# User-Event Graph Embedding Learning for Context-Aware Recommendation

Dugang Liu*
Shenzhen University
Shenzhen, China
dugang.ldg@gmail.com

Mingkai He
Jinwei Luo
Shenzhen University
Shenzhen, China

Jiangxu Lin
Meng Wang
Southeast University
Nanjing, China

Xiaolian Zhang
Huawei Technologies
Co Ltd
Shenzhen, China

Weike Pan†*
Zhong Ming†*
Shenzhen University
Shenzhen, China

## ABSTRACT

Most methods for context-aware recommendation focus on improving the feature interaction layer, but overlook the embedding layer. However, an embedding layer with random initialization often suffers in practice from the sparsity of the contextual features, as well as the interactions between the users (or items) and context. In this paper, we propose a novel user-event graph embedding learning (UEG-EL) framework to address these two sparsity challenges. Specifically, our UEG-EL contains three modules: 1) a graph construction module is used to obtain a user-event graph containing nodes for users, intents and items, where the intent nodes are generated by applying intent node attention (INA) on nodes of the contextual features; 2) a user-event collaborative graph convolution module is designed to obtain the refined embeddings of all features by executing a new convolution strategy on the user-event graph, where each intent node acts as a hub to efficiently propagate the information among different features; 3) a recommendation module is equipped to integrate some existing context-aware recommendation model, where the feature embeddings are directly initialized with the obtained refined embeddings. Moreover, we identify a unique challenge of the basic framework, that is, the contextual features associated with too many instances may suffer from noise when aggregating the information. We thus further propose a simple but effective variant, i.e., UEG-EL-V, in order to prune the information propagation of the contextual features. Finally, we conduct extensive experiments on three public datasets to verify the effectiveness and compatibility of our UEG-EL and its variant.

## CCS CONCEPTS

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

## KEYWORDS

Context-aware recommendation, Graph embedding learning, User-event graph, User intent

*Also with Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ).

†Co-corresponding authors


*KDD '22, August 14–18, 2022, Washington, DC, USA*

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539458

**ACM Reference Format:**
Dugang Liu, Mingkai He, Jinwei Luo, Jiangxu Lin, Meng Wang, Xiaolian Zhang, Weike Pan, and Zhong Ming. 2022. User-Event Graph Embedding Learning for Context-Aware Recommendation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3534678.3539458

## 1 INTRODUCTION

Previous studies have shown that users' behaviors are often influenced by the contextual information, and context-aware recommender systems (CARS) are proposed to integrate these contextual information to make more accurate fine-grained recommendations to users [3, 21, 37]. The contextual information can be either explicitly observed or inferred from the latent space of the embedding vectors [6, 26], and they can be used in the pre-filtering, post-filtering, or modeling stages of a recommendation task [1]. Most of the existing works for CARS focus on the modeling stage and can be classified into two lines according to the adopted model architecture: 1) the first line is to extend a recommendation task to the multidimensional settings based on some machine learning methods to model the contextual information, especially matrix factorization-, tensor factorization-, and factorization machine-based methods [2, 4, 14, 23, 36, 38]; and 2) to improve the modeling of higher-order and nonlinear relationships among the features, the second line introduces some complex neural network architectures into CARS, such as attention mechanisms [22, 39], convolutional networks [10, 34], and graph learning techniques [5, 19, 30].

Although existing methods for CARS have shown promising results, most of them focus on improving the feature interaction layer in the model to mine more beneficial information for recommendation, but overlook the embedding layer. However, an embedding layer with random initialization suffers from the sparsity of the contextual features, as well as the interactions between the users (or items) and the contextual features in practice. The performance of models for context-aware recommendation often heavily relies on the learning of the contextual features. However, there are numerous sparse contextual features in a real recommender system, i.e., they rarely appear in the training set. To validate this observation, as shown in the right column of Figure 1, we visualize the frequency distribution of the contextual features for each of the three datasets used in the experiments. We can see that most of the contextual features have a low frequency. This means that it is difficult for most existing methods to learn a good embedding for these sparse contextual features due to insufficient training examples, i.e., the model performance will suffer from random initialization. We refer to this challenge as **feature sparsity**. Additionally, accurately capturing the relationship between the contextual features and users

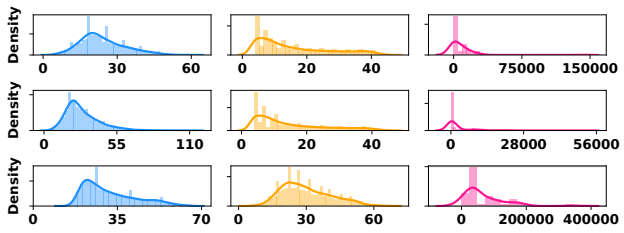



**Figure 1: The distributions of the number of contextual features associated with each user (left column) and item (middle column), as well as the frequency statistics of the contextual features (right column), on Yelp-NC (top row), Yelp-OH (middle row) and Amazon-Book (bottom row).**

(or items) can also benefit the recommendation performance. However, from the left and middle columns of Figure 1, we can observe a long-tailed distribution of the numbers of the associated contextual features w.r.t. the users (or items) in each dataset. Therefore, the existing methods may have a performance bottleneck for these inactive users or unpopular items due to the insufficient preference information for the contextual features. We refer to this challenge as **interaction sparsity**.

Integrating and utilizing graph representation has shown to be effective in mitigating data sparsity in other recommendation tasks [9, 13, 27]. Motivated by this, in this paper, we first construct a novel user-event graph for CARS, where the contextual features are used to construct some additional user-intent nodes according to the proposed intent-node attention (INA). These intent nodes can act as a hub to build complex interactions among the users, items, and contextual features, and are potentially useful for capturing user preferences for different contextual features. Based on user-event graph, we then integrate and leverage graph embedding learning to obtain refined embeddings for the users, items, and contextual features. Specifically, we propose a user-event collaborative graph convolution, where the users, items and contextual features can all benefit from the information propagation process of graph embedding learning through the constructed intent nodes. This means that the nodes of the users, items and contextual features trapped in the above two sparsity challenges may gain more synergistic information to facilitate learning better embedding representations. The obtained refined embeddings of all features can later be used in a certain existing CARS model to improve the performance. Therefore, we integrate all the above modules to obtain a general embedding learning framework, which is called UEG-EL. In particular, we also observe a unique challenge of UEG-EL in practical applications that the contextual features associated with too many instances may suffer from noise when aggregating information, and thus propose a simple but effective variant to alleviate it. Finally, we conduct extensive experiments on three public datasets to verify the effectiveness and compatibility of our UEG-EL and its variant.

## 2 RELATED WORK

In this section, we briefly review some relevant works on two research topics, including graph neural network for recommendation and context-aware recommender systems.

