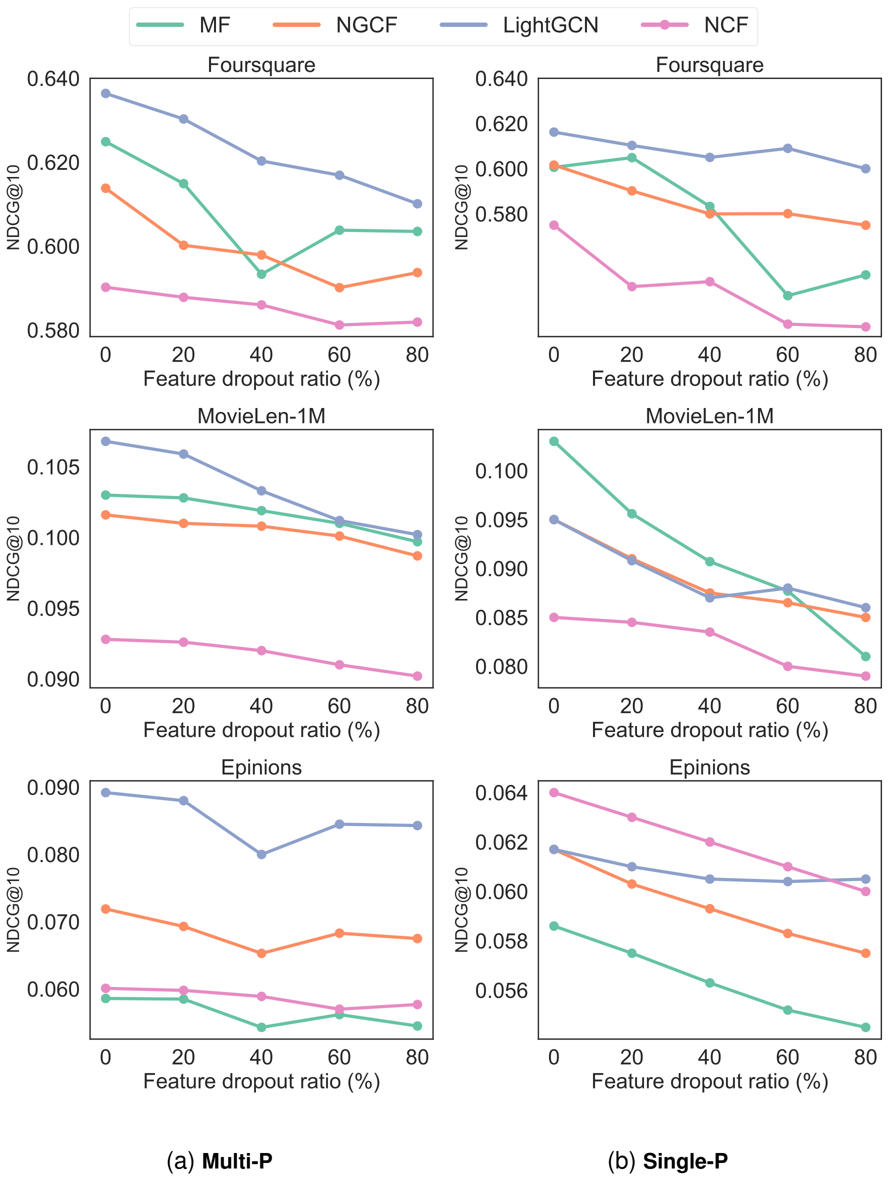

Fig. 6. Results of the baselines deploying the **Multi-P** and **Single-P** models over different dropout ratio of entity features.

6.5 *Cold-start* Analysis

To answer **RQ5**, we evaluate the performances of LightGCN, PT-GNN, LightGCN$_{\mathbf{+Single\text{-}P}}$ and LightGCN$_{\mathbf{+Multi\text{-}P}}$ on the Foursquare and Epinions datasets[8] over different groups of users, respectively. Specifically, following existing work [17, 33], we consider that the *cold-start* users

[8] We only use the Foursquare and Epinions datasets because the Movielens-1M dataset has only users with more than 20 interactions. However, typically, users with more than 20 interactions can hardly be called as *cold-start* users.

