Front.Comput.Sci.
DOI

RESEARCH ARTICLE

Graph Convolution Machine for Context-aware Recommender System

Jiancan Wu[1], **Xiangnan He**(✉)[1], **Xiang Wang**[2], **Qifan Wang**[3], **Weijian Chen**[1], **Jianxun Lian**[4], **Xing Xie**[4]

1 University of Science and Technology of China, Hefei, China
2 National University of Singapore
3 Google Research
4 Microsoft Research Asia

Abstract The latest advance in recommendation shows that better user and item representations can be learned via performing graph convolutions on the user-item interaction graph. However, such finding is mostly restricted to the collaborative filtering (CF) scenario, where the interaction contexts are not available. In this work, we extend the advantages of graph convolutions to context-aware recommender system (CARS, which represents a generic type of models that can handle various side information). We propose *Graph Convolution Machine* (GCM), an end-to-end framework that consists of three components: an encoder, graph convolution (GC) layers, and a decoder. The encoder projects users, items, and contexts into embedding vectors, which are passed to the GC layers that refine user and item embeddings with context-aware graph convolutions on user-item graph. The decoder digests the refined embeddings to output the prediction score by considering the interactions among user, item, and context embeddings. We conduct experiments on three real-world datasets from Yelp and Amazon, validating the effectiveness of GCM and the benefits of performing graph convolutions for CARS.

Keywords Context-Aware Recommender Systems, Graph Convolution

Received month dd, yyyy; accepted month dd, yyyy

E-mail: *hexn@ustc.edu.cn*

1 Introduction

Recommendation has become a pervasive service in today's Web, serving as an important tool to alleviate information overload and improve user experience. The key data source for building a recommendation service is user-item interactions, *e.g.,* clicks and purchases, which spawn wide research efforts on collaborative filtering (CF) [10, 21, 32] that leverage the interaction data only to predict user preference. Recently, inspired by the success of graph neural networks (GNNs) [13,27], researchers have attempted to employ GNNs on recommendation in which CF signals are exhibited as high-order connectivity [32,33,35,42]. While CF provides a universal solution for recommendation, it falls short in utilizing the side information of interaction contexts. In many scenarios, the current contexts could have a strong impact on user choice. For example, in restaurant recommendation, the current time and location can effectively filter out unsuitable candidates; in E-commerce, the click behaviors in recent sessions provide strong signal on user next purchase. As such, it is important to develop context-aware recommender system (CARS) that can effectively integrate contexts (and possibly other side information like user profiles and item attributes) into user preference prediction [24].

Inspired by the matrix completion view of CF, early research naturally extended the problem of CARS to tensor

completion [11], which however suffers from high complexity. Later on, Rendle proposed factorization machine (FM) [20], which to the first time addressed CARS from the view of standard supervised learning. Specifically, it converts all information related to an interaction to a feature vector via multi-hot encoding, modeling the second-order feature interactions to predict the interaction label. Due to its generality and effectiveness, FM soon becomes a prevalent solution for CARS and is followed by many work. For example, in the era of deep learning, Wide&Deep [3] and Deep Crossing [23] replaced the second-order interaction modeling with a neural network for implicit interaction modeling; recently, Neural FM [8], Attentional FM [37], xDeepFM [17], and Convolutional FM [38] extended FM with various kinds of neural networks to enhance its expressiveness.

Summarizing existing CARS models, we can find a common drawback: they follow the standard supervised learning scheme that ignores the relationship among data instances. This may limit the model's effectiveness in capturing the CF effect, since it needs to consider multiple interactions simultaneously to recognize the CF patterns. An evidence is from the neural graph collaborative filtering (NGCF) work [32], which demonstrates that connecting the interactions in the predictive model significantly improves the embedding quality for CF. Since in CARS user-item interactions still play an important role by reflecting user preference, it is reasonable to believe that properly modeling the relationship among interactions can improve the model quality. Moreover, the recent neural network-based methods like xDeepFM [17] and Convolutional FM [38] suffer from low efficiency in online serving, since each candidate item needs be scored separately with the deep model architecture that models complex feature interactions, which could be very time-consuming.

In this work, we aim to propose new CARS model by addressing the above-mentioned limitations. Firstly, we cast the data in CARS as an attributed user-item graph, where the side information of users and items are represented as node features, and the contexts are represented as edge features (Figure 1). Secondly, we propose an end-to-end model that consists of three components: an encoder, graph convolution (GC) layers, and a decoder (Figure 2). The encoder projects users, items, and contexts into embedding vectors; the GC layers then exploit the interactions to refine the embeddings via performing graph convolutions; lastly, the decoder models the interactions among embeddings via FM to output the prediction score. After the model is trained, the refined embeddings by GC layers can be pre-computed before serving. As such, the time complexity of online serving is the same as FM, being much more efficient than the recent neural network methods.

We summarize the contributions of this work as follows:

- We highlight the limitation of the mainstream supervised learning schemes and the necessity of exploiting the relationship among data instances in the predictive model of CARS.
- We propose a new model named Graph Convolution Machine (GCM), unifying the strengths of graph convolution network and factorization machine for CARS.
- We conduct extensive experiments on three real-world datasets which demonstrate the effectiveness and efficiency of GCM.

2 Related Work

2.1 Context-aware Recommendation

Extensive studies on context-aware recommender system (CARS) [8, 17, 20] have been conducted and achieved great success. Learning informative representations, based on user-item interactions (*e.g.,* clicks, purchases) and contextual features (*e.g.,* location, time, last purchase), has been a central theme of research on CARS. Towards this end, modeling interactions among different feature is showing promise. Early, factorization machine (FM) [20] embeds each feature into a vector representation, and utilizes inner product to capture their pairwise relationships (*e.g.,* the second-order feature interactions). Due to its generality and effectiveness, FM becomes a prevalent solution for CARS. Many works resort to this paradigm, such as FFM [22]. Recent works [3,6,8,17,37] leverage deep neural networks to model higher-order feature interactions, so as to generate better representations and enhance recommendation performance. For example, NFM [8] proposes a bilinear interaction operation which uses a sum pooling over the pair-wise dot-product of feature vectors; AFM [37] learns the importance of each feature interaction via the attention mechanism; xDeepFM [17] extends the Cross Network [28] to the Compressed Interaction Network (CIN) which models high-order interactions explicitly at vector-wise level; while Convolutional FM [38] models second-order interaction with outer product, forming an interaction cube, then applying 3D convolution to learn high-order interactions.

Despite effectiveness, we argue that present works treat user interactions as isolated data instances, while forgoing their relationships (*e.g.,* user behaviors happened at the same

time and location are highly likely to reflect user preferences). This would easily lead to suboptimal representations and limit the performance. We hence aim to explore relationships among user behaviors in this work.