**Graph Neural Network for Recommendation.** Graph neural network has been shown to be able to learn node embeddings more accurately by capturing complex structural information, which are beneficial for alleviating data sparsity [7, 11, 28]. These characteristics have made it attract a lot of attention in the community of recommender systems recently. Most existing works adopt a user-item bipartite graph to organize the user behaviors, and design different graph learning strategies based on the bipartite graph to learn the user and item embeddings [27, 31]. Since graph convolution usually requires a large computational cost, some follow-up works aim to optimize its efficiency [13, 20]. In addition, graph neural network can be combined with some additional information for other recommendation tasks, such as sequential recommendation [29, 33], social recommendation [25, 32], and CTR prediction [9, 16, 17]. However, few works focus on context-aware recommendation, in particular of designing a specialized graph structure for this task, while our UEG-EL is an exploration in this direction.

**Context-Aware Recommender Systems.** Context-aware recommendation aims to make a finer-grained recommendation to users by utilizing the rich contextual information, which can be either explicitly observed or implicitly inferred from the latent space of the embedding vectors [1, 6, 26]. The contextual information can be used in the pre-filtering, post-filtering, or modeling stages of a recommendation task, the last of which is a more popular paradigm [1]. Existing works in this line can be divided into two classes according to the adopted model architecture, including machine learning and neural network-based methods. The former aims to extend a recommendation task to the multidimensional settings to model the contextual information, where some machine learning methods, especially matrix factorization, tensor factorization, and factorization machine, are adopted [4, 14, 23, 24, 36, 38]. The latter further captures higher-order and nonlinear relationships between different features by introducing some complex neural network structures, such as attention mechanisms [22, 39], convolutional networks [10, 34], and graph learning techniques [5, 19, 30]. One of the important branches is its combination with factorization machine, such as NFM [12] and xDeepFM [18]. As the most related work to ours, GCM [30] proposes to use the contextual features as the edge features on user-item bipartite graph, and designs a new graph convolution to learn the graph structure. Our UEG-EL differs significantly from it: 1) our proposed user-event graph introduces some additional intent nodes to better connect different features; and 2) treating the contextual features as nodes enables them to benefit from graph embedding learning similar to the user and item nodes, i.e., our UEG-EL refines the embeddings of all features instead of ignoring the contextual features in GCM.

## 3 PRELIMINARIES

### 3.1 Problem Definition

In this subsection, we formally define the context-aware recommendation task with the necessary notations. A typical context-aware recommender system (CARS) usually consists of a set of $M$ users $\mathcal{U} = \{u_1, u_2, \ldots, u_M\}$, a set of $N$ items $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$, and a set of $R$ fields of contextual features denoted as $C = \{C^1, C^2, \ldots, C^R\}$, such as timestamps and locations. In addition, a set of $J$ fields of user attributes $\mathcal{A} = \{\mathcal{A}^1, \mathcal{A}^2, \ldots, \mathcal{A}^J\}$ and a set of $K$ fields of item

attributes $\mathcal{B} = \{\mathcal{B}^1, \mathcal{B}^2, \ldots, \mathcal{B}^K\}$ may be provided as the additional information. Let $\mathcal{S} = \{(\mathbf{s}_1, y_1), (\mathbf{s}_2, y_2), \ldots, (\mathbf{s}_I, y_I)\}$ denote a set of $I$ user-item interactions and their corresponding labels, an instance of which can be represented as,

$$\mathbf{s}_i = \left[u^i, v^i, \mathbf{A}_{u^i}, \mathbf{B}_{v^i}, \mathbf{C}^i\right], \tag{1}$$

where $u^i \in \mathcal{U}$, $v^i \in \mathcal{V}$ and $\mathbf{C}^i \subset C$ denote the user, item, and context involved in the $i$th instance, and $\mathbf{A}_{u^i} \subset \mathcal{A}$ and $\mathbf{B}_{v^i} \subset \mathcal{B}$ are a list of attributes associated with $u^i$ and $v^i$, respectively. These information are usually encoded as a one-hot or multi-hot vector in practice, and an encoding example of the item ID, item attributes and contextual information are shown as follows,

$$\underbrace{[0, 0, \ldots, 1, 0]}_{v:\,ItemID} \qquad \underbrace{[1, 1, \ldots, 0]}_{\mathbf{B}_v:\,Brand\&Price} \qquad \underbrace{[1, 1, \ldots, 1, 0]}_{\mathbf{C}:\,Year\&Month\&Day} \tag{2}$$

The user ID and user attributes have a similar encoding process, which are thus omitted for simplicity. The goal of context-aware recommendation is to accurately predict an item $v$ that is most likely to be interacted by a user $u$ under a context $\mathbf{C}$, where the integration of the contextual information is crucial. However, both the contextual information and the interactions between the users (or items) and context may suffer from the sparsity challenge.

## 3.2 Base Model

Previous works have shown that factorization machine (FM) and its variants are promising solutions for context-aware recommendation due to their effectiveness and efficiency [4, 8, 34]. Therefore, we refer to factorization machine as the base model in this paper, which will be combined with our framework as a downstream recommendation model. Note that we will also analyze the compatibility of our framework with other types of recommendation models in the experiments.

*3.2.1 Initial Embedding.* As described in Eq.(2), an instance $\mathbf{s}_i$ in context-aware recommendation is usually represented in a sparse high-dimensional binary form. We first need to apply an embedding layer to compress the input into a dense low-dimensional real valued form, where the embedding layer contains an embedding table associated with the feature values. For a one-hot vector $u$ (or $v$), we can obtain a single embedding representation vector $\mathbf{e}_u$ (or $\mathbf{e}_v$), and for a multi-hot vector $\mathbf{A}_u$ (or $\mathbf{B}_v$, $\mathbf{C}$), a list of embedding representation vectors $\mathbf{e}_{\mathbf{A}_u}$ (or $\mathbf{e}_{\mathbf{B}_v}$, $\mathbf{e}_{\mathbf{C}}$ ) can be obtained. In this work, we focus on leveraging graph embedding learning to better model the complex interactions among the users, items and contextual features and refine their representations, rather than exploring fine-grained features of users or items. Therefore, we simply use average pooling to aggregate the embeddings of all the features of a user (or an item) as the representation of the node of this user (or item), i.e., $\mathbf{e}'_u = average\_pooling([\mathbf{e}_u, \mathbf{e}_{\mathbf{A}_u}])$ (or $\mathbf{e}'_v = average\_pooling([\mathbf{e}_v, \mathbf{e}_{\mathbf{B}_v}])$). Finally, these embedding representation vectors are concatenated to obtain the representation of the entire instance,

$$\mathbf{E}_{\mathbf{s}_i} = [\mathbf{e}'_{u^i}, \mathbf{e}'_{v^i}, \mathbf{e}_{\mathbf{C}^i}]. \tag{3}$$

*3.2.2 Feature Interaction.* After getting the embedding representation of an instance as input, most recommendation models include a well-designed feature interaction layer to capture the user preferences. We adopt the architecture of factorization machine [23] in the implementation,

$$\hat{y}(\mathbf{s}_i) = \sigma(b_g + \sum b_\star + \frac{1}{2}[(\sum \mathbf{e}_\star)^2 - \sum \mathbf{e}_\star^\top \mathbf{e}_\star]), \tag{4}$$

where $\star \in \{u^i, v^i, \mathbf{C}^i\}$, $b_g$ is the global bias, $b_\star$ is the feature bias term, and $\sigma(\cdot)$ is the sigmoid activation function.