Table 6. NDCG@10 performances of our proposed scheme and those of the baselines across different groups of users on the two used datasets.

Model	Foursquare			Epinions		
	Overall	Cold-start	Regular	Overall	Cold-start	Regular
LightGCN	0.6012	0.5117	0.6238	0.0611	0.0532	0.0673
PT-GNN	0.6048	0.5334	0.6239	0.0697	0.0603	0.0700
LightGCN$_{\textbf{+Single-P}}$	0.6162	0.5458	0.6397	0.0717	0.0653	0.0731
LightGCN$_{\textbf{+Multi-P}}$	0.6364	0.5529	0.6458	0.0792	0.0701	0.0802

are those users with fewer than 10 interactions and the *regular* users are those users with more than 10 interactions. In Table 6, we report the NDCG@10 performances of the overall (i.e. all users included), *cold-start* (i.e. *cold-start* users only) and *regular* groups (i.e. *regular* users only) using all the evaluated models, respectively. In particular, we specifically choose PT-GNN as a baseline since it is especially designed to improve the *cold-start* recommendation [13]. In addition, we choose LightGCN and its two pre-trained variants LightGCN$_{\textbf{+Single-P}}$ and LightGCN$_{\textbf{+Multi-P}}$ for their excellent overall performances as shown in Table 4. From Table 6, we observe that our LightGCN$_{\textbf{+Single-P}}$ and LightGCN$_{\textbf{+Multi-P}}$ models consistently outperform the original LightGCN model and the pre-training baseline, i.e. PT-GNN, on both used datasets across different groups of users. It is worth noting that although PT-GNN markedly outperforms the LightGCN baseline for the *cold-start* users on the Foursquare dataset (0.5117 $\rightarrow$ 0.5334) and on the Epinions dataset (0.0532 $\rightarrow$ 0.0603), the improvements for the *regular* users are relatively marginal, especially for the Foursquare dataset, where NDCG@10 is only improved from 0.6238 to 0.6239. On the contrary, our proposed scheme can boost the recommendation performance for different groups of users instead of only focusing on the *cold-start* users. The reason why our proposed scheme can consistently improve the performance of the original model is that we use abundant side information to construct the relational graphs. Since the side information is available for different groups of users (sometimes the *regular* users have even more attributes), in general our proposed scheme can evenly enhance the representations of users. In comparison, PT-GNN relies on the interaction graph to construct the embeddings of the *cold-start* users, which might only benefit these *cold-start* users instead of all users.

To conclude, we have shown that our proposed scheme is able to alleviate the cold-start problem, while still ensuring an effective recommendation for all other users in comparison to strong baselines.

6.6 Hyperparameter Analysis

To answer **RQ6**, we study how the embedding dimension affects the recommendation performance. Figure 7 shows the performance comparison results of our pre-trained models (i.e. LightGCN$_{\textbf{+Single-P}}$, NCF$_{\textbf{+Single-P}}$, NCF$_{\textbf{+Multi-P}}$ and LightGCN$_{\textbf{+Multi-P}}$) with their baselines (i.e. LightGCN and NCF). From Figure 7, we observe that the size of the embedding dimension does affect the final recommendation performances of all these evaluated models. We can also observe that when the size of the latent dimensions is $\leq$40, all three models show relatively poor performances, which can be further boosted when the dimension size increases; almost all models' performances reach their highest points when the dimensions are between 60 to 80. Recall that, for a fair comparison, we fixed the embedding dimension to 64 for all the implemented models, which can be further justified from these obtained results. Moreover, Figure 7 demonstrates that our pre-training scheme can bring consistent improvements to the exiting models across different embedding dimensions. We also plot the performances evaluated by NDCG over different cut-offs (k) of the items ranking in Figure 8. We can also see that both LightGCN$_{\textbf{+Multi-P}}$ and NCF$_{\textbf{+Multi-P}}$ consistently improve over their baseline models for different cut-offs. In particular, we see that even for some low rank cut-offs (e.g. $k = \{1, 3, 5\}$) or for deep cut-offs (e.g. $k = 50$) our **Multi-P** model can still enhance the LightGCN and NCF models with a marked improvement, which means that our per-training scheme can help improve the recommendation performance under different circumstances when different amount of items (i.e. cut-off values) are chosen to be exposed to the users.