2.2 Graph Neural Networks for Recommendation

Another relevant research line is to leverage graph neural networks (GNNs) for recommendation. In particular, GNN models [7, 13, 27] exploit graph structure to guide the representation learning. The basic idea is the embedding propagation mechanism, which aggregates the embeddings of neighbors to update the target node's embedding. By recursively performing such propagations, the information from multi-hop neighbors is encoded into the representation of target node. GNN models has been widely used in many fundamental tasks due to their strong representation ability, spanning from node classification [5], link prediction [41], to graph classification [39], and achieved remarkable improvements.

Inspired by their success, researchers have attempts to employ GNNs on recommendation. Recent works on collaborative filtering (CF), such as NGCF [32], GC-MC [26], SpectralCF [42] and PinSage [40], reorganize historical user behaviors in the form of a user-item bipartite graph, exhibit CF signals as high-order connectivity, and encode such signals into representations. Another recommendation task — CTR (click through rate) prediction — has also witnesses the success of GNN models. Fi-GNN [16] takes multi-field features into consideration by constructing feature graph for each instance and converting the task of modeling feature interactions among fields into modeling node interactions on the feature graph. GIN [15] models implicit user intention by the multi-layered intention diffusion and aggregation on the co-occurrence click relationship graph. [29] builds the multi-relational item graph and applies GNN to capture complex transition relations between items in user bahavior sequences. Moreover, GNN models have also been employed on other recommendation tasks, including social recommendation [4, 35], sequential recommendation [25, 36], and knowledge-aware recommendation [2, 30]. As such, aggregating useful information from multi-hop neighbors is able to achieve better expressiveness, than single ID embeddings. Hence, it is reasonable to believe that graph learning is a promising solution to properly model the relationships among interactions.

3 Problem Definition

We divide the data used for CARS into four types: users, items, contexts, and interactions. Following [22], we define context as the information that is associated with an interaction, *e.g.,* the current location, time, previous click, etc. Figure 1 illustrates the data in CARS, where the main data is the user-item-context interaction tensor. In the sparse tensor, each nonzero entry (u, i, c) denotes that the user u has interacted with the item i under the context c; we give such entries a label of 1, *i.e.,* $y_{uic} = 1$. Each u, i, c is respectively associated with a multi-hot feature vector $\mathbf{u}$, $\mathbf{i}$, and $\mathbf{c}$, which contain the features that describe the user, item, and context. For example, $\mathbf{u}$ includes static user profiles like gender and interested tags, $\mathbf{i}$ includes static item attributes like category and price, and $\mathbf{c}$ includes dynamic contexts like the current location of the user and the time.

Given such data, we convert it to the form of attributed user-item bipartite graph that has the same representation power. Specifically, each vertex represents a user or an item, and each edge represents the interaction between the connected user and item. Each vertex or edge is associated with a feature vector $\mathbf{u}$, $\mathbf{i}$, or $\mathbf{c}$. Note that there may exist multiple edges between a user-item pair, since a user may interact with the same item multiple times under different contexts. We denote all edges in the graph as the set $\mathcal{Y} = \{(u, i, c)|y_{uic} = 1\}$, the neighbors of the user u as the set $\mathcal{N}_u = \{(i, c)|y_{uic} = 1\}$, and neighbors of the item i as the set $\mathcal{N}_i = \{(u, c)|y_{uic} = 1\}$.

We formulate the problem of CARS as:

Input: User-item-context interactions $\{(u, i, c)|y_{uic} = 1\}$, feature vectors of users $\{\mathbf{u}\}$, items $\{\mathbf{i}\}$, and contexts $\{\mathbf{c}\}$.

Output: Prediction function $f : \mathbf{u}, \mathbf{i}, \mathbf{c} \rightarrow \mathbb{R}$, which takes the feature vector of a user, an item, and a context as the input, and outputs a real value that estimates how likely the user will interact with the item under the context.

4 Graph Convolution Machine (GCM)

We present our method in this section. We first describe the predictive model, followed by the model complexity analyses and optimization details.

4.1 Predictive Model

Figure 2 illustrates the model framework, which consists of three components: an encoder, graph convolution layers, and

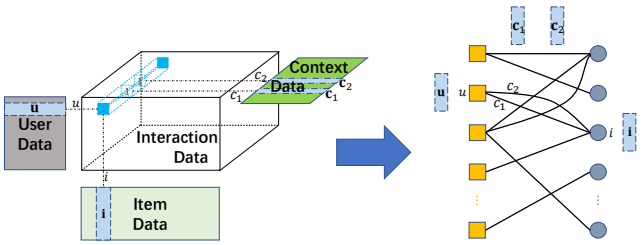


Fig. 1: The data used for building a CARS. The mixture data of interaction tensor and user/item/context feature matrices are converted to an attributed user-item bipartite graph without loss of fidelity.

a decoder. We next describe each component one by one.

4.1.1 Encoder

The input to the encoder has three fields: user-field features $\mathbf{u}$, item-field features $\mathbf{i}$, and the context-field features $\mathbf{c}$. We include the ID feature into the user-field and item-field features, since it helps to differentiate users (items) when their profiles (attributes) are the same[1)]. For each nonzero feature, we associate it with an embedding vector, resulting in a set of embeddings to describe the input user, item, and context, respectively. We then pool the set of user (and item) field into a vector, so as to feed the vector into the the following GC layers to refine the user (and item) representations. Specifically, we adopt average pooling, that is,

$$\mathbf{p}_u^{(0)} = \frac{1}{|\mathbf{u}|}\mathbf{P}^T\mathbf{u}, \tag{1}$$

where $|\mathbf{u}|$ denotes the number of nonzero features in $\mathbf{u}$, and $\mathbf{P} \in \mathbb{R}^{U\times D}$ is the embedding matrix for user features, where U denotes the number of total user features and D denotes the embedding size. $\mathbf{p}_u^{(0)}$ denotes the initial representation vector for u. Similarly, we get the initial representation vector for item i as $\mathbf{q}_i^{(0)}$.

Note that other pooling mechanisms can be applied here, such as the attention-based pooling [19, 34, 38] which learns varying weights for feature embeddings. However, we tried that and find it does not improve the performance. Thus we keep the simplest average pooling and avoid introducing additional parameters. Since we do not update the context representation in the following GC layers, we do not perform pooling on the context field. We denote the set of context-field embeddings as $\mathcal{V}_c = \{\mathbf{v}_s | s \in \mathbf{c}\}$, where $s \in \mathbf{c}$ denotes the nonzero feature in $\mathbf{c}$ and $\mathbf{v}_s$ denotes the embedding vector for context feature s. The encoder outputs $\mathbf{p}_u^{(0)}$, $\mathbf{q}_i^{(0)}$, and $\mathcal{V}_c$, which are fed into the next component of GC layers.