*3.2.3 Model Training.* We use the point-wise log loss that is widely used in recommender systems as the objective function,

$$\mathcal{L} = -\frac{1}{|\mathcal{S}'|} \sum_{(\mathbf{s}_i, y) \in \mathcal{S}'} y_i \log \hat{y}(\mathbf{s}_i) + (1 - y_i) \log(1 - \hat{y}(\mathbf{s}_i)), \tag{5}$$

where $\mathcal{S}' = \mathcal{S} \cup \mathcal{S}^-$, and $\mathcal{S}^-$ is a set of negative instances randomly selected for each positive instance in $\mathcal{S}$ from a candidate set of items that the corresponding user has not interacted with under the same context.

# 4 USER-EVENT GRAPH EMBEDDING LEARNING

As mentioned in Sec. 2, most existing context-aware recommendation methods focus on improving the feature interaction layer, but overlook the feature embedding layer. However, a feature embedding layer with random initialization often suffers from the two sparsity challenges mentioned earlier, especially for the contextual features that are critical for this task. To further improve the performance of context-aware recommendation, in this paper, we propose a user-event graph (UEG) to better model complex interactions among the users, items and contextual features, and integrate and leverage graph embedding learning based on the base model to refine their embedding vectors. We coin the framework as user-event graph embedding learning (UEG-EL). Our UEG-EL consists of three modules, including graph construction, user-event collaborative graph convolution, and recommendation using the obtained refined embedding vectors. In this section, we describe each module in detail along the training pipeline. We illustrate the architecture of our proposed framework in Figure 2.

## 4.1 Graph Construction

*4.1.1 Personal Graph.* As the most relevant work to ours, the graph structure adopted by GCM [30] can be viewed as a personal graph $\mathcal{G}_{pg} = \langle \mathcal{V}_{pg}, \mathcal{E}_{pg} \rangle$ , which organizes a user's historical behaviors centered on the node representing himself or herself as shown on the left side of Figure 2. The nodes include the user ID and user attributes, as well as the interacted items and their corresponding attributes, i.e., $\mathcal{V}_{pg} = \{u\} \cup \mathbf{A}_u \cup \mathcal{V} \cup \mathcal{B}$. In addition to the association between the IDs and the attributes, the edges also include interactions between the user and the items, and the temporal relationships between items, i.e., $\mathcal{E}_{pg} = \mathbf{E}_{ua} \cup \mathcal{E}_{vb} \cup \mathcal{E}_{uv} \cup \mathcal{E}_{vv}$. Note that, unlike the user-item bipartite graph commonly used in previous recommendation models, a list of contextual features are used as edge features on a user's interactions with items. In addition, a personal graph can be further combined with external knowledge graphs to form the personal knowledge graph [35].

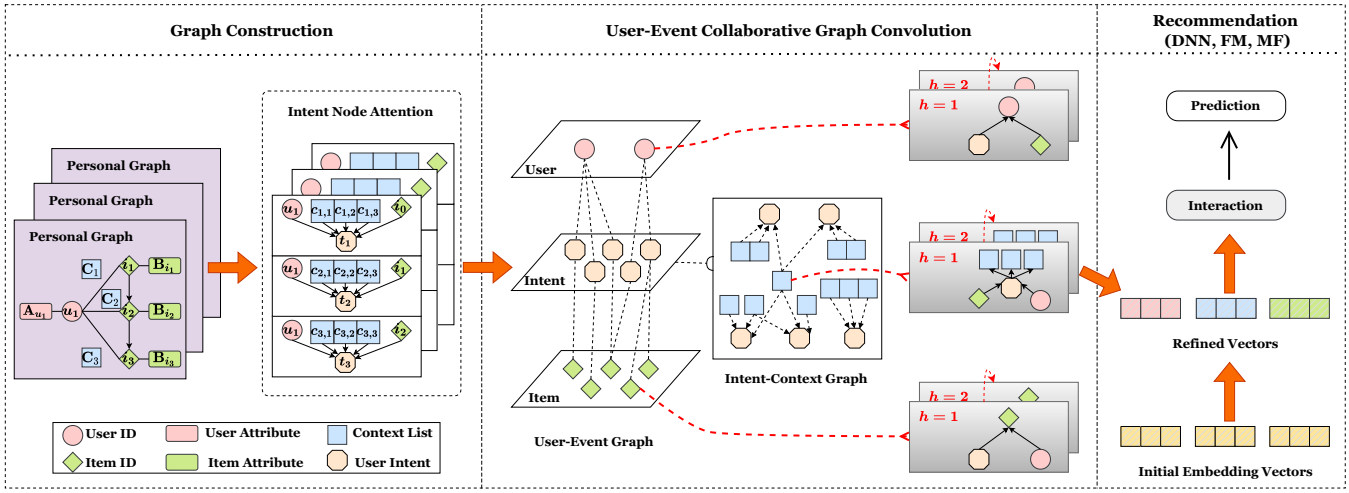



**Figure 2: The architecture of the user-event graph embedding learning (UEG-EL) framework consists of three modules: 1) the graph construction module is used to construct the user-event graph, where intent node attention (INA) is used to obtain the required intent nodes from the original personal graph; 2) the user-event collaborative graph convolution module is used to learn the refined embeddings of the users, items and contextual features; and 3) the recommendation module receives the refined feature embeddings to improve the performance of a downstream recommendation model. Note that the length of the context list is assumed to be 3.**

*4.1.2 User-Event Graph.* We argue that modeling the contextual features as edge features between users and items may limit the recommendation performance. On one hand, the contextual features cannot benefit from the information propagation process of graph embedding learning as users and items do. On the other hand, it is difficult to accurately capture a user's intent, i.e., to identify the subset from the current contextual features that triggers the user's interaction event.

