Social Boosted Recommendation with Folded Bipartite Network Embedding

Hongxu Chen, Hongzhi Yin, Tong Chen, Weiqing Wang, Xue Li, Xia Hu

Abstract—With the prevalence of online social platforms, social recommendation has emerged as a promising direction that leverages the social network among users to enhance recommendation performance. However, the available social relations among users are usually extremely sparse and noisy, which may lead to an inferior recommendation performance. To alleviate this problem, this paper innovatively exploits the implicit higher-order social influences and dependencies among users to enhance social recommendations. In this paper, we propose a novel embedding method for general bipartite graphs, which defines inter-class message passing between explicit relations and intra-class message passing between implicit higher-order relations via a novel sequential modelling paradigm. Inspired by recent advances in self-attention-based sequence modelling, the proposed model features a self-attentive representation learning on the user-user implicit relations. Moreover, this paper also explores the inductive embedding learning for social recommendation problems to improve the recommendation performance in cold-start settings. The proposed inductive learning paradigm for social recommendation enables embedding inference for those cold-start users and items (unseen during training) as long as they are linked to existing nodes in the original network during evaluation. Extensive experiments on real-world datasets demonstrate the superiority of our method and suggest that higher-order implicit relationship among users is beneficial to improving social recommendations.

Index Terms—Social recommendation, bipartite graph embedding, network embedding

1 Introduction

Recent years have witnessed an increasing amount of research attention on social recommendation, which harnesses social relations to boost the performance of recommender systems [1], [2], [3], [4], [5]. The most impactful theories behind social recommendation are homophily assumption [6] and social influence [7]. In the recommendation context, homophily assumption assumes that similar users share similar preferences, which will lead to similar purchase decisions. In addition to homophily, social influence is another important aspect as it implies users' purchases are influenced by people having close social ties to them, such as friends, relatives, colleagues, etc. On this basis, social recommendation methods are widely applied as an enhancement upon traditional recommender systems with proven effectiveness in recommendation tasks [3], [5].

For mining user preferences, most recommender systems treat user-item interactions as an indispensable information source, which can be viewed as bipartite graphs. From the user-item bipartite graphs, vectorized representations (a.k.a. embeddings) of both the users and items can be learned via observed interactions, and future user-item interactions can be easily inferred via the learned embeddings [8], [9]. To enrich the semantics within the learned user embeddings, existing social recommender systems primarily adopt the user-user social relationships to thoroughly model user preferences [2], [3], [5]. Specifically, these methods learn two embeddings for each user from social domain and item domain, and then fuse both embeddings to obtain a comprehensive user representation used for estimating user-item interactions.

However, in recommendation tasks, the sparsity issue is protuberant when utilizing social relations to enrich the information in a user-item bipartite graph. Even for online social platforms, the observable connections between users are also sparse. For example, Yelp[1] is an online review platform where users can rate and review a business (e.g., a restaurant) they have visited. Besides, Yelp users are also able to make friends and join different communities. Whilst the platform is gaining extensive popularity over time, most users only use it as a tool for business searching, regardless of its social functions. Consequently, statistics of Yelp's user network (see Section 4) show that, though there are more than 70 million possible user-user links in Edinburgh, only less than 0.04% are actually established, contributing to an extreme sparsity of 99.96%. Data from Wisconsin shows even more scattered user friendships on Yelp with a 99.99% sparsity. Therefore, due to the sparsity issue, existing social recommendation may consistently fail as the limited social relations would not provide enough auxiliary information to improve the recommendation performance. A similar investigation from IBM reveals that Facebook delivered just 0.68% referral traffic on Black Friday, while the percentage was 0 on Twitter [10]. Furthermore, recent research on recommendation [1], [11], [12], [13] points out that the

- *H. Chen, H. Yin, T. Chen and X. Li are with the School of Information Technology & Electric Engineering, The University of Queensland, St Lucia, QLD 4072, Australia. E-mail: hongxu.chen, h.yin1, tong.chen@uq.edu.au, xueli@itee.uq.edu.au.*
- *W. Wang is with the Faculty of Information Technology, Monash University. E-mail: weiqing405@gmail.com.*
- *X. Hu is with the Department of Computer Science and Engineering at Texas A&M University. E-mail: xiahu@tamu.edu.*

(Corresponding author: Hongzhi Yin.)
Manuscript received September 29, 2019.

1. https://www.yelp.com/

performance gain from introducing social data is marginal due to the limited amount of social relations in the context of recommender systems.

On top of the sparsity issue, active users tend to interact with different items more frequently [14], and traditional recommender systems can easily identify the preferences of active users and recommend the right items to them. In contrast, inactive users are less likely to have abundant interactions with items, hence social recommenders intend to additionally incorporate the social interactions of users to maintain the recommendation quality on these inactive users. However, the performance of social recommenders are hugely constrained by the characteristics of users' social data. On one hand, inactive users are also reluctant to establish social connections with other users [15], so it is impractical to assume the sufficiency of social data for such users. On the other hand, as active users usually have a wide variety of friends [16], the casually formed friendships and inactive social friends are not fully filtered out by existing social recommenders, thus taking excessive noises when modelling each user's preferences from her/his social ties. Confronted with the sparse and noisy social data, the capability of mining high-quality latent semantics from users' interactions with users/items becomes rather important for social recommender systems. Unfortunately, while the explicitly observable user friendships can be easily modelled as first-order relations for social recommendation, the rich semantic information within indirect higher-order relationships are largely neglected by all the aforementioned social recommenders. For instance, two users who reviewed the same business or purchased the same item should possess an implicit social relationship where similar preferences can be implied. This inevitably sets an obstacle for obtaining satisfying recommendation results, particularly for inactive or new users whose interaction records are typically sparse.

To this end, in this paper, we seek solutions to remedy the deficiencies of existing social recommender systems. Specifically, we propose a general bipartite embedding method FBNE (short for ***F**olded **B**ipartite **N**etwork **E**mbedding*) for social recommendation. FBNE tackles the aforementioned challenges of social recommendation by exploring the higher-order relations among users and items in user-item bipartite graphs and learning user/item embeddings via a novel sequence-aware paradigm. We briefly introduce these two key aspects of FBNE below.

Exploring higher-order relations. In a user-item bipartite graph, we innovatively split the relations among user/item nodes into two types, namely **inter-class relations** and **intra-class relations**. Inter-class relations are observed links between users and items, while intra-class relations are implicitly inferred from the original bipartite graph for vertices of the same types (i.e., either users or items). To discover the hidden intra-class relations, FBNE employs a folding process that folds a bipartite graph into two homogenous networks for each vertex type. For example, in a user-item bipartite graph, the folding process will generate a user-user graph and an item-item graph. In the folded graphs, a user-user link represents the implicit higher-order links between two users that have purchased the same items, while an item-item link can be identified between two items that are commonly purchased together. Therefore, by leveraging the higher-order intra-class relations, we are able to augment the sparse user-user social graph data and generate more informative embeddings of both users and items.

Sequential influence modelling. Compared with the inter-class graphs, learning distinctive node representations from the folded intra-class graphs is non-trivial because the relationships are implicitly established. Recently, graph neural network-based (GNN-based) methods [17], [18], [19] start to show favorable effectiveness in social recommendation tasks [3], [5], where each node's representation is obtained by aggregating the information from its local neighbors. Nevertheless, their complex non-linear forms prevent them from modelling higher-order relationships, and the modelling of only first-order or-second order relationships is adopted for a trade-off between efficiency and performance. Hence, in FBNE, we consider sequence-based methods that learn node representations via node sequences sampled from graphs. As traditional sequence-based methods like node2vec [20], deepwalk [21], and metapath2vec [22] are computationally expensive when modelling long sequences [23], we propose a novel self-attention scheme to learn node representations via sequential influence modelling. Self-attention mechanism [24] is highly efficient and capable of mining the sequential and semantic patterns across the sampled node sequences, and the influence from noisy social data can be effectively mitigated by assigning larger attentive weights on influential nodes.

Apart from the effectiveness evaluation under the traditional social recommendation setting, to fully demonstrate our models capability of utilizing limited user-item interactions and user-user social relationships for recommendation, we further study a novel problem of inductive social recommendation. With the novel folding mechanism and self-attention based node embedding scheme, FBNE is fully able to learn representations of cold-start users and items. Meanwhile, we propose a joint training procedure that links different bipartite graphs to account for multiple information sources. As such, FBNE supports inductive representation learning that gathers information from observed user/item nodes and generate embeddings for those unseen users/items. Therefore, FBNE achieves superior performance on generic social recommendation problems, and can be effectively generalized to cold-start recommendation scenarios.