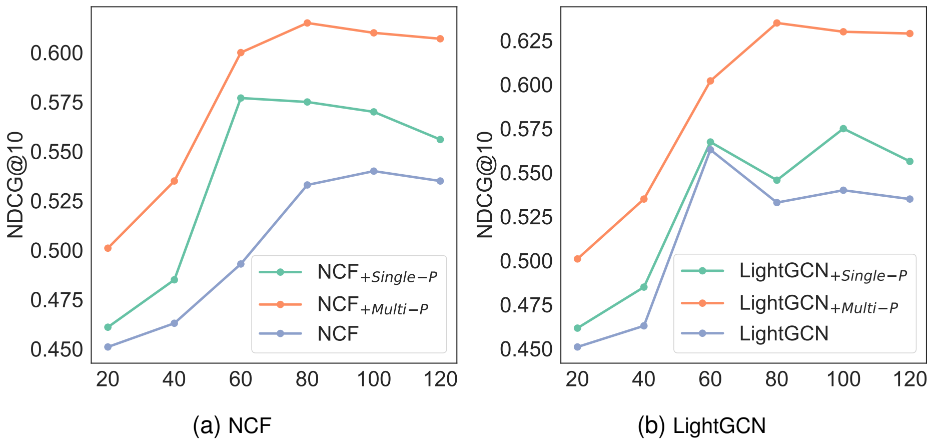

Fig. 7. The performance comparison over different dimensions on the Foursquare dataset.

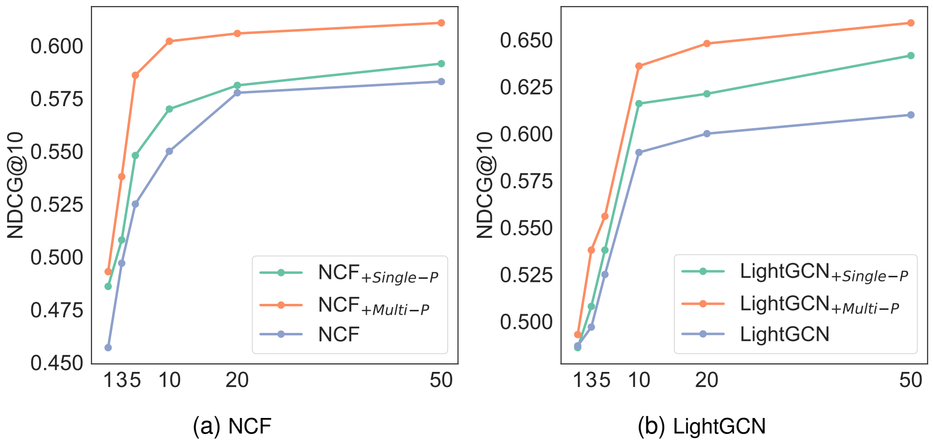

Fig. 8. The performance comparison over different cut-off values on the Foursquare dataset.