To address the above problems, we first propose a new graph structure called user-event graph (UEG) for context-aware recommendation. Specifically, to construct the user-event graph, we first propose intent node attention (INA) to capture the user intent in each instance and generate some additional intent nodes $\mathcal{T} = \{t^1, t^2, \ldots, t^I\}$. Let the list of contextual features for an instance be denoted as $\mathbf{C}^i = \{c_1^i, c_2^i, \ldots, c_Z^i\}$, where $Z$ is the length of the contextual features in each instance. Since a user's behavior may also be influenced by the preceding behaviors rather than the context alone, we additionally consider the user's last interacted item before the current instance in INA. We use $c_{Z+1}^i$ to denote this particular item for the sake of notational brevity. The intent node attention of an instance is computed as follows,

$$\alpha_z^i = \text{Softmax}(\mathbf{W}_0^\top \text{Relu}(\mathbf{W}_1 \mathbf{e}'_{u^i} + \mathbf{W}_2 \mathbf{e}_{c_z^i} + \mathbf{b}_1)). \tag{6}$$

$$\mathbf{e}_{t^i} = \sum_{z=1}^{Z+1} \alpha_z^i \mathbf{e}_{c_z^i}, \tag{7}$$

where $\mathbf{W}_0 \in \mathbb{R}^{d\times 1}, \mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{d\times d}, \mathbf{b}_1 \in \mathbb{R}^{d\times 1}$ are trainable parameters, $d$ is the embedding size, and $\mathbf{e}_{t^i}$ is the embedding representation of the intent node corresponding to this instance. An illustration of INA can be found on the left side of Figure 2. Based on the obtained intent nodes, we can then build a user-event graph $\mathcal{G}_{ueg} = \langle \mathcal{U} \cup \mathcal{V} \cup \mathcal{T}, \mathcal{E}_{ut} \cup \mathcal{E}_{vt} \rangle$ shown in the middle of Figure 2. The intent nodes explicitly model a user's intent to better capture the user's preferences over different contextual features. Additionally, it acts as a hub to establish connections between users and items, which helps model complex interactions among the users, items, and contextual features.

*4.1.3 Intent-Context Graph.* To propagate the information in graph embedding learning to contextual features, there is also an intent-context graph $\mathcal{G}_{icg} = \langle C \cup \mathcal{T}, \mathcal{E}_{ct} \rangle$ on the intent layer. Specifically, each intent node is associated with $Z$ contextual features, which are used to compute this intent node in INA. Furthermore, each edge between an intent node and a contextual feature has an intent attention as a weight, i.e., $\alpha_z^i$ in Eq.(6). This means that if a contextual feature has a large weight in the generation of this intent node, it will get more emphasis when the information is propagated.

*4.1.4 Remarks.* Our proposed user-event graph is clearly different from user-item bipartite graphs [13, 27] and personal graphs [30] previously applied to recommender systems. By using contextual features to model user intent nodes, which in turn act as hubs to generate connections between users and items, we expect to obtain a better solution for context-aware recommendation. As we will show in the experiments, the proposed user-event graph has a significant advantage over the above graph structures. In addition, the user-event graph is also potentially helpful to other recommendation tasks via some extensions. For example, we can integrate user behavior types or data from different domains as a special contextual feature into the user-event graph.

## 4.2 User-Event Collaborative Graph Convolution

Existing graph embedding learning techniques are not applicable to our user-event graph due to its peculiar structure, i.e., the intent

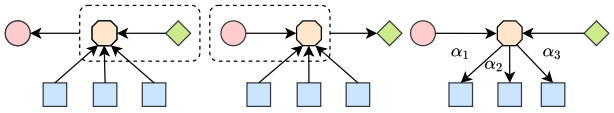



**Figure 3: An illustration of information propagation for the nodes of users, items and contextual features in user-event collaborative graph convolution.**

nodes. Therefore, in this section we introduce the proposed user-event collaborative graph convolution to exploit the power of the user-event graph. Specifically, we aim to fully exploit the pivotal role of the intent nodes to explore the connections among the users, items, and contextual features, especially to identify a focused subset of users in contextual features. This in turn feeds back into intent node attention to get a more accurate node embedding, i.e., the first two modules of our framework are synergistic. The information propagation for different nodes in user-event collaborative graph convolution is illustrated in Figure 3.

*4.2.1 Information Propagation for the Users.* For a user, we expect that it will be fully explored with the item and context. Therefore, as shown on the left side of Figure 3, we use the intent nodes as a hub to propagate the information about the items and contextual features to the users,

$$\mathbf{p}_{u^i,i}^{(h)} = \mathbf{p}_{v^i}^{(h-1)} + \mathbf{p}_{t^i}^{(h-1)}, \tag{8}$$

where $\mathbf{p}_{u^i,i}^{(h)}$ is the information representation passed to the user associated with the $i$th instance in layer $h$ of graph convolution, $\mathbf{p}_{v^i}^{(h-1)}$ and $\mathbf{p}_{t^i}^{(h-1)}$ are the item embedding and intent node embedding at layer $h-1$, respectively, and $\mathbf{p}_{v^i}^{(0)} = \mathbf{e}'_{v^i}, \mathbf{p}_{t^i}^{(0)} = \mathbf{e}_{t^i}$. For each user $u$, we then aggregate the information about all instances associated with it to get the embedding of layer $h$,

$$\mathbf{p}_u^{(h)} = \frac{1}{\sqrt{|\{i|u^i = u\}|}} \sum_{i,u^i=u} \mathbf{p}_{u,i}^{(h)}. \tag{9}$$

The intuition behind Eq.(9) is to capture which type of items a user $u$ will interact with under which contextual features. Finally, a refined user embedding can be obtained by averaging the embeddings of each layer,

$$\hat{\mathbf{p}}_u = \frac{1}{H+1} \sum_{h=0}^{H} \mathbf{p}_u^{(h)}. \tag{10}$$

*4.2.2 Information Propagation for the Items.* Similarly, as shown in the middle of Figure 3, we use the intent nodes as a hub to propagate the information about the user and contextual features to the items,

$$\mathbf{p}_{v^i,i}^{(h)} = \mathbf{p}_{u^i}^{(h-1)} + \mathbf{p}_{t^i}^{(h-1)}, \tag{11}$$

where $\mathbf{p}_{u^i}^{(h-1)}$ is the user embedding at layer $h-1$. The embedding of an item $v$ at layer $h$ can be obtained by aggregating the information of all instances associated with it,

$$\mathbf{p}_v^{(h)} = \frac{1}{\sqrt{|\{i|v^i = v\}|}} \sum_{i,v^i=v} \mathbf{p}_{v,i}^{(h)}. \tag{12}$$

We design this equation to capture which type of users interact with item $v$ under which contextual features. We then obtain a refined item embedding as follows,

$$\hat{\mathbf{p}}_v = \frac{1}{H+1} \sum_{h=0}^{H} \mathbf{p}_v^{(h)}. \tag{13}$$

*4.2.3 Information Propagation for the Context.* To propagate the user and item information to the contextual features, as shown on the right side of Figure 3, the information is first propagated to an intent node,

$$\mathbf{p}_{t^i,i}^{(h)} = \mathbf{p}_{u^i}^{(h-1)} + \mathbf{p}_{v^i}^{(h-1)}. \tag{14}$$