[1)] Note that there is no need to include ID into the context-field features, since a context c and its features $\mathbf{c}$ are one-to-one mapping.

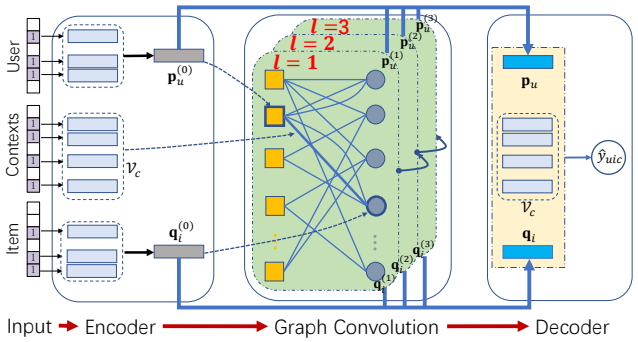


Fig. 2: The Graph Convolution Machine model.

4.1.2 Graph Convolution Layers

This is the core component of GCM, designed to address the limitation of existing supervised learning-based CARS models. It refines $\mathbf{p}_u^{(0)}$ and $\mathbf{q}_i^{(0)}$ by exploiting holistic user-item interaction data, which can augment the user and item representations with explicit collaborative filtering signal [32]. The GC on user-item graph is typically formulated as a message propagation framework:

$$\mathbf{p}_u^{(l+1)} = \sum_{i\in\mathcal{N}_u} g(\mathbf{p}_u^{(l)}, \mathbf{q}_i^{(l)}); \quad \mathbf{q}_i^{(l+1)} = \sum_{u\in\mathcal{N}_i} g(\mathbf{q}_i^{(l)}, \mathbf{p}_u^{(l)}), \tag{2}$$

where $\mathbf{p}_u^{(l)}$ and $\mathbf{q}_i^{(l)}$ denote the refined user representation and item representation of the l-th GC layer, respectively, and $g(\cdot)$ is a self-defined function. Recursively conducting such message propagation relates the representation of a user with her high-order neighbors, *e.g.,* first-order for interacted items and second-order for co-interacted users, which is beneficial for collaborative filtering; and the same logic applies to item representation.

However, the standard GC does not consider the features on edges. In our constructed user-item graph, the edges between a user and an item carry the context features, which are important to understand the context-dependent interaction patterns. For example, a user may prefer bars on Friday, and a restaurant is more popular on lunch time. As such, better user and item representations can be obtained if the context features can be properly integrated into the GC.

To this end, we propose a new GC operation that incorporates the edge features of contexts:

$$\begin{aligned}
\mathbf{p}_u^{(l+1)} &= \sum_{(i,c)\in\mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}}\left(\mathbf{q}_i^{(l)} + \frac{1}{|\mathcal{V}_c|}\sum_{\mathbf{v}_s\in\mathcal{V}_c}\mathbf{v}_s\right), \\
\mathbf{q}_i^{(l+1)} &= \sum_{(u,c)\in\mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}}\left(\mathbf{p}_u^{(l)} + \frac{1}{|\mathcal{V}_c|}\sum_{\mathbf{v}_s\in\mathcal{V}_c}\mathbf{v}_s\right).
\end{aligned} \tag{3}$$

Next we explain the rationality of the GC of the user side, since the item side can be interpreted in the same way. Here $|\mathcal{N}_u|$ denotes the number of edges connected with the user u, and the coefficient $\frac{1}{\sqrt{|\mathcal{N}_u|}}$ is a normalization term to avoid the scale of embedding values increasing with the GC. We incorporate the context features by averaging their embeddings and adding to the connected user embedding. Through this way, we build the connection between a user with both her interacted item and the interacted context. It is expected to capture the effect that if a user likes to choose an item under a certain context, then the similarity among their representations is similar. Note that we have tried more complicated mechanisms like incorporating the pairwise interactions among $\mathcal{V}_c$ and $\mathbf{q}_i^{(l)}$, and using a MLP to capture high-order interactions. However these ways do not lead to performance improvements. Thus we use this simple average operation, which is easy to interpret and train (no additional parameters are introduced).

By stacking multiple such GC layers, a user (or an item) representation can be refined by its multi-hop neighbors. Since the representation of different layers carry different semantics, we next combine the representations of all layers to form a more comprehensive representation:

$$\mathbf{p}_u = \sum_{l=0}^{L} \alpha_l \mathbf{p}_u^{(l)}; \quad \mathbf{q}_i = \sum_{l=0}^{L} \alpha_l \mathbf{q}_i^{(l)}, \tag{4}$$

where α_l denotes the weight of the l-th layer representation. We treat α_l as hyper-parameters, tuning them via grid search with the constraint that $\alpha_l \geq 0$ and $\sum_{l=0}^{L} \alpha_l = 1$. A possible extension is to learn α_l, *e.g.,* designing attention mechanism or optimizing them on the validation data. We leave this extension as future work, since it is not the focus of this work.

In what follows, we provide the matrix form of GC layers for implementation. Let user-item interaction matrix be $\mathbf{R}_{\text{ui}} \in \mathbb{R}^{N \times M}$, where N and M denotes the number of users and items. Each entry $r_{ui} \in \mathbf{R}_{\text{ui}}$ is the number of times user u interacts with item i. Similarly, we utilize $\mathbf{R}_{\text{uc}} \in \mathbb{R}^{N \times L}$ and $\mathbf{R}_{\text{ic}} \in \mathbb{R}^{M \times L}$ to denote user-context interaction matrix and item-context interaction matrix respectively, where L is the number of contexts. Then we define the adjacency matrix of user-item-context graph as

$$\mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{R}_{\text{ui}} & \mathbf{R}_{\text{uc}} \\ \mathbf{R}_{\text{ui}}^T & \mathbf{0} & \mathbf{R}_{\text{ic}} \\ \mathbf{0} & \mathbf{0} & 2\mathbf{I} \end{pmatrix}, \tag{5}$$

where $\mathbf{0}$ is all-zero matrix, $\mathbf{I}$ is identity matrix. Let $\mathbf{D}$ be diagonal degree matrix of $\mathbf{A}$, that is, the t-th diagonal element $\mathbf{D}_{tt} = \sum_j \mathbf{A}_{tj}$. The normalized adjacency matrix can be expressed as

