MVIN: Learning Multiview Items for Recommendation

Chang-You Tai  
Academia Sinica  
Taipei, Taiwan  
johnnyjana730@gmail.com

Meng-Ru Wu  
Academia Sinica  
Taipei, Taiwan  
ray7102ray7102@gmail.com

Yun-Wei Chu  
Academia Sinica  
Taipei, Taiwan  
yunweichu@gmail.com

Shao-Yu Chu  
Academia Sinica  
Taipei, Taiwan  
shaoyu0966@gmail.com

Lun-Wei Ku  
Academia Sinica  
Taipei, Taiwan  
lwku@iis.sinica.edu.tw

ABSTRACT

Researchers have begun to utilize heterogeneous knowledge graphs (KGs) as auxiliary information in recommendation systems to mitigate the cold start and sparsity issues. However, utilizing a graph neural network (GNN) to capture information in KG and further apply in RS is still problematic as it is unable to see each item’s properties from multiple perspectives. To address these issues, we propose the multi-view item network (MVIN), a GNN-based recommendation model which provides superior recommendations by describing items from a unique mixed view from user and entity angles. MVIN learns item representations from both the user view and the entity view. From the user view, user-oriented modules score and aggregate features to make recommendations from a personalized perspective constructed according to KG entities which incorporates user click information. From the entity view, the mixing layer contrasts layer-wise GCN information to further obtain comprehensive features from internal entity-entity interactions in the KG. We evaluate MVIN on three real-world datasets: MovieLens-1M (ML-1M), LFM-1b 2015 (LFM-1b), and Amazon-Book (AZ-book). Results show that MVIN significantly outperforms state-of-the-art methods on these three datasets. In addition, from user-view cases, we find that MVIN indeed captures entities that attract users. Figures further illustrate that mixing layers in a heterogeneous KG plays a vital role in neighborhood information aggregation.

CCS CONCEPTS

- Information systems → Recommender systems.

KEYWORDS

Recommendation, Graph Neural Network, Higher-order Connectivity, Embedding Propagation, Knowledge Graph

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SIGIR ’20, July 25–30, 2020, Virtual Event, China.  
© 2020 Association for Computing Machinery.  
ACM ISBN 978-1-4503-8016-4/20/07. . . $15.00  
https://doi.org/10.1145/3397271.3401126

1 INTRODUCTION

Recommender systems (RSs), like many other practical applications with extensive learning data, have benefited greatly from deep neural networks. Collaborative filtering (CF) with matrix factorization [14] is arguably one of the most successful methods for recommendation in various commercial fields [17]. However, CF-based methods’ reliance on past interaction between users and items leads to the cold-start problem [19], in which items with no interaction are never recommended. To mitigate this, researchers have experimented with incorporating auxiliary information such as social networks [11], images [38], and reviews [43].

Among the many types of auxiliary information, knowledge graphs\(^1\), denoted as KGs hereafter, have widely been used since they can include rich information in the form of machine-readable entity-relation-entity triplets. Researchers have successively utilized KGs in applications such as node classification [6], sentence completion [13], and summary generation [15]. In view of the success of KGs in a wide variety of tasks, researchers have developed KG-aware recommendation models, many of which have benefited from graph neural networks (GNNs) [24, 27, 28, 30, 31, 34, 35] which capture high-order structure in graphs and refine the embeddings of users and items. For example, RippleNet [24] propagates users’ potential preferences in the KG and explores their hierarchical interests. Wang et al. [28] employ an KG graph convolutional network (GCN) [12], which is incorporated in a GNN to generate high-order item connectivity features. However, in these models, items look identical to all users [24, 30, 35], and using GCN with KGs still has drawbacks such as missing comparisons between entities of different layers [1].

We further give some examples that explain the user view and the entity view. Imagine some users are interested in books of the same author, and other users are interested in a certain book genre, where authorship and genre are two relations between the book and its neighborhood (author, genre type) in the knowledge base. We can say that in the real world every user has a different view of a given item. In the entity-view, item representations are defined by the entities connected to it in the KG. A sophisticated representation can be

\(^{1}\)A knowledge graph is typically described as consisting of entity-relation-entity triplets, where the entity can be an item or an attribute.
generated by incorporating smart operations of entities. For example, this paper refines it by leveraging the layer-wise entity difference to keep information from neighborhood entities. To illustrate the need for this difference feature, imagine that we seek to emphasize a new actor in a movie directed by a famous director, contrasting entities related to the famous director at the second layer to the director at the first layer will have stronger expressiveness than aggregating all of the directors he has co-worked with back him.

Overall, there are still challenges with GNN-based recommendation models: (1) user-view GNN enrichment and (2) entity-view GCN refinement. In this paper, we investigate GNN-based recommendation and propose a network that meets the above challenges. We propose a knowledge graph multi-view item network (MVIN), a GNN-based recommendation model equipped with user-entity and entity-entity interaction modules. To enrich user-entity interaction, we first learn the KG-enhanced user representations, using which combine entities and relations of a KG into continuous vector spaces.