Then, the contextual features receive different information sent by the intent nodes according to the attention distribution,

$$\mathbf{p}_{c_z^i,i}^{(h)} = \alpha_z^i \mathbf{p}_{t^i,i}^{(h)}. \tag{15}$$

Finally, the embedding of the contextual features at layer $h$ and the final refined embedding can be obtained,

$$\mathbf{p}_{c_z}^{(h)} = \frac{1}{\sqrt{|\{i|c_z^i = c_z\}|}} \sum_{i,c_z^i=c_z} \mathbf{p}_{c_z^i,i}^{(h)}, \tag{16}$$

$$\hat{\mathbf{p}}_{c_z} = \frac{1}{H+1} \sum_{h=0}^{H} \mathbf{p}_{c_z}^{(h)}. \tag{17}$$

Note that after getting $\mathbf{p}_{c_z}^{(h)}$, we need to feed the information back to the intent node to get the embedding of the intent node at layer $h+1$, which will be used in Eq.(8) and Eq.(11),

$$\mathbf{p}_{t^i}^{(h+1)} = \sum \alpha_z^i \mathbf{p}_{c_z^i}^{(h)}. \tag{18}$$

The intuition behind Eq.(16) is capturing the affinities between different users (and items) and specific contextual features. Comparing with Eq.(3), after performing user-event collaborative graph convolution, we can obtain a set of corresponding refined embeddings,

$$\mathbf{P}_{\mathbf{s}_i} = [\hat{\mathbf{p}}_{u^i}, \hat{\mathbf{p}}_{v^i}, \hat{\mathbf{p}}_{\mathbf{C}^i}]. \tag{19}$$

*4.2.4 Pruning the Information Propagation of Context.* We find in practice that the above information propagation for contextual features may have a limitation. For example, when the contextual features are associated with too many instances, aggregating the information of all instances may suffer from noise. Therefore, we further propose a simple but effective variant, i.e., UEG-EL-V, by pruning the information propagation of the context. Specifically, for each context $c_z$, we first obtain a mean vector of the intent node embeddings for all instances associated with it. Then, we compute the distances of these instances from the mean vector, and use a preset pruning rate $\theta$ to remove those instances with larger distances. The idea behind this pruning operation is that instances with a different intent than the majority driven by this contextual feature are more likely to be noise.

## 4.3 Complexity Analysis

In this subsection, we analyze the time complexity of our UEG-EL. Since the refined embeddings can be obtained in offline training and be directly used for online inference, the time complexity of our UEG-EL in this case is the same as that of the base model. For model training, user-event collaborative graph convolution dominates the time cost, and its computational complexity is $O(Z \cdot |\mathcal{G}_{ueg}| \cdot d)$, where $|\mathcal{G}_{ueg}|$ denotes the number of edges in $\mathcal{G}_{ueg}$. Compared with

that of a bipartite graph model commonly used in recommender systems, the complexity is linear with the number of the edges between the intent nodes and contextual features. Therefore, to speed up model training, a pre-filtering of the contextual feature set, and a pruning operation similar to that described in Sec. 4.2.4, are necessary.

# 5 EMPIRICAL EVALUATIONS

In this section, we conduct experiments with the aim of answering the following six key questions. Note that the source codes are available at https://github.com/dgliu/KDD22_UEG.

- RQ1: How does our UEG-EL perform compared to the baselines?
- RQ2: What is the role of each module in our UEG-EL?
- RQ3: What is the effect of our UEG-EL variant (i.e., UEG-EL-V)?
- RQ4: How is the compatibility of our UEG-EL?
- RQ5: What are the characteristics of intent attention obtained in our UEG-EL?
- RQ6: Does our UEG-EL alleviate the two sparsity challenges of context-aware recommendation?

## 5.1 Experimental Setup

*5.1.1 Datasets.* Following the settings of a previous work [30], we conduct experiments on three public datasets including Yelp-NC, Yelp-OH and Amazon-Book[1]. Yelp-NC and Yelp-OH are the records of North Carolina (NC) and Ohio States (OH), respectively, in the Yelp dataset[2], which contains a large number of user reviews of local businesses. Amazon-Book is a subset of the Amazon dataset[3], which includes numerous user interactions with book products.

*5.1.2 Dataset Preprocessing.* In Yelp-NC and Yelp-OH, each user contains two attributes, i.e., $average\_stars$ and $yelping\_year$. Each item has three attributes, i.e., $city$, $scores$ and $is\_open$. There are also four kinds of context, for where and when an interaction occurs, i.e., $city$, $month$, $hour$, and $day\_of\_week$ (DoW). In the Amazon-Book dataset, the item attributes are $price$ and $brand$, and contextual features include $year$, $month$, $hour$, and $day\_of\_week$. For each dataset, we further remove a user's records if the number of records is smaller than 10. For each item sequence of a user, we take the last interaction as the test data, and the remaining as the training and validation data. We summarize the statistics of the three processed datasets in Table 1.

**Table 1: Statistics of the processed 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 Attribute | 24 | 24 | - |
| #Item Attribute | 68 | 213 | 24,816 |
| #Contextual Feature | 206 | 350 | 69 |

*5.1.3 Baselines.* We choose the representative methods among the two lines of context-aware recommendation summarized in Sec. 2. For the first line, we use the basic matrix factorization (MF) [15] and factorization machine (FM) [23] as our baselines since previous works have shown that tensor factorization based methods are generally less efficient and effective [34]. It is also used as an additional base model in Sec. 5.5 to evaluate the compatibility of the proposed framework. For the second line, we first choose two representative neural network-based factorization machine methods as our baselines, i.e., NFM [12] and xDeepFM [18]. We then focus on the sub-line using graph neural networks, which is more relevant to our work. We choose three recent representative methods as our baselines, i.e., LightGCN [13], GIN [16] and GCM [30], where LightGCN is an important reference for using graph neural networks in recommendation, GIN builds an item-item graph and incorporates context to better capture user preferences, and GCM is the most relevant method to ours and is thus taken as a most important baseline.

*5.1.4 Evaluation Metrics.* We evaluate the recommendation performance via two widely used ranking-oriented metrics, i.e., hit ratio (HR@$k$) and normalized discounted cumulative gain (NDCG@$k$). We report the results with $k$ set to 10 and 50 [30]. The candidate items to be recommended for a user are from the set of items that have not been interacted by the user.

*5.1.5 Implementation Details.* We implement our UEG-EL and its variant in TensorFlow 1.15. For all baselines, we use the open-source implementation and parameter settings provided in [30], where the embedding size is set to 64, the batch size is set to 2048, the learning rate is set to 0.001, and Adam is used as the optimizer. For our method, we tune the number of GNN layers $H$ in the range of $\{1, 2, 3\}$, the L2 regularization term in the range of $\{1e^{-1}, 1e^{-2}, 1e^{-3}, 1e^{-4}, 1e^{-5}\}$, and the pruning rate $\theta$ in the range of $\{0.1, 0.3, 0.5, 0.7, 0.9\}$, and set the other parameters the same as the baselines. We perform grid search to tune the hyper-parameters by evaluating the summation of HR@10 and NDCG@10. To avoid over-fitting, we also adopt an early stopping strategy with the patience set to 50 times.