The primary contributions of our research are summarized as follows:

- To cope with the sparse and noisy social data in social recommendation tasks, we propose FBNE, a general yet effective bipartite graph embedding model. FBNE investigates higher-order relationships between nodes by folding the user-item bipartite graphs and learn representative user/item embeddings. The self-attention-based node embedding scheme enables FBNE to adaptively distinguishes important nodes from noisy ones.
- The proposed FBNE further supports inductive learning for new users/items, thus allowing for social recommendation under the cold-start setting. Based on the learned embedding of partially observ-

able nodes, FBNE can effectively generate the embedding for an inductive user/item node through aggregating the information from related nodes within multiple bipartite graphs.
- We conduct extensive experiments to evaluate the performance of FBNE on two real large-scale e-commerce datasets. In the comparison with state-of-the-art baselines, the advantageous performance of FBNE fully demonstrates its superiority in social recommendation tasks.

The remainder of the paper is organized as follows. We provide preliminaries and define the problems in Section 2. Section 3 details our proposed bipartite graph-based embedding methods. We report the experimental settings and results in Section 4 and Section 5, respectively. Section 6 reviews the related work and Section 7 concludes the paper.

2 Preliminaries and Problem Definitions

In this section, we introduce the necessary definitions and notations used in this paper.

Definition 1. **(User-User Graph)** A User-User graph, denoted as $\mathcal{G}_{UU} = (\mathcal{V}_U, \mathcal{E}_{UU})$, captures the social relationships among users, where $\mathcal{V}_U$ is a set of users and $\mathcal{E}_{UU}$ is a set of edges between users. Given two users u and u', if they are friends in the social network, there is an edge $e_{uu'}$ between them.

Definition 2. **(User-Business Bipartite Graph)** A User-Business graph $\mathcal{G}_{UB} = (\mathcal{V}_U, \mathcal{V}_B, \mathcal{E}_{UB})$ is a bipartite graph where $\mathcal{V}_U$ denotes the user set and $\mathcal{V}_B$ denotes the business (item) set (i.e. item set), while $\mathcal{E}_{UB} \subseteq \mathcal{V}_U \times \mathcal{V}_B$ is the set of edges between users and businesses. If a user u has visited a business b, there is an edge e_{ub} between them. If user ratings are available, the edge weight w_{ub} is defined as the rating; otherwise, the weight w_{ub} is set to 1.

Definition 3. **(Business-Attribute Bipartite Graph)** A Business-Attribute graph $\mathcal{G}_{BA} = (\mathcal{V}_B, \mathcal{V}_A, \mathcal{E}_{BA})$ is a bipartite graph where $\mathcal{V}_B$ denotes the business (item) set and $\mathcal{V}_A$ denotes the attribute set, while $\mathcal{E}_{va} \subseteq \mathcal{V}_B \times \mathcal{V}_A$ defines the inter-set edges between businesses and their attributes. If a business b has been described with an attribute a, there is a link between them. The weight w_{ba} is set to 1 if the edge exists.

Definition 4. **(Business-Category Bipartite Graph)** A Business -Category graph, denoted as $\mathcal{G}_{BC} = (\mathcal{V}_B, \mathcal{V}_C, \mathcal{E}_{BC})$, is a bipartite graph, where $\mathcal{V}_B$ denotes the business (item) set and $\mathcal{V}_C$ denotes the category set, and $\mathcal{E}_{BC} \subseteq \mathcal{V}_B \times \mathcal{V}_C$ defines the inter-set edges between businesses and their categories. If a business b is categorized as a category c, there is a link between them; otherwise, none. The weight w_{bc} is set to 1 if the edge exists.

Definition 5. **(Heterogenous Bipartite Graph)** A Heterogenous Bipartite Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a joint graph of multiple bipartite graphs, it consists of multiple different types of vertices, and multiple different types of links, where $\mathcal{V} = \mathcal{V}_U \bigcup \mathcal{V}_B \bigcup \mathcal{V}_A \bigcup \mathcal{V}_C$, and $\mathcal{E} = \mathcal{E}_{UU} \bigcup \mathcal{E}_{UB} \bigcup \mathcal{E}_{BA} \bigcup \mathcal{E}_{BC}$.

Finally, we formally define the problems investigated in our work. We aim to solve the following problems:

Problem 1. **(Bipartite Network Embedding)** Given a heterogenous bipartite graph for a social network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we aim to learn low-dimensional vector representations for each vertex in $\mathcal{V}$, such that in the embedding space, both the inter-class explicit interactions between vertices of different type and the implicit higher-order interactions between vertices of same type can be preserved. To keep the notation simple, we use bold lower-case letter $\mathbf{v}_i$ to denote the embedding vector for a vertex v_i, and use corresponding bold upper-case letter $\mathbf{V}$ to represent an embedding matrix.

Problem 2. **(Social Boosted Recommendation)** Given a heterogenous bipartite graph for a social network $\mathcal{G}$ (a joint graph of multiple bipartite graphs), we consider the following two recommendation tasks: 1). (**General Social Recommendation**) Given a user $u \in \mathcal{V}_U$, the goal is to recommend top-n items (businesses) $b \in \mathcal{V}_B$ that u is interested in. 2). (**Cold-start Social Recommendation**). For a cold-start user $u \notin \mathcal{V}_U$ (u is not seen during the training phrase, but has links to $\mathcal{V}_U$), our goal is to recommend top-n items (businesses) $b \in \mathcal{V}_B$ that u is interested in.

3 Folded Bipartite Network Embedding for Social Recommendation

In what follows, we detail our proposed FBNE which performs social recommendation by embedding the bipartite networks. The purpose of a bipartite graph embedding model is to preserve the linkage information between two disjoint sets of nodes (i.e., users and items) with learned low-dimensional node embedding vectors. These node embeddings are expected to be able to infer and reconstruct the original pairwise user-item links. To achieve this goal, FBNE models the bipartite graph from two perspectives: (1) message passing from observed links between two sets of nodes; and (2) higher-order message passing schemas between same typed nodes from unobserved but transitive links.

Bipartite graphs are special cases of general graphs, where links are formed between two disjoint sets of nodes, and these two sets of nodes are usually different in terms of the types, carrying different semantics. Compared with modelling homogenous graphs, additional information can be mined from bipartite graphs. First, the main structure of a bipartite graph can be directly modelled through the observed links between two sets of nodes (e.g., user-item interactions in a user-item bipartite graph), which reveals the information flow exchanged between two different types of nodes. We term such links as **inter-class relations**. Besides, for each specific node set, there exists unobserved higher-order message passing among the same typed nodes through transitive links (e.g., co-purchase relationship exists between two users if they have purchased one same item), which is referred to as **intra-class relations**. Therefore, learning embeddings for a bipartite graph by using traditional graph embedding methods is insufficient and will eventually lead to information loss, and a sensible embedding

(a) User-User Graph (b) User-Business Graph (c) Business-Category Graph (d) Business-Attribute Graph

Fig. 1: A Yelp network that consists of multiple bipartite graphs and a social graph

approach for bipartite graphs should be able to preserve both important properties.

3.1 Modelling Inter-Class Message Passing

First, to model the inter-class similarities between two sets of nodes, taking the user-item bipartite graph as an example, we adopt the idea of weight sharing in GCNs [17], [25], and create message passing channels for each user and item through their interactions in a user-item bipartite graph. The graph convolutions mimics the message passing schema, and the representation of each node is computed by aggregating the information passed from its direct heterogeneous neighbours. We assume each node has at least one inter-class links. Therefore, for each link in a user-item bipartite graph, we model the message passing as an information transformation from one end of the link to the other. For example, the message passing from item b_j to user u_i is the following form:

$$\mu_{ji} = \frac{1}{c_{ij}} \mathbf{W}^{ub} x_j^b \tag{1}$$

where c_{ij} is a normalization constant, which we choose to be either $|N(u_i)|$ or $\sqrt{|N(u_i)||N(b_j)|}$ (symmetric normalization), with $N(u_i)$ denoting the set of neighbors of user node i, and $N(b_j)$ denoting the set of neighbors of item node j. W^{ub} is the parameter matrix and x_j^b is the feature vector of item node j.

After the message passing step, we aggregate incoming messages from every node by using an aggregation function $Agg(\cdot)$ that aggregates the neighbors $N(u_i)$ connected by links between user i and her/his interacted items. Mathematically, we define the aggregation as follows:

$$\mathbf{h}_i^U = \sigma(\mathbf{W} \cdot Agg(\{\mu_{ji}, j \in N(u_i)\})) \tag{2}$$

where $\sigma(\cdot)$ is an element-wise activation function such as $ReLU(\cdot) = max(0, \cdot)$. W is a learnable parameter matrix. We will refer to Eq.(2) as a graph convolution layer [17], [25]. Analogously, at the other side of a given bipartite graph (e.g., all items in a user-item graph), the item embedding h_i^B is computed with the same parameter matrix W by performing graph convolutions according to Eq.(2). The graph convolution steps will capture the messages exchanged between two sets of nodes, and therefore preserve the main structure of a bipartite graph. Next, we discuss our design of the aggregation function $Agg(\cdot)$.