$$\hat{\mathbf{A}} = \sqrt{2}\mathbf{D}^{-\frac{1}{2}}\mathbf{A}. \tag{6}$$

Then, we get the matrix form of the layer-wise propagation rule which is equivalent to Eq. 3:

$$\mathbf{E}^{(l)} = \hat{\mathbf{A}}\mathbf{E}^{(l-1)}, \tag{7}$$

where $\mathbf{E}^{(l)} \in \mathbb{R}^{(N+M+L) \times D}$ is the concatenate of user, item and context embedding matrix. $\mathbf{E}^{(0)}$ is set as the concatenate matrix of encoded embedding tables from Encoder, which can be expressed as

$$\mathbf{E}^{(0)} = [\underbrace{\mathbf{p}_{u_1}^{(0)}, \cdots \mathbf{p}_{u_N}^{(0)}}_{\text{user embeddings}}, \underbrace{\mathbf{q}_{i_1}^{(0)}, \cdots \mathbf{q}_{i_M}^{(0)}}_{\text{item embeddings}}, \underbrace{\mathbf{r}_{c_1}, \cdots \mathbf{r}_{c_L}}_{\text{context embeddings}}]^T. \tag{8}$$

Lastly, we get the final embedding matrix

$$\begin{aligned} \mathbf{E} &= \alpha_0\mathbf{E}^{(0)} + \alpha_1\mathbf{E}^{(1)} + \alpha_2\mathbf{E}^{(2)} \cdots + \alpha_L\mathbf{E}^{(L)} \\ &= \alpha_0\mathbf{E}^{(0)} + \alpha_1\hat{\mathbf{A}}\mathbf{E}^{(0)} + \alpha_2\hat{\mathbf{A}}^2\mathbf{E}^{(0)} + \cdots + \alpha_L\hat{\mathbf{A}}^L\mathbf{E}^{(0)}. \end{aligned} \tag{9}$$

4.1.3 Decoder

The GC layers output refined representation of user $\mathbf{p}_u$ and item $\mathbf{q}_i$, and keep the embeddings of context features unchanged. The role of the decoder is to output the prediction score by taking in the representations. The standard choice of decoder is multi-layer perceptron (MLP), which however falls short here since it only models feature interactions in an implicit way. In CARS, explicitly modeling the interactions between features is known to be important for user preference estimation [8]. For example, the classic factorization machine (FM) models the pairwise interactions between feature embeddings and has long been a competitive model for CARS.

Inspired by the simplicity (linear model) and the effectiveness of FM, we adopt it as the decoder of GCM. The idea is to explicitly model the pairwise interactions between the (refined) representations of user, item, and contexts with inner product. Specifically, let the set of vectors $\mathcal{V}$ be $\mathcal{V}_c \cup \mathbf{p}_u \cup \mathbf{q}_i$, the decoder outputs the prediction score as:

$$\hat{y}_{uic} = \frac{1}{2}\left(\sum_{\mathbf{v}_s \in \mathcal{V}} \sum_{\mathbf{v}_t \in \mathcal{V}} \mathbf{v}_s^T\mathbf{v}_t - \sum_{\mathbf{v}_s \in \mathcal{V}} \mathbf{v}_s^T\mathbf{v}_s\right). \tag{10}$$

Here the self-interactions $\mathbf{v}_s^T\mathbf{v}_s$ are excluded since they are useless for the prediction. The bias terms for each user, item, and context feature are omitted for clarity.

Note that our FM-based decoder slightly differs from the vanilla FM, which models the interactions between the embeddings of all input features. Here we project each user (item) into a vector, rather than retaining the embeddings of her (its) features. An advantage is that this way abandons the internal interactions of user-field (item-field) features, shedding more light on the interactions between user (item) and context features, which is as expected.

4.2 Model Complexity Analyses

We analyze the complexity of GCM from two aspects: the number of trainable parameters and the time complexity.

All trainable parameters come from the encoder layer, *i.e.,* the embeddings of input features, since the GC layers and the decoder layer introduce no parameters to train. Let the feature number for the user field, item field, and context field as U, I, and C, respectively, and the embedding size be D. Then the embedding layer costs $(U + I + C) \times D$ parameters. This demonstrates the low model complexity of GCM, since the number of trainable parameters is the same as FM — the most simple embedding-based CARS model.

For model training, since the complexity of the encoder plus the decoder is the same as that of FM, we analyze the additional time complexity caused by the GC layers. We implement the training in the batch-wise matrix form. Assume a batch contains all interactions. Then performing one GC layer takes time $O((|\mathcal{Y}| + N + M)D)$, where N and M denote the number of users and items, respectively. This complexity increases linearly with the number of GC layers.

After the model is trained, we perform one pass of GC layers to obtain the refined representations of all users and items, which can be done offline before online serving. As such, during online serving, we only need to execute the decoder, which has the same time complexity of FM. This is much faster than the recently emerging deep neural network-based CARS models like xDeepFM [17] and Convolutional FM [38]. Table 1 shows the model inference time of evaluating 1000 Yelp-OH users in which each interaction has 10 nonzero features of embedding size 64 and batch size is 4000. The testing platform is GeFore GTX 1080Ti with 16GB memory CPU. As can be seen, GCM takes similar time as FM, being 24.5 and 157.7 times faster than xDeepFM and Convolutional FM, respectively.

4.3 Optimization

To optimize model parameters, we opt for the pointwise log loss, which is a common choice in recommender system [10, 17]. In each training epoch, we randomly sample 4 (or 2) non-observed interactions for each instance in $\mathcal{Y}$ of Yelp (or Amazon) dataset, forming the negative set $\mathcal{Y}^-$. Then we minimize the following objective function:

$$L = -\sum_{(u,i,c)\in\mathcal{Y}} \log \sigma(\hat{y}_{uic}) - \sum_{(u,i,c)\in\mathcal{Y}^-} \log\left(1 - \sigma(\hat{y}_{uic})\right) + \lambda \|\Theta\|_2^2, \quad (11)$$

where $\sigma(\cdot)$ is the sigmoid function, λ controls the L_2 regularization to prevent over-fitting. The optimization is done by mini-batch Adam [12].

Table 1: Model inference time of evaluating 1,000 Yelp-OH users (14 million interactions and 10 nonzero features per interaction).

Model	FM	GCM	GIN	xDeepFM	Convolutional FM
Time/s	8.51	14.93	35.45	365.82	2354.25

5 Experiments

We evaluate experiments on three benchmark datasets, aiming to answer the following research questions:

- **RQ1:** Compared with the state-of-the-art models, how does GCM perform *w.r.t.* top-k recommendation?
- **RQ2:** How do different settings (*e.g.,* depth of layer, modeling of context features, design of decoder) affect GCM?
- **RQ3:** How do the representation learning benefit from multiple interactions among users, items and contexts for item cold start issue?