## 5.2 RQ1: Performance Comparison

We report the comparison results in Table 2. From the results in Table 2, we can have the following observations: 1) FM-based methods can capture the relationship between users, items and contexts, and achieve better performance than MF, which indicates the key role of contextual features in CARS; and 2) For GNN-based methods, LightGCN has a better result than MF, which shows the advantage of graph neural network in capturing user preferences. GIN mines user intent based on co-occurrence item graph and achieves a better result, indicating that the modeling of user intent is helpful for understanding user preferences. By introducing contextual features into the bipartite graph as edge features, GCM performs the best among the baselines. This means that a reasonable graph structure for context-aware recommendation can contribute a better performance. Our UEG-EL consistently outperforms all baselines. This demonstrates the effectiveness of the proposed intent node attention (INA) and user-event graph. In particular, our UEG-EL can be seen as an integration and further improvement of all the above beneficial observations. We also note that our UEG-EL has a

[1]https://github.com/wujcan/GCM
[2]https://www.yelp.com/dataset
[3]http://jmcauley.ucsd.edu/data/amazon/

**Table 2: Results on all datasets, where the best and second best results are marked in bold and underlined, respectively. Note that * indicates a significance level of $p \leq 0.05$ based on two-sample t-test between our method and the best baseline.**

| Dataset | Yelp-NC | | | | Yelp-OH | | | | Amazon-Book | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metrics | HR@10 | HR@50 | NDCG@10 | NDCG@50 | HR@10 | HR@50 | NDCG@10 | NDCG@50 | HR@10 | HR@50 | NDCG@10 | NDCG@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 |
| 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 | 0.0587 | 0.1477 | 0.0323 | 0.0514 | 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 |
| LightGCN | 0.0499 | 0.1394 | 0.0241 | 0.0431 | 0.0518 | 0.1520 | 0.0249 | 0.0461 | 0.0543 | 0.1466 | 0.0274 | 0.0473 |
| GIN | 0.0866 | 0.2175 | 0.0449 | 0.0722 | 0.2304 | 0.4965 | 0.1238 | 0.1818 | 0.0939 | 0.2189 | 0.0502 | 0.0774 |
| GCM | 0.1042 | 0.2451 | 0.0546 | 0.0850 | 0.2584 | 0.5147 | 0.1428 | 0.1990 | 0.0983 | 0.2222 | 0.0550 | 0.0819 |
| UEG-EL | **0.1067*** | 0.2470 | 0.0570 | 0.0875 | 0.2656 | 0.5350 | 0.1481 | **0.2073*** | 0.0992 | 0.2385 | 0.0555 | 0.0857 |
| UEG-EL-V | 0.1062 | **0.2476*** | **0.0572*** | **0.0878*** | **0.2706*** | **0.5354*** | **0.1484*** | 0.2063 | **0.1112*** | **0.2555*** | **0.0623*** | **0.0936*** |

**Table 3: Results of the ablation studies on all datasets, where the best results are marked in bold.**

| Dataset | Yelp-NC | | | | Yelp-OH | | | | Amazon-Book | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metrics | HR@10 | HR@50 | NDCG@10 | NDCG@50 | HR@10 | HR@50 | NDCG@10 | NDCG@50 | HR@10 | HR@50 | NDCG@10 | NDCG@50 |
| UEG-EL | **0.1067** | **0.2470** | **0.0570** | **0.0875** | **0.2656** | **0.5350** | **0.1481** | **0.2073** | **0.0992** | **0.2385** | **0.0555** | **0.0857** |
| w/o CGC ($H$=1) | 0.1012 | 0.2238 | 0.0544 | 0.0810 | 0.2567 | 0.5104 | 0.1398 | 0.1950 | 0.0983 | 0.2211 | 0.0551 | 0.0818 |
| w/o CGC ($H$=2) | 0.0901 | 0.2268 | 0.0474 | 0.0769 | 0.2147 | 0.4219 | 0.1158 | 0.1608 | 0.0860 | 0.1954 | 0.0481 | 0.0717 |
| w/o CGC, INA ($H$=1) | 0.0974 | 0.2252 | 0.0514 | 0.0788 | 0.2489 | 0.5075 | 0.1358 | 0.1920 | 0.0817 | 0.1940 | 0.0450 | 0.0693 |
| w/o CGC, INA ($H$=2) | 0.1042 | 0.2451 | 0.0546 | 0.0850 | 0.2584 | 0.5147 | 0.1428 | 0.1990 | 0.0983 | 0.2222 | 0.0550 | 0.0819 |

relatively small improvement on Amazon-Book with a large number of instances and fewer contextual features, and our UEG-EL-V has a significant performance gain. This validates the observations described in Sec. 4.2.4 and the effectiveness of the proposed pruning method in our UEG-EL-V.

## 5.3 RQ2: Ablation Study of UEG-EL

To analyze the contribution of intent node attention (INA) and information propagation of context nodes in user-event collaborative graph convolution (CGC) in our UEG-EL, we conduct an ablation study and report the results in Table 3. We test the performance of our UEG-EL without CGC (denoted as 'w/o CGC'), UEG-EL without CGC and INA (denoted as 'w/o CGC, INA', i.e., GCM). We have the following observations: 1) **'w/o CGC ($H$=1)' vs. 'w/o CGC, INA ($H$=1)'**. UEG-EL without CGC beats UEG-EL without CGC and INA, indicating that our proposed INA can well capture the different user intents on each sample. The user intent will enter the user node and item node of the user-event graph via information propagation, making the refined vectors of the users and items more in line with user intent. 2) **'w/o CGC ($H$=2)' vs. 'w/o CGC ($H$=1)'**. UEG-EL without CGC using 2-layer GNN will be weaker than 1-layer, which means that the traditional convolution method is not suitable for the case with INA. The reason why the higher-order information cannot be accurately modeled in higher-order aggregation may be that the learning of attention becomes complicated. However, UEG-EL without CGC and INA still performs well when $H$=2, which motivates us to design a new convolution mode, i.e., our CGC. 3) **UEG-EL vs. 'w/o CGC, INA ($H$=2)', 'w/o CGC ($H$=2)'**. UEG-EL achieves the best results, which means that the proposed new convolution mode CGC can help INA to capture user intent better, i.e., our UEG-EL is synergistic. In particular, CGC can enable a context node to obtain effective high-order collaborative information on the basis of INA.