7 CONCLUSIONS

In this paper, we introduced a novel pre-training scheme for recommender systems to leverage the entity side information in a general manner. In particular, we proposed two models for pre-training the entity representations to capture the within-entity contextual knowledge, based on the graphs constructed from the entity side information. The extensive evaluation of our pre-training scheme with the fine-tuning of four existing representation-based recommenders showed that effectively pre-training the embeddings with both the users and items' side information improved these existing models in terms of both effectiveness (always significantly, see Figure 5) and stability (see Table 5). Furthermore, compared to the existing state-of-the-art recommender baselines, which integrate the same side information, our **Multi-P** model exhibited up to 7% improvement in NDCG@10 for the Movielen-1M dataset, 21% improvement on the Foursquare dataset and 48.6% improvement on the Epinions dataset (see Table 4). In addition, our **Multi-P** model consistently and significantly outperformed recent baselines that incorporate self-supervised learning, graph contrastive learning or graph pre-training techniques. Moreover, we have also shown through an in-depth analysis that by leveraging the side information through pre-training, our **Single-P** and **Multi-P** models can successfully alleviate the classical *cold-start* problem while ensuring effective recommendations for all other users. We also showed that the **Multi-P** model, pre-trained using multi-graphs, can always outperform the **Single-P** model, which suggests that more information is captured through the use multi-graphs. Our pre-training scheme provides a general framework for leveraging side information, which can be used to enhance a general representation-based recommendation model. As future work, we aim to investigate using other types of graph neural networks in the pre-training scheme. We will also explore whether the proposed pre-training scheme can benefit other recommendation scenarios, such as in sequential and session-based recommendations.

REFERENCES

[1] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Conference on Neural Information Processing Systems*. 1–9.

[2] Tianle Cai, Shengjie Luo, Keyulu Xu, Di He, Tie-yan Liu, and Liwei Wang. 2021. Graphnorm: A Principled Approach to Accelerating Graph Neural Network Training. In *International Conference on Machine Learning*. 1204–1215.

[3] David Chang, Ivana Balazevic, Carl Allen, Daniel Chawla, Cynthia Brandt, and Richard Andrew Taylor. 2020. Benchmark and Best Practices for Biomedical Knowledge Graph Embeddings. In *Biomedical Natural Language Processing Workshop*. 167–176.

[4] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *International Conference on Learning Representations*.

[5] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting Graph Based Collaborative Filtering: A Linear Residual Graph Convolutional Network Approach. In *AAAI Conference on Artificial Intelligence*. 27–34.

[6] Yifan Chen and Maarten de Rijke. 2018. A Collective Variational Autoencoder for Top-N Recommendation with Side Information. In *Deep Learning for Recommender Systems*. 3–9.

[7] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & Deep Learning for Recommender Systems. In *Workshop on Deep Learning for Recommender Systems*. 7–10.

[8] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-gcn: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks. In *SIGKDD International Conference on Knowledge Discovery & Data Mining*. 257–266.

[9] Leonardo Cotta, Christopher Morris, and Bruno Ribeiro. 2021. Reconstruction for Powerful Graph Representations. In *Conference on Neural Information Processing Systems*.

[10] Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative Memory Network for Recommendation Systems. In *SIGIR Conference on Research and Development in Information Retrieval*. 515–524.

[11] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-machine Based Neural Network for CTR Prediction. In *International Joint Conference on Artificial Intelligence*. 1725–1731.

[12] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Conference on Neural Information Processing Systems*. 1024–1034.

[13] Bowen Hao, Jing Zhang, Hongzhi Yin, Cuiping Li, and Hong Chen. 2021. Pre-training Graph Neural Networks for Cold-start Users and Items Representation. In *International Conference on Web Search and Data Mining*. 265–273.

[14] 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 Conference on Research and Development in Information Retrieval*. 639–648.

[15] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Web Conference*. 173–182.

[16] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. 2020. GPT-GNN: Generative Pre-training of Graph Neural Networks. In *SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[17] Chao Huang, Huance Xu, Yong Xu, Peng Dai, Lianghao Xiao, Mengyin Lu, Liefeng Bo, Hao Xing, Xiaoping Lai, and Yanfang Ye. 2021. Knowledge-aware Coupled Graph Neural Network for Social Recommendation. In *AAAI Conference on Artificial Intelligence*. 4115–4122.

[18] Tinglin Huang, Yuxiao Dong, Ming Ding, Zhen Yang, Wenzheng Feng, Xinyu Wang, and Jie Tang. 2021. MixGCF: An Improved Training Method for Graph Neural Network-based Recommender Systems. In *SIGKDD Conference on Knowledge Discovery & Data Mining*. 665–674.

[19] Bo Hui, Da Yan, and Wei-Shinn Ku. 2021. Node-Polysemy Aware Recommendation by Matrix Completion with Side Information. In *IEEE International Conference on Big Data*. 636–642.