5.1 Experimental Settings

5.1.1 Dataset Description

To demonstrate the effectiveness of GCM, we conduct experiments on three datasets from Yelp and Amazon, which are publicly available and vary in domain and size. We summarize the statistics of datasets in Table 2.

- **Yelp**: This dataset is released by Yelp and records users' reviews on local businesses like bars and restaurants. In particular, we extract records happened in two different areas of USA — North Carolina, Ohio States — to construct datasets, termed Yelp-NC and Yelp-OH respectively.
- **Amazon**: Amazon review data is widely used in recommendation [32]. We select book subset from the collection in this work, and term it Amaon-book.

Table 2: Statistics of the datasets.

Dataset	Yelp-NC	Yelp-OH	Amazon-book
#User	6,336	5,170	44,709
#Item	13,003	12,997	46,831
#Instance	185,408	143,884	1,174,785
#User Feature	24	24	-
#Item Feature	68	213	24,816
#Context Feature	13,209	13,347	46,900

In what follows, we briefly introduce the features of users, items, and contexts. Specifically, for Yelp-NC and Yelp-OH, each user profile includes *yelping_since_year*[2] and *average_stars*, while the pre-existing features of items are composed of three attributes: *city*, *stars* and *is_open*. We treat each review record as an observed instance, and collect *city*[3], *month*, *hour*, *day_of_week* and *last_purchase* as its context feature. For Amaon-book, the static features of items are composed of two attributes: *price* and *brand*. Similarly, each review record is treated as an observed instance, and *year*, *month*, *day*, *day_of_week* and *last_purchase* are collected as its context feature. Moreover, for all datasets, the 10-core setting is adopted to ensure data quality, *i.e.,* retaining users with at least ten interactions.

For each user, we select the last interaction record to constitute the test set, while the remains are served as the training set. To emphasize model capability in recommending novel items for a user, we further filter the training set if the user-item pairs have appeared in the test set.

5.1.2 Evaluation Metrics

In the evaluation phase, for each user in the test set, we view all items that she has not consumed before as recommendation candidates. Each method outputs a ranking list over the candidates. We then adopt two widely-used protocols to evaluate the quality of ranking lists: Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG). In particular, HR@K measures whether the test item is in the top-K positions of the recommended list, whereas NDCG@K assigns higher scores to the top-ranked items. In our experiments, we report the results of $K = 10$ and $K = 50$.

5.1.3 Baselines

We compare our GCM with several methods as follows:

- **MF** [14]: This exploits the user-item interactions only to learn user and item embeddings, while forgoing the context features.
- **LightGCN** [9]: Such model is the state-of-the-art GNN-based CF recommender, which incorporates high-order connectivity in user-item interaction graph into embeddings, while neglecting context features.
- **FM** [20]: This takes into account all information related to an interaction by converting all information to a feature vector then modeling second-order feature interaction to predict user preference.
- **NFM** [8]: This model leverages a MLP to capture non-linear and high-order interaction among user, item, and context features.
- **xDeepFM** [17]: This is a recent neural FM model which combines explicit and implicit high-order feature interactions.
- **GIN** [15]: This is a graph-based model which mines user intention by applying implicit intention propagation and attention mechanism on commodity similarity graph.

Fi-GNN [16] is a recent work on click-through rate prediction with graph neural network, which is highly relevant with our work. It differs from GCM in graph construction — it builds a feature graph for each interaction, rather than the user-item graph. As a graph needs be built for each interaction to obtain its prediction, the method is very slow in evaluation since all recommendation candidates need be scored. As such, this method is not suitable for our all-ranking CARS evaluation, and we do not further compare with it. The Convolutional FM is not compared for the same reason (see Table 1 for model inference time). It's worth mentioning that the core of CARS and CTR prediction are common essentially, that is, modeling the complex feature interactions. The key difference lies in evaluation protocol: most CARS models adopt top-k recommendation protocol while CTR prediction models measure log loss or AUC metrics on positive/negative samples.

5.1.4 Parameter Settings

We implement our GCM model and all baselines in Tensorflow and will release our code upon acceptance. We apply the mini-batch Adam to optimize all models, the learning rate and batch size are set to 0.001 and 2048 respectively. A grid search is conducted for confirming optimal hyperparameters: for LightGCN, the number of gcn layers is fixed to 3, as suggested by the authors; for NFM, the number of hidden lay-

[2] We only keep the *year* of the *yelping_since* field which indicates the time the user joined Yelp.

[3] The context feature *city* means which city does the interaction happen on. It is set as the city of the interacted item.

ers is set to 1, the dropout rate is tuned in $\{0.9, 0.8, \cdots, 0.1\}$ for bi-interaction layer and hidden layer respectively; for xDeepFM, the number of cross layers is searched in $\{1, 2, 3\}$ with neuron number per layer in $\{10, 20, 50, 100, 200\}$, the number of DNN layers is same to that of cross layers, while the neuron number per layer is set to 100; for GIN, the length of previous records is 1 since we only keep the last purchase information in the datasets, the depth parameter is searched in $\{1, 2\}$, the number of neighbor nodes is tuned in $\{10, 20\}$, the neighbor is selected by the Top-N function according to the edge weight (for nodes with few neighbors, we randomly sample from unconnected nodes as their potential neighbors), a 5-layer full-connection perceptron with ReLU activation is adopted as the setting in [15]; for the proposed GCM, we search the model depth amongst $\{1, 2, 3\}$, and adopt average pooling to generate the final refined representations of GC layers. For all models, the coefficient of L_2 regularization term is searched in $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. Moreover, we set the embedding size to 64 and train all models from the scratch.