## 5.4 RQ3: Effectiveness of UEG-EL-V

In this subsection, we explore the effect of different values of the pruning rate $\theta$ on our UEG-EL-V, and show the results in Figure 4. From Figure 4, we can see that as the pruning rate increases, the effect of our UEG-EL-V on Yelp-NC and Yelp-OH gradually decreases. The reason is that they have a relatively small number of samples, causing the pruning operation to exclude a lot of context and our UEG-EL-V can thus not learn the appropriate high-order collaborative information well. On Amazon-Book, the performance improves as the pruning rate increases, indicating that our proposed pruning method can effectively alleviate the influence of noise in aggregating information for contextual features when a large number of instances are available. On all datasets, we find that when the pruning rate increases, the evaluation time becomes shorter, which is a merit of a typical pruning strategy in improving the efficiency.

## 5.5 RQ4: Compatibility Evaluation of UEG-EL

To verify the improvement of our UEG-EL on different downstream recommendation models, we use three typical recommendation models, i.e., MF, FM and MLP, in our experiments. We also compare GCM as it is the most relevant method to ours. The results are shown in Figure 5. Our UEG-EL outperforms the base downstream model that uses the initial embedding vector of user, item and context in all cases and outperforms GCM in most cases. This suggests that our UEG-EL can be used as a general framework to improve the performance of different downstream recommendation models. And unlike GCM, the proposed user-event graph can refine the contextual feature embeddings, which is beneficial to downstream models. Note that 'UEG-EL+MF' does not exceed 'GCM+MF' on Amazon-Book. The reason is that MF only uses the refined vectors of the users and items, and if we do not use the refined vectors of the context in a downstream model, the user intent in our INA may

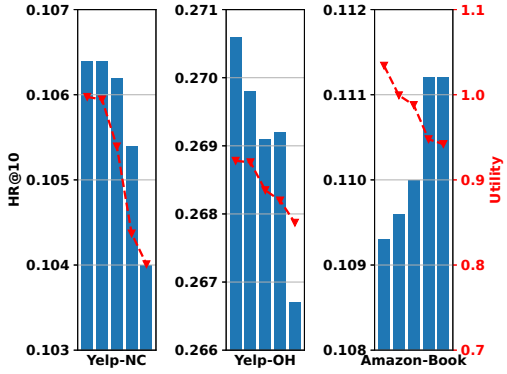
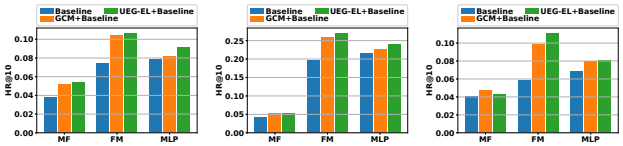



**Figure 4: Recommendation performance of UEG-EL-V and the ratio of its computation time compared with that of UEG-EL, where the values on the x-axis (i.e., the pruning rates) are** 0.1, 0.3, 0.5, 0.7 **and** 0.9**, respectively.**

not be learned well. In contrast, both our 'UEG-EL+FM' and 'UEG-EL+MLP' exceed 'GCM+FM' and 'GCM+MLP' when the refined vectors of the context are used.



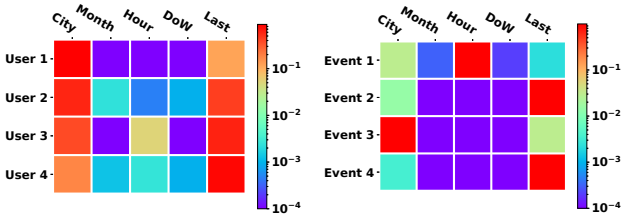

(a) Yelp-NC (b) Yelp-OH (c) Amazon-Book

**Figure 5: Recommendation performance of GCM and our UEG-EL with different downstream models, i.e., MF, FM and MLP, on the three datasets.**

## 5.6 RQ5&RQ6: In-depth Analysis of UEG-EL

In this subsection, we conduct a case study of our UEG-EL on Yelp-OH as an example. Specifically, we select four representative users and calculate each user's average attention on all the interacted records as the global user intent, which is shown in Figure 6(a). Note that we use 'last' to refer to the last item the user interacted with. We can find that the users' intents are different, e.g., user 1 and 2 pay more attention to the context '*city*', while user 3 and 4 pay more attention to the last interacted items. The temporal context '*month*', '*hour*' and '*day_of_week* (DoW)' are often of low values, but user 3 is sensitive to '*hour*'.

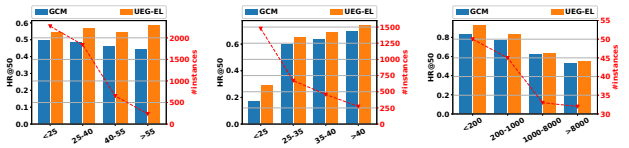
Moreover, we select some interactions of user 3 for fine-grained visualization of the intent. As shown in Figure 6(b), we can see that this user has different intents in different events. For event 1, the user is more inclined to interact at a certain moment, which is likely to be a habitual behavior. For events 2 and 4, the user chooses to interact because it is an item similar to the most recently interacted ones. For event 3, it is likely because of an item that the user interacts with in a certain city. In general, different users have different intents in different contexts, and a user's intents for the contexts vary in different interactions, which justifies the necessity of our proposed INA.




(a) global user intent of the four users (b) fine-grained intent of user 3

**Figure 6: Visualization of the attention of our UEG-EL w.r.t. some users in Yelp-OH.**

To identify the effect of our UEG on the two problems of feature sparsity and interaction sparsity, we conduct three studies about GCM and our UEG-EL on Yelp-OH. We first group all users according to the number of contextual features they had interacted with, and count the average result within each group in turn. The results are shown in Figure 7(a), and we can find that our UEG-EL has a significant improvement on all groups. In particular, when a user associates more contextual features, our UEG-EL has a better result since the proposed INA can effectively identify the attention subset of this user. Similarly, we group all items according to the number of associated contextual features and compute their average result. As shown in Figure 7(b), our UEG-EL has a more significant improvement on item groups associated with fewer contextual features. Finally, we group the contextual features by their respective frequencies, and the average results for each group are shown in Figure 7(c). We can see a bigger boost with our UEG-EL on groups with lower frequencies. The above results demonstrate that our UEG-EL can effectively alleviate these two key challenges in CARS.




(a) u-c interaction (b) i-c interaction (c) context frequency

**Figure 7: Recommendation performance of our UEG-EL with different user-context (u-c) interaction levels, item-context (i-c) interaction levels, and context frequencies on Yelp-OH.**

## 6 CONCLUSIONS AND FUTURE WORK

In this paper, we propose a novel user-event graph embedding learning (UEG-EL) framework to address two sparsity challenges suffered by a typical embedding layer with random initialization in existing context-aware recommendation models. Our UEG-EL includes three modules, i.e., a graph construction module for obtaining the user-event graph, a user-event collaborative graph convolution module for refining the embeddings of all features, and a recommendation module to improve the performance of some existing context-aware recommendation model using the refined embeddings. Finally, we conduct extensive experiments on three real-world datasets to verify the effectiveness and compatibility of our solution.