To aggregate the message passed from neighbourhood nodes, the original graph convolutional network (GCN) has to obtain the entire graph in order to run the graph convolutional kernel, which is inefficient and inflexible when handling large-scale bipartite graphs. To alleviate this problem, GraphSAGE [18] recently makes an attempt that allows GCN to sample and aggregate a fixed number of neighbours. However, for each node, GraphSAGE uniformly samples a number of neighbours to form a receptive field, which treats all neighbours equally and neglects the varied importance of different neighbourhoods. In fact, different neighbour nodes are not equally influential in terms of passing information to the target node. As a result, a naive neighbourhood sampling scheme can neither highlight influential neighbours nor filter out the noise from irrelevant neighbours.

In light of this, we follow our previous work [26] to sample the neighbours of a node based on their centrality measurements. To be specific, we investigate various centrality indices and perform biased sampling according to their centrality information. Figure 2 shows the adopted neighbourhood sampling strategy, which first ranks the neighbourhoods of a target node according to a specific centrality measurement (e.g., degree, closeness, betweenness, etc.), and then picks a fixed amount of top-ranked node sequences. The sampling approach guarantees that the computation of a node is based on its most influential neighbours. Furthermore, a node defines its own computation of embeddings and no longer needs the presence of the entire topology of a graph (it only requires each node to know its immediate adjacent neighbours).

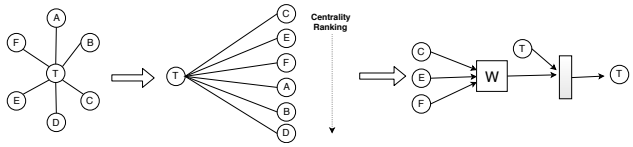


Fig. 2: Centrality based sampling

After the sampling, the model has to aggregate the information passed from the sampled nodes. According to existing works on GCNs, popular aggregation functions are $Mean(\cdot)$, $Max(\cdot)$, $LSTM(\cdot)$, $Pooling(\cdot)$, etc. We adopt the mean aggregator where it simply takes the element-wise mean of the vectors in $\{\mu_{ji}, j \in N(u_i)\}$. We do not compare different aggregators as this has been done by previous works [17], [18], [27], [28]. However, despite the significance of distinguishing the varied influences of different neighbour nodes, the mean aggregator assumes all neighbours contribute equally to represent a node, which can hardly generate optimal embeddings. Inspired by [2], [19] and attention mechanism [29], [30], we allow neighbours to contribute differently to a target node's latent factor

by assigning each neighbour node a different weight. The graph convolutional layer is now written as:

$$\mathbf{h}_i^U = \sigma(\mathbf{W} \cdot Mean(\{\alpha_{ji}\mu_{ji}, j \in N(u_i)\})), \quad (3)$$

where α_{ji} is the attentive weight for message μ_{ji} passed from neighbour (an item node in this case) $b_j, j \in N(u_i)$ to the target node (a user node). Specifically, we parameterize the attention weight α_{ji} on item j with a two-layer attention network. The input of the attention network is the message representation μ_{ji} through the interaction and the target user u_i's embedding $\mathbf{u}_i$. Formally, the attention network is defined as,

$$w_{ji} = \mathbf{W}_2^T \cdot \sigma(\mathbf{W}_1 \cdot [\mu_{ji} \oplus \mathbf{u}_i] + b_1) + b_2 \quad (4)$$

The final attention weights are obtained by normalizing the above attentive scores using Softmax function:

$$\alpha_{ji} = \frac{exp(w_{ji})}{\sum_{j \in N(u_i)} exp(w_{ji})} \quad (5)$$

3.2 Modelling Intra-Class Message Passing

In addition to inter-class relationship modelling, it is also crucial to model the intra-class message passing between two nodes of the same type. Compare with general homogeneous graphs, a special property of a bipartite graph is that links in a bipartite graph can be only formed between two disjoint sets of nodes, and there is no direct links between same typed nodes. For example, in Figure 3, there are two different sets of nodes U and V, and links between U and V construct the bipartite graph while there are no links among the nodes in the same set (i.e., $u \in U$ and $v \in V$). It is insufficient for a bipartite graph embedding model to only consider the direct links between two different sets of nodes, because the the implicit relationships between the same typed nodes also carry substantial semantic information.

Although no explicit links can be directly observed between same typed nodes, there always exists higher-order relationships among the nodes in the same set. If two nodes from set U are commonly connected by a node from set V, we regard these two nodes from U possess a higher-oder connectivity (second-order proximity in this example). For example, in Figure 3, node **1** and **2** are both connected to node **A**, which implies node **1** and **2** are also connected through the path **1-A-2**. We extend this simple observation to a more general claim that in a bipartite graph, for two vertices of the same type, if there exists a path between them, there should be certain implicit intra-class relationship between them.

Similar to the inter-class relationship modelling, we also consider the message passing among the nodes that are in the same set, and we name it the **intra-class message passing** in this paper. Intuitively, for two vertices of the same type, if there exists a path between them in a bipartite graph, there should be certain message passing between them, and the number of the paths and the distances between two nodes in a path indicate the strength of the messaged passed between them. For example, starting from node **7**, the intra-class message passing exists between node **7** and all the remaining nodes {**1,2,3,4,5,6**} through different paths (e.g., paths from node **7** to node **6** and **5** through node **D**; from node **7** to node **4** through the path {**D,5,C**}). Obviously, the strength of the message passings starting from node **7** to node **5** and **4** are different, the message from node **7** to node **5** is stronger than that from node **7** to node **4** as the distance between node **7** and **5** (distance of 2) is smaller than the distance between node **7** to node **4** (distance of 4).

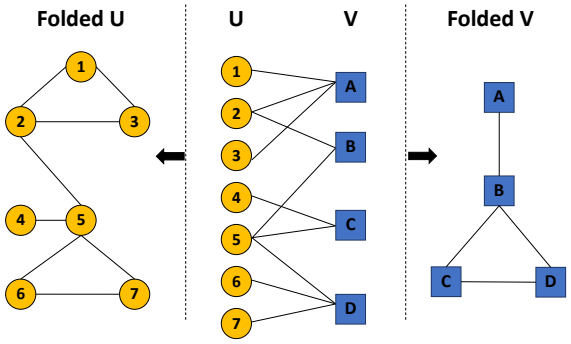


Fig. 3: Folding a bipartite graph

Unfortunately, counting the paths and distances between two vertices has a rather high complexity of an exponential order, which is infeasible for implementation on large networks. To encode such higher-order message passing among vertices in a bipartite network, we exploit the **folded bipartite graph** derived from the original bipartite graph, and generate node sequences from the folded graphs. By modelling the generated node sequences, the message passing schema and the signal strength between any two same typed nodes can be captured. Specifically, the bipartite graph is firstly converted to two folded homogenous graphs, as illustrated in Figure 3. In the middle of Figure 3, there is a bipartite graph consisting two different types of nodes (U and V), the folding process folds the original bipartite graph into two homogenous graphs (graph of U on the left and graph of V on the right) by linking the same typed vertices via transitive links through the other type of nodes. Mathematically, the folding process can be expressed via adjacency matrix self-multiplication:

$$\mathbf{A}_{folded} = \mathbf{A} \times \mathbf{A}^T \quad (6)$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the adjacency matrix of the original bipartite graph, $\mathbf{A}_{folded}$ is the folded graph adjacency matrix, in which two folded homogenous network can be obtained.

Then, to model the message passing and the signal strength between two intra-class nodes, for each folded homogenous graph, we sample two corpora of vertex sequences with random walks, then the embeddings are learned from the corpora which encodes high-order message passing among vertices. In what follows, we first elaborate how to generate two quality corpora for a bipartite network. After that, we introduce a novel embedding method to work directly on the generated node sequences to learn representative node embeddings.

Starting from each target node in a folded bipartite graph, we regulate a random walker to walk over the graph to generate a fixed length node sequence. Mathematically, for each target node u, we will obtain a node sequence sampled from the folded bipartite graphs, denoted as $S^u = (u_1, u_2, ..., u_l)$, where l is a scalar parameter, indicating the length of the sequence. We will explore the

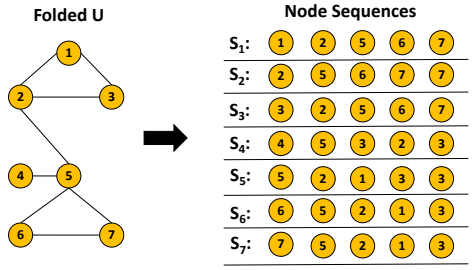


Fig. 4: Generated homogenous node sequences

effects of different sizes of l in the experimental discussions. As such, the random walker will respectively generate $|U|$ and $|V|$ node sequences for nodes in U and V. Figure 4 is an example of generated node sequences. The left part is a folded graph that only contains user nodes, while the right part shows 7 sequences (with length of 5) starting from each node in the folded graph generated by the random walker. All the sampled node sequences will form the corpora used for learning the node embeddings.