5.2 Performance Comparison (RQ1)

We report the empirical results of all models in Table 3 and have the following observations:

- Clearly, MF achieves the worst performance on three datasets, indicating that modeling user-item pairs as isolated instances limits the representation ability severely. LightGCN obtains consistent improvements over MF. We attribute such improvements to the modeling of user-item connectivity. However, neither MF nor LightGCN takes the context features into consideration, ignoring important factors and being insufficient for CARS.
- FM, NFM and xDeepFM consistently outperform MF and LightGCN across all cases. This is reasonable since they incorporate context features into the representation learning, so as to achieve better expressiveness and help to solve the data sparsity issue; Among them, NFM and xDeepFM perform better than FM by a large margin since they model more complex feature interactions: NFM employs MLP on user, item, and context features to capture their nonlinear and complex interactions, while xDeepFM learns high-order feature interactions in a more explicit way through a CIN network. This verifies that simply linear functions (*e.g.,* inner product adopted by MF and LightGCN) might limit the representation learning and interaction modeling.
- GIN is the strongest baseline in all cases except for NDCG@10 and NDCG@50 in Yelp-NC. Such improvements is mainly because of GIN's capability to model user intention by applying message propagation in commodity similarity graph, which also verify the necessity of bridging the relationship among data instances.
- GCM consistently outperforms all baselines *w.r.t.* all measures. In particular, GCM achieves noticeable improvements over the strongest baselines *w.r.t.* HR@10 by 21.21%, 14.93%, and 3.09%, in Yelp-NC, Yelp-OH, and Amazon-book, respectively. We attribute such improvements to 1) GCM employs the embedding propagation over the attributed graph to distill useful information from neighbors and improve the representation ability; 2) Comparing with GIN which only propagates item embedding in the graph, GCM integrates the representations of users, items and contexts into the graph for information propagation, which may results in a more unified representations; and 3) Having established the refined representations, GCM further adopts FM to explicitly model the feature interactions.

5.3 Study of GCM (RQ2)

We next report ablation studies to verify the rationality of some designs in GCM, *i.e.,* analyzing the influence of model depth, context modeling, normalization term, and decoder.

5.3.1 Impact of Model Depth

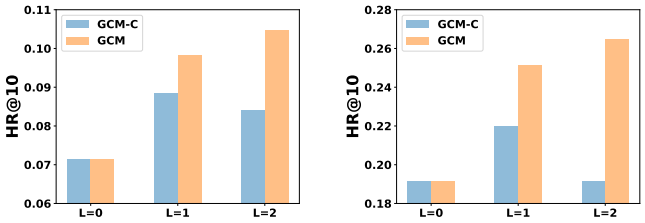
As GC is the core of GCM and stacking more GC layers is expected to augment the user and item representations with information propagated from multi-hop neighbors, we investigate how the number of GC layers affects the performance. In particular, we search the number of GC layers, L, in the range of $\{0, 1, 2\}$ and report the empirical results in Figure 3.

We use GCM-1 to represent the model with one GC layer, and similar notations for others. We have several findings:

- GCM-0 disables the embedding propagation over user-item attributed graph and downgrades to a FM-like linear model, thereby achieving poor performance. This again justifies the importance of GC layers.
- Obviously, increasing the number of GC layers consistently results in better performance across all cases. In particular, GCM-2 performs better than GCM-1. It is reasonable since the signals passing from multi-hop neighbors (*e.g.,* the second-order connectivity between behaviorally similar users or co-purchased items) are encoded into user and item representations of GCM-2, while GCM-1 only exploits personal history to enrich

Table 3: Overall Performance Comparison.

	Yelp-NC				Yelp-OH				Amazon-book			
	HR		NDCG		HR		NDCG		HR		NDCG	
	@10	@50	@10	@50	@10	@50	@10	@50	@10	@50	@10	@50
MF	0.0384	0.1173	0.0175	0.0341	0.0429	0.1261	0.0206	0.0383	0.0402	0.1243	0.0203	0.0382
LightGCN	0.0499	0.1394	0.0241	0.0431	0.0513	0.1503	0.0254	0.0464	0.0543	0.1466	0.0274	0.0473
FM	0.0739	0.1804	0.0396	0.0624	0.1959	0.4201	0.1049	0.1538	0.0587	0.1477	0.0323	0.0514
NFM	0.0824	0.2110	0.0419	0.0695	0.2248	0.4836	0.1161	0.1725	0.0808	0.1954	0.0444	0.0692
xDeepFM	0.0851	0.2086	0.0458	0.0723	0.2296	0.4799	0.1218	0.1762	0.0886	0.2119	0.0481	0.0748
GIN	0.0863	0.2140	0.0441	0.0715	0.2304	0.4965	0.1238	0.1818	0.0939	0.2189	0.0502	0.0774
GCM	**0.1046**	**0.2421**	**0.0557**	**0.0854**	**0.2648**	**0.5166**	**0.1457**	**0.2008**	**0.0968**	**0.2232**	**0.0536**	**0.0810**



(a) Yelp-NC (b) Yelp-OH

Fig. 3: The impact of depth and context modeling in GC.

representations. This observation is consistent with that in NGCF [32]. We also tried to stack more GC layers (*i.e.,* GCM-3), finding improvement degrades and over-smoothing issue. This suggests that GCM benefits from the first- and second-order neighbors most, but may suffer from degradation when higher-order neighbors are involved.

5.3.2 Impact of Context Modeling

One major contribution of GCM is to organize the context features as edges in the attributed user-item graph. We hence perform ablation study, to demonstrate the rationality and effectiveness of this design. In particular, we build the variant GCM-C by removing the context features from the attributed graph and keeping only the vanilla user-item interaction graph. We show the comparison between GCM and GCM-C in Figure 3 and have the following observations.

- Modeling context features as the edges endows GCM with better generalization ability. In particular, GCM is superior to GCM-C consistently. This again demonstrates the rationality of our context modeling.
- Jointly analyzing Table 3 and Figure 3, we find that GCM-C without considering contexts achieves better performance than other baselines in Yelp-NC and comparable performance in Yelp-OH. This empirically suggests that propagating embeddings over interaction graphs is of importance to generate high-quality representations.
- The performance of GCM-C decreases with the increase of the number of gc layers,, indicating that incorporating the context into the graph for information propagation can improve the generalization capability of the model meanwhile alleviate the oversmoothing issue.

5.3.3 Impact of normalization term

For convenience, we only present the variants of the GC of the user side, since the same logic can be applied to the item side. In GCM, we employ sqrt normalization term $\frac{1}{\sqrt{|\mathcal{N}_u|}}$ on each neighbor embedding when performing neighborhood aggregation. To verify its rationality, we explore two different variants and report their empirical results here. The first variant uses symmetric normalization term, *i.e.,* $\frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}}$, which is a common choice in GCN-based models [9], we term it GCM-sym. The other variant uses L_1 normalization, *i.e.,* $\frac{1}{|\mathcal{N}_u|}$, we term it GCM-$L_1$. Table 4 shows the results of the 2-layer GCM. We have the following observations:

- The best setting in general is using sqrt normalization term on single side (*i.e.,* the current degign of GCM). Adding additional regularization coefficients will greatly drop the performance.
- The smaller the normalization term, the worse the performance. To understand this observation, we can see the following inequalities: $\frac{1}{\sqrt{|\mathcal{N}_u|}} > \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} > \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_u|}} = \frac{1}{|\mathcal{N}_u|}$. The second inequality is due to $|\mathcal{N}_u| > |\mathcal{N}_i|$ on everage in Yelp-NC and Yelp-OH.