For future works, we plan to explore more robust user-event collaborative graph convolutions for massive instances. In particular, we will conduct an in-depth analysis of the information propagation mechanism of a contextual feature node to further identify its relationship to different instances. In addition, we are also interested in extending and applying the user-event graph to other recommendation tasks.

## ACKNOWLEDGMENTS

We thank the anonymous reviewers for their expert and constructive comments and suggestions, and the support of National Natural Science Foundation of China Nos. 61836005 and 62172283. We thank Mr. Wei Guo for his helpful discussions.

## REFERENCES

[1] Gediminas Adomavicius, Konstantin Bauman, Alexander Tuzhilin, and Moshe Unger. 2022. Context-aware recommender systems: From foundations to recent developments. In *Recommender Systems Handbook*. 211–250.

[2] Faisal M Almutairi, Nicholas D Sidiropoulos, and George Karypis. 2017. Context-aware recommendation-based learning analytics using tensor and coupled matrix factorization. *IEEE Journal of Selected Topics in Signal Processing* 11, 5 (2017), 729–741.

[3] Linas Baltrunas, Bernd Ludwig, and Francesco Ricci. 2011. Matrix factorization techniques for context aware recommendation. In *Proceedings of the 5th ACM Conference on Recommender Systems*. 301–304.

[4] Chong Chen, Min Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. 2020. Efficient non-sampling factorization machines for optimal context-aware recommendation. In *Proceedings of The Web Conference 2020*. 2400–2410.

[5] Zhang Chuanyan and Hong Xiaoguang. 2021. Neural graph filtering for context-aware recommendation. In *Proceedings of the 13th Asian Conference on Machine Learning*. 969–984.

[6] Xichen Ding, Jie Tang, Tracy Liu, Cheng Xu, Yaping Zhang, Feng Shi, Qixia Jiang, and Dan Shen. 2019. Infer implicit contexts in real-time online-to-offline recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2336–2346.

[7] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. 2018. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1416–1424.

[8] Tianyi Gu, Kaiwen Huang, Jie Zhang, Kai Zhang, and Ping Li. 2021. Fast convolutional factorization machine with enhanced robustness. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[9] Wei Guo, Rong Su, Renhao Tan, Huifeng Guo, Yingxue Zhang, Zhirong Liu, Ruiming Tang, and Xiuqiang He. 2021. Dual graph enhanced embedding neural network for CTR prediction. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 496–504.

[10] Wei Guo, Can Zhang, Huifeng Guo, Ruiming Tang, and Xiuqiang He. 2020. Multi-branch convolutional network for context-aware recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1709–1712.

[11] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems* 30 (2017), 1025–1035.

[12] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 355–364.

[13] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 639–648.

[14] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. 2010. Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the 4th ACM Conference on Recommender Systems*. 79–86.

[15] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[16] Feng Li, Zhenrui Chen, Pengjie Wang, Yi Ren, Di Zhang, and Xiaoyu Zhu. 2019. Graph intention network for click-through rate prediction in sponsored search. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 961–964.

[17] Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. 2019. Fi-GNN: Modeling feature interactions via graph neural networks for CTR prediction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 539–548.

[18] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xDeepFM: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1754–1763.

[19] Yahui Liu, Furao Shen, and Jian Zhao. 2019. Pairwise interactive graph attention network for context-aware recommendation. *arXiv preprint arXiv:1911.07429* (2019).

[20] Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. 2021. UltraGCN: Ultra simplification of graph convolutional networks for recommendation. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*. 1253–1262.

[21] Kevin Meehan, Tom Lunney, Kevin Curran, and Aiden McCaughey. 2013. Context-aware intelligent recommendation system for tourism. In *2013 IEEE International Conference on Pervasive Computing and Communications Workshops*. 328–331.

[22] Lei Mei, Pengjie Ren, Zhumin Chen, Liqiang Nie, Jun Ma, and Jian-Yun Nie. 2018. An attentive interaction network for context-aware recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 157–166.

[23] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 635–644.

[24] Yue Shi, Martha Larson, and Alan Hanjalic. 2013. Mining contextual movie similarity with matrix factorization for context-aware recommendation. *ACM Transactions on Intelligent Systems and Technology* 4, 1 (2013), 1–19.

[25] Changhao Song, Bo Wang, Qinxue Jiang, Yehua Zhang, Ruifang He, and Yuexian Hou. 2021. Social recommendation with implicit social influence. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1788–1792.

[26] Moshe Unger and Alexander Tuzhilin. 2022. Hierarchical latent context representation for context-aware recommendations. *IEEE Transactions on Knowledge and Data Engineering* 34, 7 (2022), 3322–3334.

[27] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 165–174.

[28] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *Proceedings of The Web Conference 2019*. 2022–2032.

[29] Ziyang Wang, Wei Wei, Gao Cong, Xiao-Li Li, Xian-Ling Mao, and Minghui Qiu. 2020. Global context enhanced graph neural networks for session-based recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 169–178.

[30] Jiancan Wu, Xiangnan He, Xiang Wang, Qifan Wang, Weijian Chen, Jianxun Lian, and Xing Xie. 2022. Graph convolution machine for context-aware recommender system. *Frontiers of Computer Science* 16, 6 (2022), 1–12.

[31] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 726–735.

[32] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A neural influence diffusion model for social recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 235–244.

[33] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. 346–353.

[34] Xin Xin, Bo Chen, Xiangnan He, Dong Wang, Yue Ding, and Joemon M Jose. 2019. CFM: Convolutional factorization machines for context-aware recommendation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 3926–3932.

[35] Yu Yang, Jiangxu Lin, Xiaolian Zhang, and Meng Wang. 2022. PKG: A personal knowledge graph for recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[36] Fajie Yuan, Guibing Guo, Joemon M Jose, Long Chen, Haitao Yu, and Weinan Zhang. 2016. Optimizing factorization machines for top-N context-aware recommendations. In *Proceedings of the 17th International Conference on Web Information Systems Engineering*. 278–293.

[37] Yong Zheng, Bamshad Mobasher, and R. Burke. 2013. The role of emotions in context-aware recommendation. In *Workshop on Human Decision Making in Recommender Systems (Decisions@RecSys' 13)*.

[38] Yong Zheng, Bamshad Mobasher, and Robin Burke. 2014. Deviation-based contextual SLIM recommenders. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. 271–280.

[39] Jin Peng Zhou, Zhaoyue Cheng, Felipe Pérez, and Maksims Volkovs. 2020. TAFA: Two-headed attention fused autoencoder for context-aware recommendations. In *Proceedings of the 14th ACM Conference on Recommender Systems*. 338–347.