[20] Gangwei Jiang, Hao Wang, Jin Chen, Haoyu Wang, Defu Lian, and Enhong Chen. 2021. xLightFM: Extremely Memory-Efficient Factorization Machine. In *SIGIR Conference on Research and Development in Information Retrieval*. 337–346.

[21] Chuanze Kang, Han Zhang, Zhuo Liu, Shenwei Huang, and Yanbin Yin. 2022. LR-GNN: A Graph Neural Network Based on Link Representation for Predicting Molecular Associations. *Briefings in Bioinformatics* 23, 1 (2022), bbab513.

[22] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive Sequential Recommendation. In *IEEE International Conference on Data Mining*. 197–206.

[23] Rahul Katarya and Yamini Arora. 2020. Capsmf: A Novel Product Recommender System Using Deep Learning Based Text Analysis Model. *Multimedia Tools and Applications* 79, 47 (2020), 35927–35948.

[24] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*.

[25] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

[26] Walid Krichene and Steffen Rendle. 2020. On Sampled Metrics for Item Recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1748–1757.

[27] Dongha Lee, SeongKu Kang, Hyunjun Ju, Chanyoung Park, and Hwanjo Yu. 2021. Bootstrapping User and Item Representations for One-class Collaborative Filtering. In *SIGIR Conference on Research and Development in Information Retrieval*. 317–326.

[28] Hui Li, Yanlin Wang, Ziyu Lyu, and Jieming Shi. 2020. Multi-task Learning for Recommendation over Heterogeneous Information Network. *IEEE Transactions on Knowledge and Data Engineering* 34 (2020), 789 – 802.

[29] Sheng Li, Jaya Kawale, and Yun Fu. 2015. Deep Collaborative Filtering via Marginalized Denoising Auto-encoder. In *Conference on Information & Knowledge Management*. 811–820.

[30] Defu Lian, Haoyu Wang, Zheng Liu, Jianxun Lian, Enhong Chen, and Xing Xie. 2020. Lightrec: A Memory and Search-efficient Recommender System. In *Web Conference*. 695–705.

[31] Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. 2019. Pareto multi-task learning. In *Advances in neural information processing systems*.

[32] Fan Liu, Zhiyong Cheng, Lei Zhu, Chenghao Liu, and Liqiang Nie. 2020. Aˆ 2-GCN: An Attribute-aware Attentive GCN Model for Recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2020), 1 – 1.

[33] Siwei Liu, Iadh Ounis, Craig Macdonald, and Zaiqiao Meng. 2020. A Heterogeneous Graph Neural Model for Cold-start Recommendation. In *SIGIR Conference on Research and Development in Information Retrieval*. 2029–2032.

[34] Tianqiao Liu, Zhiwei Wang, Jiliang Tang, Songfan Yang, Gale Yan Huang, and Zitao Liu. 2019. Recommender Systems with Heterogeneous Side Information. In *Web Conference*. 3027–3033.

[35] Ilya Loshchilov and Frank Hutter. 2016. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*.

[36] Qianwen Ma, Chunyuan Yuan, Wei Zhou, and Songlin Hu. 2021. Label-Specific Dual Graph Neural Network for Multi-Label Text Classification. In *Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*. 3855–3864.

[37] Jarana Manotumruksa, Craig Macdonald, and Iadh Ounis. 2017. A Deep Recurrent Collaborative Filtering Framework for Venue Recommendation. In *Conference on Information & Knowledge Management*. 1429–1438.

[38] Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. 2021. UltraGCN: Ultra Simplification of Graph Convolutional Networks for Recommendation. In *Conference on Information & Knowledge Management*. 1253–1262.

[39] Zaiqiao Meng, Shangsong Liang, Jinyuan Fang, and Teng Xiao. 2019. Semi-supervisedly Co-embedding Attributed Networks. In *Conference on Neural Information Processing Systems*. 6507–6516.