5.3.4 Impact of Decoder

Having applied GC layers, we equip GCM with a decoder to model the pairwise interactions between the refined representations of users, items, and contexts. Here we investigate the

Table 4: The variants of GCM with different normalization terms and decoders

	Yelp-NC		Yelp-OH	
	HR@10	NDCG@10	HR@10	NDCG@10
GCM	0.1046	0.0557	0.2648	0.1457
GCM-L_1	0.0810	0.0421	0.2373	0.1246
GCM-sym	0.0994	0.0527	0.2507	0.1383
GCM-MLP	0.0892	0.0458	0.2263	0.1211
GCM-MF	0.0497	0.0253	0.0520	0.0251

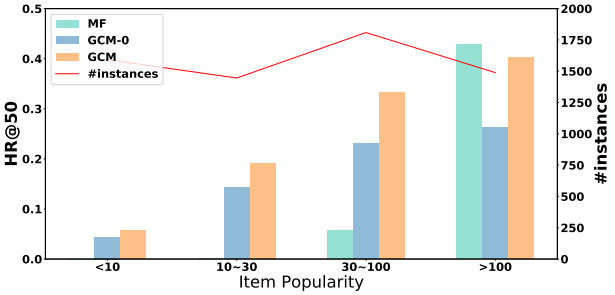
role of such decoder. Towards this end, we compare GCM with two variants, GCM-MLP and GCM-MF, which separately replace the decoder with MLP and inner product on user and item representations. Table 4 shows the comparison of results. There are several observations:

- Clearly, modeling feature interactions in the decoder enhances the predictive results. In particular, GCM and GCM-MLP perform consistently better than GCM-MF, which relies only on the inner product of user and item representations.
- While having encoded context features into user and item representations via GC layer, GCM highlights their influence in an explicit fashion, while GCM-MLP models the feature interactions in a rather implicit way. The better performances of GCM again verfy the rationality and effectiveness of FM-based decoder.

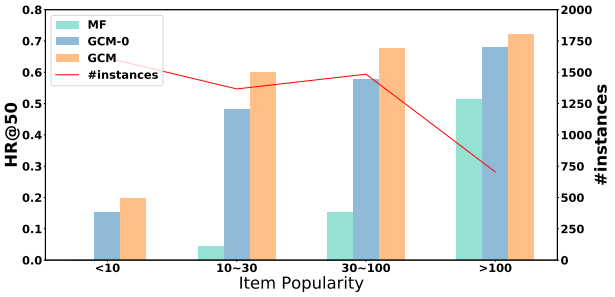
5.4 Performance *w.r.t.* Item Popularity (RQ3)

To alleviate the issue of item cold start of CF models, taking side information into account is an auxiliary strategy go beyond modeling user-item interaction. In the proposed GCM, We apply gc layers to capture high-order connectivity on user-item graph, which breaks down the independent interaction assumption of non-graph-based methods. We argue that such connectivity is a potential side information for cold-start issue. To verify this viewpoint, we split the test set according to the popularity (the number of interaction records) of the target item, and report the performance of MF [14], GCM-0 and GCM in Figure 4. We have the following observations:

- MF performs poorly at unpopular items, which indicates the item cold-start issue for CF models. GCM-0 has significant improvements in recommending uncommon items by introducing side information and modeling feature interactions. Our GCM can further improve performance by 20%-30%. We attribute such improvements to modeling high-order connectivity since gnn increases the possibility of unpopular item being exposed through high-order links, thereby expanding the training data of unpopular items.
- For popular items, MF achieves comparable performance with GCM, even prevails over GCM in Yelp-NC. The possible reason is that the data of popular items occupies the majority of the training data, making MF adopt a cautious strategy — biased to recommending generally accepted items. Instead, GCM will recommend items that are more niche but still consistent with user's taste.
- The gain brought by gcn decreases as the popularity of items increases. This shows that as the number of neighbors increases, gcn may suffer from over-smoothing, since these items have too many audiences, causing collecting information from neighbor nodes will also bring in noises.



(a) Yelp-NC



(b) Yelp-OH

Fig. 4: Performances with respect to item popularity

6 Conclusion and Future Work

In this work, we emphasize the importance of exploiting multiple interactions in CARS. Towards this end, we first convert the features of users, items, and contexts into an attributed graph involving the contexts as edges between user and item nodes. We then develop a new model, GCM, which captures the interactions among multiple user behaviors via graph neu-

ral networks, and then models the interactions among features of individual behavior via factorization machine. To demonstrate the effectiveness of GCM, we test it on three public datasets, and it shows significant improvements over the state-of-the-art CF and CARS baselines. Extensive experiments also are conducted to verify the rationality of attributed graph and offer insights into how the representations benefit from such graph learning.

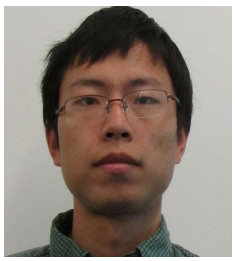
Organizing user behaviors with contextual information in graphs is a promising direction to build an effective context-aware recommender. It helps build strong representations for users and items. However, GCM simply unifies all context features as an edge, neglecting dynamic characteristics of some contexts (*e.g.,* time) and hardly capturing dynamic preference of users [1]. In future, we plan to build dynamic graphs based on contextual information, instead of one static graph, and devise a dynamic graph neural network. Furthermore, rich side information is beneficial for explaining diverse intents behind user behaviors [31]. We hence plan to model user-item relationships at a granular level of user intents to generate disentangled representations [18].