Previous sequence-based embedding methods such as Deepwalk [21] and Node2vec [20] sample node sequences through truncated random walks and feeds them into a Skip-Gram based on Word2Vec [31] model to learn node representations. However, these methods cannot be directly applied in our problem as Word2vec-based methods only uses a fixed window size to sample a small number of nearby nodes as contexts. Different from node2vec and deepwalk, in this paper, we expand the existing sequence-based approaches with a flexible sequential modelling scheme, which is also able to differentiate the strength of message passing posted by different nodes in a sequence. In other words, our assumption is that, in a node sequence, the starting node exchanges message to every other nodes, and the amount and the strength of the message passing is highly related to the importance of each node in the sequence. Therefore, the embedding method should be able to be aware of the varying dependencies between the staring node and other remaining nodes.

To serve this need, inspired by recent advances of transformer model [24] in sequential modelling of natural language processing, we borrow the idea of transformer to model the node sequences generated from each of the folded bipartite graphs. The Transformer model relies heavily on the proposed a self-attention modules to capture complex structures in sentences, and to retrieve relevant words (in the source language) for generating the next word (in the target language). Inspired by Transformer, we seek to build a new sequential embedding model based upon the self-attention approach, though the problem of sequential embedding is quite different from machine translation, and requires specially designed models.

We start with the self-attention module, a.k.a., the scaled dot-product attention, which is defined as:

$$\mathbf{H}^{\circ} = Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V} \quad (7)$$

where $\mathbf{H}^{\circ} \in \mathbb{R}^{n\times d}$ is the latent representation for all nodes after the self-attention. $\sqrt{d}$ is the scaling factor to smooth the row-wise SoftMax output and avoid extremely large values of the inner product, especially when the dimensionality is high. $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n\times d}$, respectively represent queries, keys and values obtained using linear projection:

$$\begin{aligned} \mathbf{Q} &= \mathbf{E}\mathbf{W}_Q, \\ \mathbf{K} &= \mathbf{E}\mathbf{W}_K, \\ \mathbf{V} &= \mathbf{E}\mathbf{W}_V, \end{aligned} \quad (8)$$

and $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d\times d}$ are corresponding trainable projection weight matrices for queries, keys and values. To be concise, we reformulate the self-attention module in Eq.(7) and Eq.(8) as the following:

$$\mathbf{H}^{\circ} = softmax\left(\frac{\mathbf{E}\mathbf{W}_Q \cdot (\mathbf{E}\mathbf{W}_K)^T}{\sqrt{d}}\right)\mathbf{E}\mathbf{W}_V \quad (9)$$

and each row $\mathbf{h}_i^{\circ} \in \mathbf{H}^{\circ}$ corresponds to the latent representation of the i-th node after the self-attention. Intuitively, for the i-th node feature, the self-attention encodes its interactions with all other nodes' features in a sequence into h_i by calculating a weighted sum (i.e., bit-wise interactions) of all other nodes' features. The embedding matrix $\mathbf{E} \in \mathbb{R}^{n\times d}$ here is the original embeddings of the nodes in a graph (i.e., $e_i = h_i, h_i$ is acquired from Eq.(3)).

The Rationale of Self-Attention. The self-attention described above takes the sequences (e.g., the sequences obtained from Figure 4) as the input and produce an attentive embedding for each node in a sequence, which is a weighted sum of all node features in a sequence and therefore captures the relevance between two nodes in that sequence. Link this back to the defined intra-class message passing, the strength of higher-order message passing passed between each pair of same typed nodes in a bipartite graph can be revealed and captured by the learned attentive weights. More importantly, the self-attention mechanism is computationally-efficient compared with existing sequence-based approaches.

Time Complexity of FBNE. Excluding the inter-class message passing operation that is standard in all GraphSAGE-based models, the computational cost of our model is mainly exerted by the intra-class self-attention units. Hence, for each training sample, the overall time complexity of these two components is $\mathcal{O}(l^2 d) + \mathcal{O}(\prod_1^x N_i)$, where l, N are user-specified constants denoting the sequence length and neighbourhood sampling size. x denotes the number of GCN layers, in our experiments, we found that only one layer is enough for bipartite graphs. Therefore the additional time complexity is acceptable and the proposed FBNE has linear time complexity w.r.t. the scale of the data.

3.3 Joint Multiple Bipartite Graph Embedding Learning

We have now described the general modelling process for bipartite graphs in FNBE. In this section, we jointly train the introduced bipartite embedding method on multiple bipartite graphs simultaneously to learn embeddings of multiple nodes in a more complex network. With out loss of generality, we use the graph described in Figure 1 as a typical social recommendation example, which contains multiple bipartite graphs that can be easily observed from e-commerce data.

For the user-item bipartite graph, the embeddings of user node i can be computed as the fowling:

$$\mathbf{u}_i = \sigma(\mathbf{W}(\mathbf{h}_i^u \oplus \mathbf{h}_i^\circ) + b) \tag{10}$$

where $\mathbf{h}_i^u$ denotes the user latent representation from Eq.(3), and $\mathbf{h}_i^\circ$ denotes the user latent representation obtained from Eq.(9), which is the latent user representation learned from modelling the sequences of folded user graph. Eq.(10) first concatenates these two user representations from inter- and intra-class domains, and then embeds the user's information into a projection space.

On the other side of a user-item bipartite graph, the item embedding $\mathbf{b}_i$ is computed analogously with the same weight matrix $\mathbf{W}$. To learn the weight matrix and embeddings of users/items, we formulate a link reconstruction problem on the original user-item bipartite graph. Specifically, we model the existence of a link $e_{ij} \in \mathcal{E}_{UB}$ by the following:

$$p(e_{ij} = 1) = \frac{1}{1 + exp(-\mathbf{u}_i \cdot \mathbf{b}_j^T)} \tag{11}$$

The loss function can be defined by minimizing the following negative log likelihood of the existence of a link between a user node and an item node.

$$\mathcal{L}_{\mathcal{UB}} = -\sum_{e_{ij} \in \mathcal{E}_{UB}} \log(p(e_{ij} = 1)) - \sum_{e_{ik} \notin \mathcal{E}_{UB}} \log(1 - p(e_{ik} = 1)) \tag{12}$$

Intuitively, minimizing this loss function will make two connected vertices in the bipartite graph also close with each other in the embedding space, which preserves the inter- and intra-class similarities of a bipartite graph as desired. Finally, based on the three bipartite graphs we have defined, i.e., user-item, item-attribute and item-category graphs, we embed the four bipartite graphs by minimizing the sum of all objective functions as follows (note that the item embeddings are shared across in different graphs):

$$\mathcal{O} = \lambda \mathcal{L}_{\mathcal{UB}} + \beta \mathcal{L}_{\mathcal{BA}} + \gamma \mathcal{L}_{\mathcal{BC}} \tag{13}$$

where λ, β, γ are wights for each bipartite graph, and

$$\mathcal{L}_{\mathcal{BA}} = -\sum_{e_{ij} \in \mathcal{E}_{BA}} \log(p(e_{ij} = 1)) - \sum_{e_{ik} \notin \mathcal{E}_{BA}} \log(1 - p(e_{ik} = 1)) \tag{14}$$

$$\mathcal{L}_{\mathcal{BC}} = -\sum_{e_{ij} \in \mathcal{E}_{BC}} \log(p(e_{ij} = 1)) - \sum_{e_{ik} \notin \mathcal{E}_{BC}} \log(1 - p(e_{ik} = 1)) \tag{15}$$

The objective Eq.(13) can be optimized by training all types of graphs simultaneously by merging all the edges in the four sets $\mathcal{E}_{UB}, \mathcal{E}_{UB}, \mathcal{E}_{UB}$ together, and then deploy edge sampling, which samples an edge for model updating in each step, with the sampling probability proportional to its weight. However, the graphs are heterogeneous in our model, the weights of the edges between different graphs are not comparable to each other. A more reasonable solution is to alternatively sample from the four sets of edges respectively which called joint training.

3.4 Inductive Social Embedding

So far, we have learned the embeddings of users, items, attributes and categories. We will now introduce how to represent social relationship in the shared latent space in an inductive manner. An embedding method is proposed to incorporates users social relationships into users' embeddings acquired from item perspective. More importantly, it supports inductive learning that learns from observed samples and generate embeddings for those unseen samples during training. By introducing the inductive learning paradigm, the proposed inductive social embedding method is therefore able to seamlessly solve the general cold-start problem in recommendation systems.