[40] Xia Ning and George Karypis. 2012. Sparse Linear Methods with Side Information for Top-n Recommendations. In *Recommender Systems Conference*. 155–162.

[41] Erlin Pan and Zhao Kang. 2021. Multi-view Contrastive Graph Clustering. In *Conference on Neural Information Processing Systems*.

[42] Bo Pang, Min Yang, and Chongjun Wang. 2019. A Novel Top-n Recommendation Approach Based on Conditional Variational Auto-encoder. In *Pacific-Asia Conference on Knowledge Discovery & Data Mining*. 357–368.

[43] Sunho Park, Yong-Deok Kim, and Seungjin Choi. 2013. Hierarchical Bayesian Matrix Factorization with Side Information. In *International Joint Conference on Artificial Intelligence*. 1593–1599.

[44] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. 2018. Efficient Neural Architecture Search via Parameters Sharing. In *International conference on machine learning*. 4095–4104.

[45] 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 *SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1150–1160.

[46] Steffen Rendle. 2010. Factorization Machines. In *IEEE International conference on data mining*. 995–1000.

[47] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Conference on Uncertainty in Artificial Intelligence*. 452–461.

[48] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural Collaborative Filtering vs. Matrix Factorization Revisited. In *Recommender Systems Conference*. 240–248.

[49] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. In *European Semantic Web Conference*. 593–607.

[50] Yifei Shen, Yongji Wu, Yao Zhang, Caihua Shan, Jun Zhang, B Khaled Letaief, and Dongsheng Li. 2021. How Powerful is Graph Convolution for Recommendation?. In *Conference on Information & Knowledge Management*. 1619–1629.

[51] Jianing Sun, Yingxue Zhang, Chen Ma, Mark Coates, Huifeng Guo, Ruiming Tang, and Xiuqiang He. 2019. Multi-Graph Convolution Collaborative Filtering. In *IEEE International Conference on Data Mining*. 1306–1311.

[52] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2018. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *International Conference on Learning Representations*.

[53] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2020. Composition-based Multi-Relational Graph Convolutional Networks. In *International Conference on Learning Representations*.

[54] Flavian Vasile, Elena Smirnova, and Alexis Conneau. 2016. Meta-prod2vec: Product Embeddings Using Side-information for Recommendation. In *Recommender Systems Conference*. 225–232.

[55] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

[56] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. *Advances in neural information processing systems* 29 (2016).

[57] Mengting Wan, Di Wang, Jie Liu, Paul Bennett, and Julian McAuley. 2018. Representing and Recommending Shopping Baskets with Complementarity, Compatibility and Loyalty. In *Conference on Information & Knowledge Management*. 1133–1142.

[58] Hao Wang, Xingjian Shi, and Dit-Yan Yeung. 2015. Relational Stacked Denoising Autoencoder for Tag Recommendation. In *AAAI Conference on Artificial Intelligence*. 3052–3058.

[59] Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Multi-task Feature Learning for Knowledge Graph Enhanced Recommendation. In *Web Conference*. 2000–2010.

[60] Shen Wang, Xiaokai Wei, Cicero Nogueira Nogueira dos Santos, Zhiguo Wang, Ramesh Nallapati, Andrew Arnold, Bing Xiang, Philip S Yu, and Isabel F Cruz. 2021. Mixed-curvature Multi-relational Graph Neural Network for Knowledge Graph Completion. In *Web Conference*. 1761–1771.

[61] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *SIGKDD international conference on knowledge discovery & data mining*. 950–958.

[62] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *SIGIR Conference on Research and Development in Information Retrieval*. 165–174.

[63] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying Graph Convolutional Networks. In *International conference on machine learning*. 6861–6871.

[64] Jiancan Wu, Xiangnan He, Xiang Wang, Qifan Wang, Weijian Chen, Jianxun Lian, and Xing Xie. 2022. Graph Convolution Machine for Context-aware Recommender System. *Frontiers of Computer Science* 16, 6 (2022), 1–12.

[65] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised Graph Learning for Recommendation. In *SIGIR Conference on Research and Development in Information Retrieval*. 726–735.