References

1. Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H Chi. Latent cross: Making use of context in recurrent recommender systems. In *WSDM*, pages 46–54, 2018.
2. Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *WWW*, pages 151–161, 2019.
3. Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *DLRS*, pages 7–10. ACM, 2016.
4. Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *WWW*, pages 417–426, 2019.
5. Liyu Gong and Qiang Cheng. Exploiting edge features for graph neural networks. In *CVPR*, pages 9211–9219, 2019.
6. Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: A factorization-machine based neural network for CTR prediction. In *IJCAI*, pages 1725–1731, 2017.
7. William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.
8. Xiangnan He and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. In *SIGIR*, pages 355–364, 2017.
9. Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. *arXiv preprint arXiv:2002.02126*, 2020.
10. Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *WWW*, pages 173–182, 2017.
11. Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Recsys*, pages 79–86. ACM, 2010.
12. Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.
13. Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.
14. Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *IEEE, Computer Journal*, 42(8):30–37, 2009.
15. Feng Li, Zhenrui Chen, Pengjie Wang, Yi Ren, Di Zhang, and Xiaoyu Zhu. Graph intention network for click-through rate prediction in sponsored search. In *SIGIR*, page 961–964, 2019.
16. Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction. In *CIKM*, pages 539–548, 2019.
17. Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *SIGKDD*, pages 1754–1763, 2018.
18. Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. Disentangled graph convolutional networks. In *ICML*, pages 4212–4221, 2019.
19. Lei Mei, Pengjie Ren, Zhumin Chen, Liqiang Nie, Jun Ma, and Jian-Yun Nie. An attentive interaction network for context-aware recommendations. In *CIKM*, pages 157–166, 2018.
20. Steffen Rendle. Factorization machines. In *ICDM*, pages 995–1000, 2010.
21. Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.
22. Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *SIGIR*, pages 635–644, 2011.
23. Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *SIGKDD*, pages 255–262, 2016.
24. Yue Shi, Martha Larson, and Alan Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)*, 47(1):1–45, 2014.
25. Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. Session-based social recommendation via dynamic graph attention networks. In *WSDM*, pages 555–563, 2019.
26. Rianne van den Berg, Thomas N. Kipf, and Max Welling. Graph convolutional matrix completion. *CoRR*, abs/1706.02263, 2017.
27. Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks.

ICLR, 2018. accepted as poster.
28. Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. Deep & cross network for ad click predictions. In *ADKDD*, pages 12:1–12:7. ACM, 2017.
29. Wen Wang, Wei Zhang, Shukai Liu, Qi Liu, Bo Zhang, Leyu Lin, and Hongyuan Zha. Beyond clicks: Modeling multi-relational item graph for session-based target behavior prediction. In *WWW*, pages 3056–3062, 2020.
30. Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. KGAT: knowledge graph attention network for recommendation. In *SIGKDD*, pages 950–958, 2019.
31. Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. Tem: Tree-enhanced embedding model for explainable recommendation. In *WWW*, page 1543–1552, 2018.
32. Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *SIGIR*, pages 165–174, 2019.
33. Yinwei Wei, Xiang Wang, Liqiang Nie, Xiangnan He, Richang Hong, and Tat-Seng Chua. MMGCN: multi-modal graph convolution network for personalized recommendation of micro-video. In *MM*, pages 1437–1445, 2019.
34. Chuhan Wu, Fangzhao Wu, Tao Qi, Suyu Ge, Yongfeng Huang, and Xing Xie. Reviews meet graphs: Enhancing user and item representations for recommendation with hierarchical attentive graph neural network. In *EMNLP-IJCNLP*, pages 4886–4895, 2019.
35. Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. A neural influence diffusion model for social recommendation. In *SIGIR*, pages 235–244, 2019.
36. Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. In *AAAI*, pages 346–353, 2019.
37. Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. Attentional factorization machines: Learning the weight of feature interactions via attention networks. In *IJCAI*, pages 3119–3125, 2017.
38. Xin Xin, Bo Chen, Xiangnan He, Dong Wang, Yue Ding, and Joemon Jose. CFM: convolutional factorization machines for context-aware recommendation. In *IJCAI*, pages 3926–3932, 2019.
39. Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.
40. Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *SIGKDD*, pages 974–983, 2018.
41. Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *NIPS*, pages 5171–5181, 2018.
42. Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S Yu. Spectral collaborative filtering. In *Recsys*, pages 311–319, 2018.

Jiancan Wu received his B.S. degree in Electronic Engineering and Information Science from the University of Science and Technology of China (USTC) in 2017. He is currently a Ph.D. student at USTC. His research interests focus on Recommender Systems, Machine Learning and Graph Neural Networks.

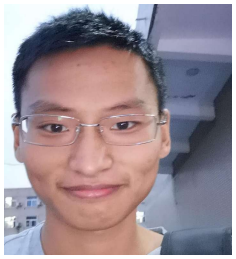


Qifan Wang received the BS and MS degrees from Tsinghua University, and the PhD degree from Purdue University, all in computer science. He is a researcher in Google Research, and his research interests include machine learning, information retrieval, data mining, computer vision and natural language processing.

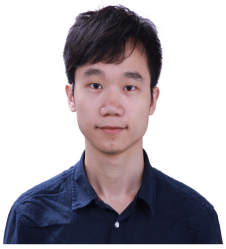
Dr. Xiangnan He is a professor at the University of Science and Technology of China (USTC). He received his Ph.D. in Computer Science from the National University of Singapore (NUS) in 2016. His research interests span information retrieval, data mining, and multi-media analytics. He has over 70 publications that appeared in several top conferences such as SIGIR, WWW, and MM, and journals including TKDE, TOIS, and TMM. His work on recommender systems has received the Best Paper Award Honorable Mention in WWW 2018 and ACM SIGIR 2016. Moreover, he has served as the PC chair of CCIS 2019, area chair of MM (2019, 2020) ECML-PKDD 2020, and PC member for several top conferences including SIGIR, WWW, KDD, WSDM etc., and the regular reviewer for journals including TKDE, TOIS, TMM, etc.

Xiang Wang is now a research fellow at National University of Singapore. He received his Ph.D. degree from National University of Singapore in 2019. His research interests include recommender systems, graph learning, and deep learning techniques. He has published some academic papers on international conferences such as KDD, WWW, SIGIR, and AAAI. He

serves as a program committee member for several top conferences such as SIGIR and WWW.



Weijian Chen is currently a Ph.D. student at the University of Science and Technology of China(USTC). His research interests focus on User Profiling, Recommender System, and Graph Neural Networks. He has served as a research intern in the JD Data Science Laboratory and a project intern in the Kwai Multimedia Understanding Department.



Jianxun Lian is now a senior researcher at Microsoft Research Asia. He received his Ph.D. degree from University of Science and Technology of China in 2018. His research interests include recommender systems and deep learning techniques. He has published some academic papers on international conferences such as KDD, IJCAI, WWW, SIGIR and CIKM. He serves as a program committee member for several top conferences such as AAAI, WWW and IJCAI.

Dr. Xing Xie is currently a senior principal research manager at Microsoft Research Asia, and a guest Ph.D. advisor at the University of Science and Technology of China. He received his B.S. and Ph.D. degrees in Computer Science from the University of Science and Technology of China in 1996 and 2001, respectively. He joined Microsoft Research Asia in July 2001, working on data mining, social computing and ubiquitous computing. During the past years, he has published over 300 referred journal and conference papers, won the 10-year impact award in ACM SIGSPATIAL 2019, the best student paper award in KDD 2016, and the best paper awards in ICDM 2013 and UIC 2010.