Similar to the introduced GCN embedding method for a bipartite graph in Section 3, we also adopt weight sharing in GCNs and define the embedding computation of a user based on his/her direct neighbour users. The idea of weight sharing in GCNs enables this local operation of convolving neighbours' representations to represent a target node to be easily shared and applied across all locations in the user to user graph. Same as the modelling process of user-item bipartite graph, we assume there exists a message passing through every edge between users. For each edge in the user-user graph, we model the following message passing from user u_i to u_j by the following form:

$$\eta_{ji} = \frac{1}{c_{ij}} \mathbf{W}^{uu} x_j^u \tag{16}$$

where c_{ij} is a normalization constant, which we choose to either be $|N(u_i)|$ or $\sqrt{|N(u_i)||N(u_j)|}$ (symmetric normalization), with $N(u_i)$ denoting the set of neighbors of user node i, and $N(u_j)$ denoting the set of neighbors of user node j. $\mathbf{W}^{uu}$ is the parameter matrix and x_j^u is the feature vector of user node j.

After the message passing step, we combine the incoming messages for node u_i by using an aggregation function $Agg(\cdot)$ that aggregates the neighbors $N(u_i)$ connected by edges between user i and the neighbours. Mathematically, we define the aggregation as follows:

$$\mathbf{h}_i^S = \sigma(\mathbf{W}' \cdot Agg(\{\eta_{ji}, j \in N(u_i)\})) \tag{17}$$

where $\sigma(\cdot)$ is an element-wise activation function such as $ReLU(\cdot) = max(0, \cdot)$. $\mathbf{W}'$ is a learnable parameter matrix. Then, the embeddings of user node i can be represented as the following in the user-user space.

$$\mathbf{u}_i^S = \sigma(\mathbf{W}^S \mathbf{h}_i^S) \tag{18}$$

Similar to Section 5.2, we model each social relationship between two users in the social graph. Specifically, we model the existence of a link $e_{ij} \in \mathcal{E}_{UU}$ by the following:

$$p(e_{ij} = 1) = \frac{1}{1 + exp(-\mathbf{u}_i^S \cdot (\mathbf{u}_j^S)^T)} \tag{19}$$

To optimize the user embedding in the user-user space, we use the following graph based loss:

$$\mathcal{L}_{\mathcal{UU}} = -\sum_{e_{ij} \in \mathcal{E}_{UU}} \log(p(e_{ij} = 1)) - \sum_{e_{ik} \notin \mathcal{E}_{UU}} \log(1 - p(e_{ik} = 1)) \tag{20}$$

With above, we have been able to learn a cold-start users' embeddings merely from its direct neighbours. That is to say, by only looking up one's neighbour users, we can inductively infer the embedding of a user.

3.5 Social Recommendation with FBNE

Once we have learned the embeddings of users and items, given a user u, we select top-k items with the highest scores that u has not visited before. More specifically, given a user u, we compute its ranking score as in Eq.(21), and select the k ones with the highest ranking scores as recommendations.

$$S(u) = \mathbf{u} \cdot \mathbf{b}^T \tag{21}$$

where $\mathbf{u}$ is the representation of u's preferences, $\mathbf{b}$ is the representation of an item, both $\mathbf{u}$ and $\mathbf{b}$ can be obtained by optimizing the objective in Eq.(13). The learnt representations also automatically capture the semantic content information of item b through the item-attribute and item-category graphs, as our FBNE model jointly learns the embedding of multiple relational networks in the same latent space.

Cold-Start Social Recommendation: As for cold-start users and items, our FBNE model can still learn their representations in the latent space based on the user-user graph, item-attributes and item-category graphs. Thus, both cold-start and normal users and items can be recommended together by the same ranking function, which distinguishes from other existing recommender models that use different functions to compute the scores for cold-start and normal items, separately.

4 Experimental Settings

In this section, we describe the datasets and experimental settings and outline the evaluation protocols of our proposed FBNE.

We conduct our experiments on two publicly available large-scale and real-life datasets, which are provided by Yelp Challenge[2] published in 2016 and MoiveLens[3]. Specifically, the Yelp dataset includes information about local business, user information, interactions between user and business (ratings, reviews), as well as friendship network among users. The original dataset contains the information in five states in the U.S, and we processed and extracted six (five individual state and one complete) large-scale heterogeneous social networks. Each network contains four different sub-networks, which are user-user, user-business, business-attribute, and business-category networks. Table 1. shows the detailed statistic of the extracted datasets.

4.1 Baseline Methods

We briefly introduce the baseline methods for comparison below. First of all, we choose the latest representative methods from three groups of methods for reccomendations including traditional recommender systems, traditional social recommender systems, and deep neural network based recommender systems as the common baselines for all ranking, classification, and regression tasks. Then, we further select several general state-of-the-art methods proposed for Graph (Bipartite Graph) Embedding as competitors.

- **PMF [32]**: Probabilistic Matrix Factorization utilizes user-item rating matrix only and models latent factors of users and items by Gaussian distributions.
- **SoRec [33]**: Social Recommendation performs collaborative factorization on the user-item rating matrix and user-user social relations matrix.
- **GCMC [25]**: This model defines message-passing and performs graph convolutions on user-item bipartite graphs for recommendations, and achieved state-of-the-art results.
- **GraphSAGE [18]**: GraphSAGE is a representative Graph Convolutional Networks style method for Graph Representation Learning. This versatile method has achieved state-of-the-art results on various tasks on graphs.
- **GraphRec [3]**: The GraphRec model is built on GraphSAGE, but it simultaneously aggregates both neighbourhood information in a user-item bipartite graph and social information in a user-user social network.
- **BiNE [34]**: BiNE is a state-of-the-art model dedicated for Bipartite Network Embedding.

4.2 Evaluation Metrics

Evaluation of Reccomendation Performance. To evaluate the ranking performance, we adopt the widely applied Hits Ratio at Rank k ($Hits@k$), Mean Reciprocal Rank (MRR) and Normalized Discounted Cumulative Gain at Rank k ($NDCG@k$) which are commonly used in information retrieval and recommender systems [35], [36], [37]. Specifically, We perform this task on Yelp dataset and further process this dataset by filtering out the nodes whose degree are less than 5. Then, we use 80-th percentile as the cut-off point so that the network linkage records before this point are used for training. In the training dataset, we choose the last 10% records as the validation data to tune the model parameters, including the dimension of latent feature vectors, margin, learning rate and the number of gradient steps. According to the above dividing strategies, we split the dataset D^+ into $D^+_{training}$, $D^+_{validate}$ and D^+_{test}. We summarise the detailed statistics of this dataset in Table 2. For each linkage information (a triple consists of two vertices connected by a link) i.e., $e_{ijr} \in D^+_{test}$:

- We randomly choose 1000 items with which vertex v_i has been never connected by link type r to replace v_j and form 1000 negative examples.
- We compute a score for e_{ijr} as well as the 1000 negative examples by calculating their inner product.
- We form a ranked list by ordering these 1001 examples according to their scores to v_i. Let $rank(e_{ijr})$ denote the position of e_{ijr} in the ranking list.
- We form a top-k prediction list by picking the k top ranked examples from the list. If $rank(e_{ijr}) <= k$, we have a hit. Otherwise, we have a miss.

The computation of Hits ratio proceeds as follows. We define $hit@k$ for a single test case as either 1, if the positive example e_{ijr} appears in the top-k results, or 0, if otherwise. The overall $Hits@k$ is defined by averaging over all test cases:

$$Hits@k = \frac{\#hit@k}{|D^+_{test}|}$$

where $\#hit@k$ denotes the number of hits in the test set, and $|D^+_{test}|$ is the number of all test cases. A good predictor

2. https://www.yelp.com/dataset/challenge
3. http://grouplens.org/datasets/movielens/

TABLE 1: Datasets statistics

	No. of Edges				No. of Nodes			
Datasets	U2U	U2B	B2A	B2C	B	U	A	C
Yelp(All)	29,271,479	4,153,150	605,231	527,229	14,4072	1,029,432	81	1,191
Yelp(WI)	57,593	88,778	14,986	18,479	3,899	25,773	81	678
Yelp(EDH)	25,695	44,631	12,676	11,972	3,539	8,371	72	456
MoiveLens	10,000,054				10,677	69,878	-	-

should achieve higher $\#hit@k$. We also use $NDCG@k$ to further evaluate whether if the model can rank the ground truth as highly as possible:

$$NDCG@k = \frac{\sum e \in D^{+}_{test} \sum^{k}_{rank=1} \frac{rel_{e,rank}}{log_2(rank+1)}}{|D^{+}_{test}|}$$

For the computation of $NDCG@k$, we set $rel_{e,rank} = 1$ if the item ranked at r is the ground truth, otherwise $rel_{e,rank} = 0$. For k in both $Hits@k$ and $NDCG@k$, we adopt the popular setting of 5, 10, 20 for presentation.