[66] Yaxiong Wu, Craig Macdonald, and Iadh Ounis. 2020. A Hybrid Conditional Variational Autoencoder Model for Personalised Top-n Recommendation. In *SIGIR on International Conference on Theory of Information Retrieval*. 89–96.

[67] Lianghao Xia, Chao Huang, Yong Xu, Peng Dai, Xiyue Zhang, Hongsheng Yang, Jian Pei, and Liefeng Bo. 2021. Knowledge-enhanced Hierarchical Graph Transformer Network for Multi-behavior Recommendation. In *AAAI Conference on Artificial Intelligence*. 4486–4493.

[68] Xin Xia, Hongzhi Yin, Junliang Yu, Qinyong Wang, Lizhen Cui, and Xiangliang Zhang. 2021. Self-supervised Hypergraph Convolutional Networks for Session-based Recommendation. In *AAAI Conference on Artificial Intelligence*, Vol. 35. 4503–4511.

[69] Zhe Xie, Chengxuan Liu, Yichi Zhang, Hongtao Lu, Dong Wang, and Yue Ding. 2021. Adversarial and Contrastive Variational Autoencoder for Sequential Recommendation. In *Web Conference*. 449–459.

[70] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations*.

[71] Shiyao Xu and Yang Xiang. 2021. Frog-GNN: Multi-perspective Aggregation Based Graph Neural Network for Few-shot Text Classification. *Expert Systems with Applications* 176 (2021), 114795.

[72] Junliang Yu, Hongzhi Yin, Min Gao, Xin Xia, Xiangliang Zhang, and Nguyen Quoc Viet Hung. 2021. Socially-aware Self-supervised Tri-training for Recommendation. In *SIGKDD Conference on Knowledge Discovery & Data Mining*. 2084–2092.

[73] Junliang Yu, Hongzhi Yin, Jundong Li, Min Gao, Zi Huang, and Lizhen Cui. 2020. Enhancing Social Recommendation with Adversarial Graph Convolutional Networks. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[74] Junliang Yu, Hongzhi Yin, Jundong Li, Qinyong Wang, Nguyen Quoc Viet Hung, and Xiangliang Zhang. 2021. Self-supervised Multi-channel Hypergraph Convolutional Network for Social Recommendation. In *Web Conference*. 413–424.

[75] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Nguyen Quoc Viet Hung. 2022. Are Graph Augmentations Necessary? Simple Graph Contrastive Learning for Recommendation. In *SIGIR conference on research and development in information retrieval*.

[76] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2019. A Simple Convolutional Generative Network for Next Item Recommendation. In *International Conference on Web Search & Data Mining*. 582–590.

[77] Weihua Yuan, Hong Wang, Xiaomei Yu, Nan Liu, and Zhenghao Li. 2020. Attention-based Context-aware Sequential Recommendation Model. *Information Sciences* 510 (2020), 122–134.

[78] Zixuan Yuan, Hao Liu, Yanchi Liu, Denghui Zhang, Fei Yi, Nengjun Zhu, and Hui Xiong. 2020. Spatio-temporal Dual Graph Attention Network for Query-poi Matching. In *SIGIR conference on research and development in information retrieval*. 629–638.

[79] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep Learning Based Recommender System: A Survey and New Perspectives. *Comput. Surveys* 52 (2019), 1–38.

[80] Yongfeng Zhang, Qingyao Ai, Xu Chen, and W Bruce Croft. 2017. Joint Representation Learning for Top-n Recommendation with Heterogeneous Information Sources. In *Conference on Information & Knowledge Management*. 1449–1458.

[81] Feipeng Zhao and Yuhong Guo. 2017. Learning Discriminative Recommendation Systems with Side Information. In *International Joint Conference on Artificial Intelligence*. 3469–3475.

[82] Hao Zhu, Yankai Lin, Zhiyuan Liu, Jie Fu, Tat-seng Chua, and Maosong Sun. 2019. Graph Neural Networks with Generated Parameters for Relation Extraction. In *Annual Meeting of the Association for Computational Linguistics*. 1331–1339.