Evaluation of Rating Prediction. Because each pair of user-item embeddings can be used to generate a score, we further test our FBNE model in an additional regression task, i.e., rating prediction which is a classic problem in recommendation. In order to evaluate the capability of the proposed model in terms of rating prediction, we evaluate the regression performance with two popular metrics to evaluate the predictive accuracy, namely Mean Absolute Error (MAE) and Root Mean Square Error (RMSE), which are popular among relevant research communities [38], [39], [40] . Smaller values of MAE and RMSE indicate better predictive accuracy. Note that small improvement in RMSE or MAE terms can have a significant impact on the quality of the top-few recommendations [8].

4.3 Parameter Settings

There are several general parameters in the proposed model. We use the grid search algorithm to obtain the optimal model hyper-parameters setup on the validation dataset. For learning rate, we set it to 0.001. The weights $\{\alpha, \beta, \gamma\}$ for different bipartite graphs in Eq.(13) are set to $\{0.8,0.1,0.1\}$. We also set default embedding size at 64, batch size at 128.

5 Experimental Results and Analysis

Following the settings in Section 4, we conduct experiments to showcase the advantage of FBNE in terms of both effectiveness and efficiency. In particular, we aim to answer the following research questions (RQs) via experiments and analytics:

- **RQ1:** How effectively our proposed approach can perform recommendation compared with state-of-the-art methods.
- **RQ2:** How effectively our proposed approach can perform inductive cold-start social recommendation.
- **RQ3:** How the hyper-parameters affect the performance of our approach in different prediction tasks.
- **RQ4:** How our proposed approach benefits from each component of the proposed model structure.

5.1 Recommendation Performance (RQ1)

The results of the recommendation task are reported in Table 2. Note that higher $Hits@k$ and $NDCG@k$ values imply better prediction performance. We have the following observations. First of all, obviously, on both EDH and WI datasets, FBNE significantly and consistently outperforms all existing social recommendation models and network embedding models with $K \in \{5, 10, 20\}$. This validates the effectiveness of our FBNE solution, especially our proposed novel techniques that exploits and integrates the user-user higher-order interaction data and the user-item bipartite network data to overcome the sparsity issue of the user-item interactions when learning user embeddings for recommendation. Another observation is that all graph convolutional network based models (i.e., GraphSAGE, GCMC, GraphRec) outperform the traditional social recommendation models, which shows that graph convolutional operations can learn better representations on graph structured data than matrix factorization based model on the user-item data. Moreover, the network embedding method for bipartite graph (BiNE) did not perform as good as dedicated methods for social recommendation and graph convolutional networks based methods. This is because BiNE is exclusively designed for general bipartite network embedding and puts a significant focus on preserving network distribution when learning node representations in a user-item bipartite graph. Nevertheless, BiNE still performs better than traditional matrix factorization models such as PMF.

5.2 Rating Prediction Performance (RQ1)

Table 3 reveals all models performance achieved in the regression task (rating prediction) on MoiveLens and Yelp datasets. For both MAE and RMSE metrics, the lower the better. As we can see from the results, FBNE yields significant improvements on the regression accuracy over all baselines, which shows that FBNE is effective and can be generalized for rating prediction tasks by only modelling the user-item interactions. Furthermore, though showing competitive regression results, the baseline GraphRec falls short in higher-order node dependency modelling. Compared to GraphRec, our proposed FBNE further incorporates higher-order dependencies among users into node representation learning, which boosts the performance of the rating regression task. Apart from that, we notice that graph convolutional networks based methods consistently outperform other methods, which shows that graph convolutions are effective of learning graph structured data. Moreover, there is no obvious winner among graph convolutional network based methods, some of them work well in one dataset but perform poorly in other dataset due to unique characteristics of the datasets. Last but not least, we can not see any advantages of social recommendation

TABLE 2: Recommendation(ranking) results. Numbers in bold face are the best results for corresponding metrics.

Method	EDH state						WI state					
	Hits@k			NDCG@k			Hits@k			NDCG@k		
	K=5	K=10	K=20	K=5	K=10	K=20	K=5	K=10	K=20	K=5	K=10	K=20
PMF [32]	0.017	0.030	0.055	0.012	0.016	0.023	0.054	0.081	0.128	0.030	0.047	0.064
SoRec [33]	0.054	0.081	0.116	0.030	0.074	0.054	0.092	0.127	0.183	0.069	0.070	0.093
BiNE [34]	0.042	0.065	0.111	0.028	0.036	0.049	0.089	0.103	0.154	0.064	0.066	0.080
GraphSAGE [18]	0.056	0.079	0.126	0.053	0.068	0.079	0.084	0.119	0.169	0.061	0.075	0.087
GCMC [25]	0.072	0.078	0.087	0.070	0.072	0.075	0.110	0.136	0.157	0.070	0.079	0.082
GraphRec [3]	0.076	0.082	0.133	0.074	0.078	0.081	0.103	0.147	0.201	0.069	0.083	0.131
FBNE	**0.107**	**0.150**	**0.213**	**0.093**	**0.118**	**0.142**	**0.133**	**0.171**	**0.253**	**0.077**	**0.089**	**0.165**

based methods as there is no social relations involved on the MoiveLens dataset; however, our proposed FNBE can still achieve the best performance compared against other methods, showing that the higher-order proximity captured from user-user implicit relations are beneficial for user-item rating regression.

TABLE 3: Rating prediction(rating regression) results. Numbers in bold face are the best results for corresponding metrics.

Method	MoiveLens		Yelp-EDH	
	MAE	RMSE	MAE	RMSE
PMF [32]	0.9137	1.1013	1.3471	1.7988
BiNE [34]	0.7653	0.9275	0.9069	1.0457
GraphSAGE [18]	0.7381	0.9132	0.7199	0.9459
GCMC [25]	0.7296	0.9104	0.8256	0.8232
GraphRec [3]	0.7538	0.9226	0.5791	0.7718
FBNE	**0.7017**	**0.9030**	**0.5653**	**0.7594**

5.3 Cold Start Recommendation Performance (RQ2)

Our proposed approach is capable of conducting inductive learning. Inductive learning means that our proposed approach first learns embedding for a portion of nodes, and then infer the embeddings for those nodes are not seen during the training. Specifically, for a user, we can infer its embedding merely from the embeddings of its direct neighbours; for an item, we can infer its embedding from its directly connected attributes or/and categories. We report the both the top-k recommendation and rating prediction performance of our proposed approach on cold start users and items.

TABLE 4: Inductive prediction performance for cold-start users and items on Yelp (all) dataset. Each row represents performance w.r.t. different portion of training nodes.

	RMSE	MAE	Hit@10	NDCG@10
30%	1.0540	0.9167	0.0125	0.0089
40%	1.0338	0.9105	0.0170	0.0110
50%	1.0089	0.9092	0.0256	0.0113
60%	1.0069	0.8989	0.0278	0.0120
70%	0.9955	0.8923	0.0287	0.0124
80%	0.9949	0.8902	0.0289	0.0126
90%	0.9895	0.8896	0.0289	0.0137

Table 4 reveals the cold-start recommendation and rating prediction results of our proposed method on the full Yelp dataset. The results are reported by averaging the results of running experiments 10 times. From the table, we can see that when we continuously adding more training data, the performance of both rating prediction (revealed by MAE, RMSE) and recommendation (HR@10, NDCG@10) are improving consistently, showing that our proposed is effective in inductive training.

5.4 Impact of Hyper-parameters (RQ3)

We investigate how the performance of our proposed approach varies w.r.t. different key hyper-parameter setups, including the latent dimension d, the maximum intra-class node sequence length l, and the number of negative samples m. For each test, based on the standard setting d = 64, l = 5, m = 5, we vary the value of one hyper-parameter while keeping the others unchanged, and record the new recommendation result achieved. To show the performance differences, we demonstrate HR@10 for recommendation, and RMSE for regression. Figure 5 lays out the results with different parameter settings.

Impact of embedding dimension d. The value of the latent dimension d is examined in {8, 16, 32, 64, 128}. As an important hyper-parameter for learning embeddings, the latent dimension is apparently associated with the models expressiveness. In general, FBNE benefits from a relatively larger d for all types of tasks, but the performance improvement tend to become less significant when d reaches a certain scale (64 in our case). It is worth mentioning that with d = 16, FBNE still outperforms nearly all the baselines in the rating regression tasks, which further proves the effectiveness of our proposed model.

Impact of node sequence length l. As one of the main component of FBNE is the intra-class node sequential modelling, we examine how different node sequences length affects the performance. From Figure 5, we can conclude that FBNE behaves consistently on different datasets when the sequence length l is adjusted in {2, 5, 10, 15, 20, 25}. When node sequence is short (i.e., l = 2), a significant performance decrease is witnessed. While when increasing the length, it becomes better than other methods in both recommendation and rating prediction. However, when continuously increasing it, the prediction results become stable, and there is no further obvious performance gain. More interestingly, we found that Moivelens dataset requires longer node sequences ($l = 10$) to achieve stable rating prediction results, which is because the number of users in moivelens dataset is far larger than that in Yelp dataset, and therefore it needs longer node sequences to model the dependencies between users.

Impact of negative samples m. We study the performance of FBNE w.r.t. the number of negative examples m. As can be concluded from Figure 5, we observe that when the number of negative examples is larger than 5,

the performance becomes very stable. Besides, when dealing with a larger number of negative samples, the training will take longer time. Therefore, for each positive example, we do not need to sample excessive negative examples and just need to sample a few (5 in our case), which ensures the training efficiency of our FBNE model.

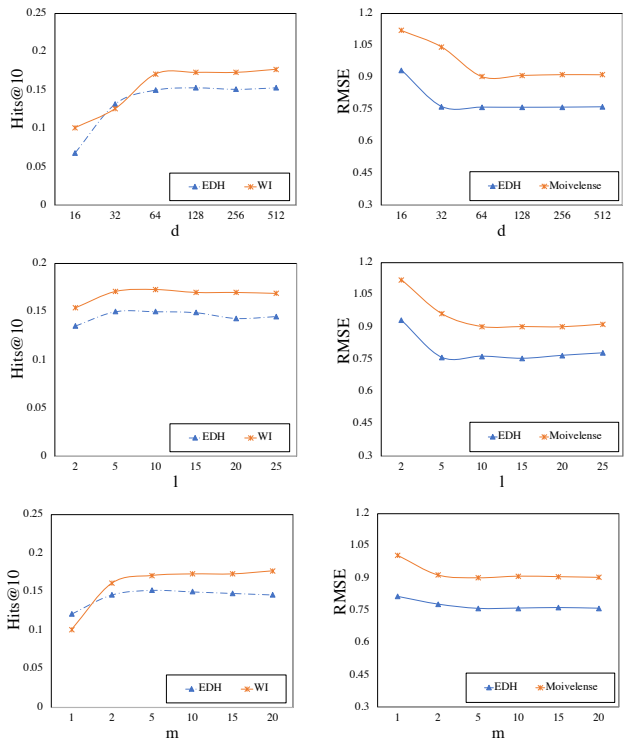


Fig. 5: Hyper-parameter sensitivity analysis w.r.t d, l, m

5.5 Importance of Key Components (RQ4)

To better understand the performance gain from the major components of our proposed method, we conduct ablation test on different degraded versions of FBNE. Each variant removes one key component from the model, and the corresponding results of three tasks are reported. Table 5 summarizes the prediction outcomes in different tasks. Similar to previous experiments, HR@10 and MAE are used. In what follows, we introduce the variants and analyze their effect respectively.

TABLE 5: Ablation test on model components. Numbers in bold face are the best results for corresponding metrics.

Method	HR@10		MAE	
	EDH	WI	Moivelens	EDH
Default	**0.1501**	**0.1713**	**0.7017**	**0.5653**
Remove CS	0.1421	0.1681	0.7238	0.5687
Remove Intra	0.1012	0.1224	0.7537	0.5712
Remove $\mathcal{L}_{\mathcal{BC}}$	0.1350	0.1584	-	0.5680
Remove $\mathcal{L}_{\mathcal{BA}}$	0.1461	**0.1712**	-	0.5755

Remove centrality-based sampling (Remove CS). The centrality-based sampling strategy allows graph convolutions sample neighbours according to an importance measure. After removing it, a subtle performance drop is consistently observed in both recommendation and rating prediction tasks. Although the performance drop is not quite significant, the experiment shows that centrality-based sampling is beneficial.

Remove Inter-class message passing (Remove Intra). As a core contribution of our proposed approach, the Intra-class message passing is able to capture the higher-order dependencies between same type of nodes by using a self-attention sequential modelling paradigm. After removing it, a noticeable performance drop has been observed in all tasks. The results verify that this sequential modelling in same types of nodes play a significant role in improving the expressiveness of learned embeddings.

Remove Business-Category Graph (Remove $\mathcal{L}_{\mathcal{BC}}$). We also consider to remove the loss term Eq.(14) of business-category bipartite graph to investigate whether this additional bipartite graph modelling can have a positive impact on the performance. After removing $\mathcal{L}_{\mathcal{BC}}$ from Eq.(13), we can see that there is an obvious performance drop in recommendation task and no obvious drop in rating prediction task. The results suggest that modelling additional business-category bipartite network is helpful in leaning embeddings of a business. Moreover, the main reason of increased MAE in rating prediction is that rating prediction task puts more weights on user-business interactions, and the category information probably will not affect users' opinion on specific businesses. For example, a user likes Japanese Food does not mean that he will give higher ratings for every Japanese restaurant.

Remove Business-Attribute Graph (Remove $\mathcal{L}_{\mathcal{BA}}$). Similar to removing $\mathcal{L}_{\mathcal{BC}}$, we also test how $\mathcal{L}_{\mathcal{BA}}$ is related to the model performance. Interestingly, after removing $\mathcal{L}_{\mathcal{BA}}$, we observe that there is a subtle performance drop in recommendation task on EDH dataset, while for WI dataset, the performance change is not obvious. Moreover, the MAE score for rating prediction on EDH dataset drops marginally. We analyse the results and found that a reasonable explanation is that, different from category information, attribute information of a business will affect users' impression significantly. For example, if a restaurant has enough parking spaces, people usually will like it with a high possibility.

6 Related Work

Our work is related to representation learning methods on bipartite networks and social recommendation systems by using network embedding techniques.

6.1 Bipartite Network Embedding

Network embedding (NE) is a recent hot topic with the focus of learning low dimensional latent vector representation for vertices in a network (usually for homogenous networks) [18], [20], [21], [22], [41]. As a special and ubiquitous graph data structure, bipartite networks consist of two different sets of vertices and have been mined for many applications such as recommendations. However, general NE models may not suitable for bipartite graphs embedding because they only consider one type of vertices. Although some heterogenous network embedding methods [22], [42] can be applied to bipartite network which can be seen as a special type of heterogeneous networks, they are not tailored for learning on bipartite networks. To build a dedicated embedding model for bipartite graphs, Latent factor model (LFM), which has been widely investigated in the

field of recommender systems and semantic analysis, is the most representative model. And a typical implementation of LFM is based on matrix factorization [43], [44]. Recent advances utilize deep learning methods to learn vertex embeddings on the user-item network for recommendation [45]. In addition to above mentioned non-deep embedding techniques, GCN based deep learning models have shown their superiority over most of existing methods in terms of downstream tasks, becoming the state-of-the-art methods. More recently, graph convolutions on matrix completion (GCMC [25]) was proposed, which learns users and items embeddings through defining differentiable message passing on the user-item bipartite graph. Nevertheless, these methods only model the explicit relations in bipartite network, which can be improved by incorporating implicit relations as shown in [9], [34], [46] .

6.2 Social Reccomendation

With the prevalence of online social platforms, exploiting social relations for boosting recommendations has been attracting significant attention in recent years. The research line develops from the study of social influences [7] and homophily assumption [6]. Based on these studies, an earlier attempt is SoRec [33], which proposed a co-factorization model that uses a shared latent user representation factorized from user-item rating matrix and social relations. By treating the social neighbors preferences as the auxiliary implicit feedbacks of an active user, TrustSVD is proposed to incorporate the trust influence from social neighbors on top of SVD++ [47]. Similar related work also includes SoDimRec [14] and TrustMF [48]. A comprehensive review of previous work on social recommendation can be found in [1].

Recently, with the fast development of deep neural networks, several deep models have been propsed to enhance social recommender systems, such as DLMF [49] and DeepSoR [2]. DLMF utilizes deep auto-encoder to learn representation for initializing an existing matrix factorization. DeepSoR utilizes deep neural networks to capture nonlinear user representations in social relations and integrate them into a probabilistic matrix factorization model for prediction. GraphRec [3] proposed a GCN-like framework that aggregates both user-item interactions and social relationships to learn representations of users and items.

In summary, all these social recommendation based models have shown the superior performance with the social network modeling. Nevertheless, current models were based on leveraging the social network structure (e.g., social regularization or combining social neighbors preferences as auxiliary feedbacks), which is far from sufficient as the available social relationship are usually noisy and sparse. Instead of only considering the social information, our work differs from these works that we model users latent preferences with additional higher-order implicit message passing among users through a sequential modelling paradigm, which is capable of encoding the preference influences and dependencies between users.

7 Conclusion

In complementary to the limited available social relations in the context of social recommendation, this paper further explores the implicit higher-order user-user relations though folding a user-item bipartite graph to boost the performance of current social recommendation. Technically, this paper also proposes a novel embedding method FBNE for general bipartite graphs. By jointly modelling several extra related bipartite graphs, this paper also improves the recommendation performance in the cold-start setting. The main findings of this paper suggests that apart from implicit social relations in social recommendation, mining higher-order implicit social relations also yields meaningful improvements for recommendation. Future works may include the exploration of the higher-order sequences generation and modelling (e.g., by regulating meta-paths to generate node sequences).

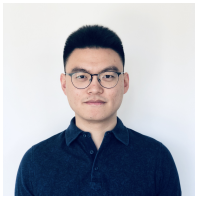
References

[1] J. Tang, X. Hu, and H. Liu, "Social recommendation: a review," *Social Network Analysis and Mining*, 2013.
[2] W. Fan, Q. Li, and M. Cheng, "Deep modeling of social relations for recommendation," in *AAAI*, 2018.
[3] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *WWW*. ACM, 2019, pp. 417–426.
[4] X. Wang, W. Lu, M. Ester, C. Wang, and C. Chen, "Social recommendation with strong and weak ties," in *CIKM*. ACM, 2016.
[5] L. Wu, P. Sun, R. Hong, Y. Fu, X. Wang, and M. Wang, "Socialgcn: An efficient graph convolutional network based model for social recommendation," 2018.
[6] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual review of sociology*, vol. 27, no. 1, pp. 415–444, 2001.
[7] P. V. Marsden and N. E. Friedkin, "Network studies of social influence," *Sociological Methods & Research*, 1993.
[8] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *SIGKDD*. ACM, 2008.
[9] L. Yu, C. Zhang, S. Pei, G. Sun, and X. Zhang, "Walkranker: A unified pairwise ranking model with multiple relations for item recommendation," in *AAAI*, 2018.
[10] IBM. (2012) Ibms black friday report says twitter delivered 0 percent of referral traffic and facebook sent just 0.68 percent. [Online]. Available: https://strme.wordpress.com/2012/11/27/ibms-black-friday-report-says-twitter-delivered-0-percent-of-referral-traffic-and-facebook-sent-just-0-68-percent/
[11] J. Leskovec, L. A. Adamic, and B. A. Huberman, "The dynamics of viral marketing," *ACM Transactions on the Web (TWEB)*, vol. 1, no. 1, p. 5, 2007.
[12] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *SIGKDD*. ACM, 2011, pp. 1082–1090.
[13] H. Gao, J. Tang, and H. Liu, "Exploring social-historical ties on location-based social networks," in *AAAI*, 2012.
[14] N. Agarwal, H. Liu, L. Tang, and P. S. Yu, "Identifying the influential bloggers in a community," in *WSDM*. ACM, 2008.
[15] J. Tang, H. Gao, X. Hu, and H. Liu, "Exploiting homophily effect for trust prediction," in *WSDM*. ACM, 2013, pp. 53–62.
[16] R. J. Mooney, P. N. Bennett, and L. Roy, "Book recommending using text categorization with extracted information," in *Proc. Recommender Systems Papers from 1998 Workshop, Technical Report WS-98-08*, 1998.
[17] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv:1609.02907*, 2016.
[18] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NIPS*, 2017.
[19] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.
[20] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *SIGKDD*. ACM, 2016.
[21] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *SIGKDD*. ACM, 2014.

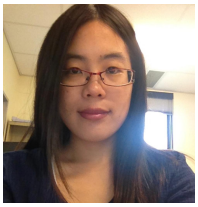
[22] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *SIGKDD*. ACM, 2017, pp. 135–144.
[23] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018.
[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017.
[25] R. van den Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," 2018.
[26] H. Chen, H. Yin, T. Chen, Q. V. H. Nguyen, W.-C. Peng, and X. Li, "Exploiting centrality information with graph convolutions for network representation learning," in *ICDE*. IEEE, 2019.
[27] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv:1312.6203*, 2013.
[28] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *NIPS*, 2016, pp. 3844–3852.
[29] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 1480–1489.
[30] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.
[31] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013.
[32] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *NIPS*, 2008.
[33] H. Ma, H. Yang, M. R. Lyu, and I. King, "Sorec: social recommendation using probabilistic matrix factorization," in *CIKM*, 2008.
[34] M. Gao, L. Chen, X. He, and A. Zhou, "Bine: Bipartite network embedding," in *SIGIR*. ACM, 2018.
[35] H. Yin, X. Zhou, B. Cui, H. Wang, K. Zheng, and Q. V. H. Nguyen, "Adapting to user interest drift for poi recommendation," *TKDE*, 2016.
[36] C. Yang, L. Bai, C. Zhang, Q. Yuan, and J. Han, "Bridging collaborative filtering and semi-supervised learning: A neural approach for poi recommendation," in *SIGKDD*. ACM, 2017, pp. 1245–1254.
[37] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-n recommendation tasks," in *RecSys*. ACM, 2010.
[38] T. Chen, H. Yin, H. Chen, L. Wu, H. Wang, X. Zhou, and X. Li, "Tada: trend alignment with dual-attention multi-task recurrent neural networks for sales prediction," in *ICDM*. IEEE, 2018.
[39] C. Cheng, F. Xia, T. Zhang, I. King, and M. R. Lyu, "Gradient boosting factorization machines," in *RecSys*. ACM, 2014.
[40] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *SIGIR*. ACM, 2018.
[41] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *WWW*. International World Wide Web Conferences Steering Committee, 2015.
[42] H. Chen, H. Yin, W. Wang, H. Wang, Q. V. H. Nguyen, and X. Li, "Pme: projected metric embedding on heterogeneous networks for link prediction," in *KDD*. ACM, 2018, pp. 1177–1186.
[43] H. Zhang, F. Shen, W. Liu, X. He, H. Luan, and T.-S. Chua, "Discrete collaborative filtering," in *SIGIR*. ACM, 2016.
[44] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua, "Fast matrix factorization for online recommendation with implicit feedback," in *SIGIR*. ACM, 2016, pp. 549–558.
[45] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *WWW*, 2017.
[46] M. Jiang, P. Cui, N. J. Yuan, X. Xie, and S. Yang, "Little is much: Bridging cross-platform behaviors through overlapped crowds," in *AAAI*, 2016.
[47] G. Guo, J. Zhang, and N. Yorke-Smith, "Trustsvd: collaborative filtering with both the explicit and implicit influence of user trust and of item ratings," in *AAAI*, 2015.
[48] B. Yang, Y. Lei, J. Liu, and W. Li, "Social collaborative filtering by trust," *TPAMI*, vol. 39, no. 8, pp. 1633–1647, 2016.
[49] S. Deng, L. Huang, G. Xu, X. Wu, and Z. Wu, "On deep learning for trust-aware recommendations in social networks," *IEEE TNNLS*, vol. 28, no. 5, pp. 1164–1177, 2016.

Hongxu Chen is currently a computer science Ph.D. candidate at the School of Information Technology and Electrical Engineering, The University of Queensland, Australia. His research interests include Network Science, Graph Theory, Information Network Representation Learning, Recommender Systems, Social User Modelling and Social Network Mining.

Hongzhi Yin received the PhD degree in computer science from Peking University, in 2014. He is a senior lecturer with the University of Queensland. He received the Australia Research Council Discovery Early-Career Researcher Award, in 2015. His research interests include recommendation system, user profiling, topic models, deep learning, social media mining, and location-based services.



Tong Chen is currently a computer science Ph.D. candidate at the School of Information Technology and Electrical Engineering, The University of Queensland. His research interests include data mining, machine learning, and artificial intelligence. More specifically, he is currently conducting research on commercial time series analysis, sequential recommender systems and user behavior modelling.



Weiqing Wang received the B.S. and M.S. degrees from Nanjing University, Nanjing, China, in 2010 and 2013, respectively, and the Ph.D. degree from the University of Queensland, Brisbane, QLD, Australia. She was a Post-Doctoral Researcher at the University of Queensland. She is currently a Lecturer with Monash University, Melbourne, Australia. Her research interests include data mining, machine learning, and their applications in recommender system.

Xue Li received the MSc and PhD degrees from The University of Queensland (UQ) and the Queensland University of Technology, in 1990 and 1997 respectively. He is currently a professor in the Dalian Neusoft University of Information, China and a professor as leave without pay in the University of Queensland, Australia. His major areas of research interests and expertise include data mining, machine learning, and intelligent web information systems. He is a member of the ACM, the IEEE, and the SIGKDD.

Xia Hu received the BS and MS degrees in computer science from Beihang University, China, and the PhD degree in computer science and engineering from Arizona State University. He is currently an assistant professor in the Department of Computer Science and Engineering, Texas A&M University. His research interests include data mining, social network analysis, machine learning. He has published more than 100 papers in major venues, including WWW, SIGIR, KDD, WSDM, IJCAI, AAAI, CIKM, SDM, etc. He has served on program committees for several major conferences such as IJCAI, SDM and ICWSM, and reviewed for multiple journals, including the IEEE Transactions on Knowledge and Data Engineering, the ACM Transactions on Information Systems, and Neurocomputing. His research attracts wide range of external government and industry sponsors, including US National Science Foundation (NSF), ONR, AFOSR, Yahoo!, and Microsoft. He is a member of the IEEE.