In addition to graph neural network (GNN) based methods, there are other models that consider interaction between users and items. For recommendation, there are other models that leverage KGs, and the second category is path-based methods [9, 21, 32, 37, 41], which utilize meta paths and related user-item pairs, exploring patterns of connections among items in a KG. For instance, MCRec [9] learns an explicit representation for meta paths in recommendation. In addition, it considers the mutual effect between the meta path and user-item pairs. Compared to embedding-based methods, path-based methods use the graph algorithm directly to exploit the KG structure more naturally and intuitively. However, they rely heavily on meta paths, which require domain knowledge and manual labor to process, and are therefore poorly suited to end-to-end training [24]. We also provide the performance of the state-of-the-art path-based model [9] as a baseline for comparison.

2 RELATED WORK

For recommendation, there are other models that leverage KGs, and there are other models that consider interaction between users and items. We introduce these below.

2.1 KG-aware Recommendation Models

In addition to graph neural network (GNN) based methods, there are two other categories of KG-aware recommendation.

The first is embedding-based methods [4, 5, 10, 25, 39, 40], which combine entities and relations of a KG into continuous vector spaces and then aid the recommendation system by enhancing the semantic representation. For example, DKN [26] fuses semantic-level and knowledge-level representations of news and incorporates KG representation into news recommendation. In addition, CKE [38] combines a CF module with structural, textual, and visual knowledge into a unified recommendation framework. However, these embedding-based knowledge graph embedding (KGE) algorithms methods are more suitable for in-graph applications such as link prediction or KG completion rather than for recommendation [29]. Nevertheless, we still select [38] for comparison.

The second category is path-based methods [9, 21, 32, 37, 41], which utilize meta paths and related user-item pairs, exploring patterns of connections among items in a KG. For instance, MCRec [9] learns an explicit representation for meta paths in recommendation. In addition, it considers the mutual effect between the meta path and user-item pairs. Compared to embedding-based methods, path-based methods use the graph algorithm directly to exploit the KG structure more naturally and intuitively. However, they rely heavily on meta paths, which require domain knowledge and manual labor to process, and are therefore poorly suited to end-to-end training [24]. We also provide the performance of the state-of-the-art path-based model [9] as a baseline for comparison.

2.2 User-Item Interaction

As users and items are two major entities involved in recommendation, many works attempt to improve recommendation performance by studying user-item interaction.

For example, as a KG-aware recommendation model, Wang et al. [28] propose KGNN, which characterizes the importance of the relationship to the user. However, the aggregation method in KGNN does not consider the informativeness of entities different from the user. Hu et al. [8] propose MCRec, which considers users' different preferences over the meta paths. Nevertheless, it neglects semantic differences of relations to users. Also, they do not employ GCN and thus information on high-order connectivity is limited.

With their KG-free recommendation model, Wu et al. [33] consider that as the informativeness of a given word may differ between users, they propose NPA, which uses the user ID embedding as the query vector to differentially attend to important words and news according to user preferences. An et al. [2] consider that users typically have both long-term preferences and short-term interests, and propose LSTUR which adds user representation into the GRU to capture the user’s individual long- and short-term interests.

3 RECOMMENDATION TASK FORMULATION

Here we clarify terminology used here and explicitly formulate MVIN, the proposed GNN-based recommendation model.

In a typical recommendation scenario, the sets of users and items are denoted as $\mathcal{U} = \{u_1, u_2, \ldots\}$ and $\mathcal{V} = \{v_1, v_2, \ldots\}$, and the user-item interaction matrix $Y = \{y_{uv} | u \in \mathcal{U}, v \in \mathcal{V}\}$ is defined according to implicit user feedback. If there is an observed interaction between user $u$ and item $v$, $y_{uv}$ is recorded as $y_{uv} = 1$; otherwise $y_{uv} = 0$. In addition, to enhance recommendation quality, we leverage the information in the knowledge graph $\mathcal{G}$, which is comprised of entity-relation-entity triplets $(h, r, t) | h, t \in \mathcal{E}, r \in \mathcal{R})$.  

---

2We release the codes and datasets at https://github.com/johnnyjana730/MVIN/
We describe in detail MVIN, the proposed recommendation model, where $W$ helps MVIN not only to aggregate high-order connectivity. To improve the user-oriented performance, we split user-entity interaction, which describes how MVIN collects KG entities from person to person. Someone may think of him as a comedian recommendation, the user’s impression of actor Will Smith varies for different users, KG entities should have different information in a user-specific way. The overall process is shown in Figure 2 and Algorithm 1.

4 MVIN

We describe in detail MVIN, the proposed recommendation model, shown in Figure 1, which enhances item representations through user-entity interaction, which describes how MVIN collects KG entities information from a user perspective, and entity-entity interaction, which helps MVIN not only to aggregate high-order connectivity information but also to mix layer-wise GCN information.

4.1 User-Entity Interaction

To improve the user-oriented performance, we split user-entity interaction into user-oriented relation attention, user-oriented entity projection, and KG-enhanced user representation.

4.1.1 User-Oriented Relation Attention. When MVIN collects information from the neighborhood of the given item in the KG, it scores each relation around the item in a user-specific way. The proposed user-oriented relation attention mechanism utilizes the information of the given user, item, and relations to determine which neighbor connected to the item is more informative. For instance, some users may think the film *Iron Man* is famous for its main actor Robert Downey Jr.; others may think the film *Life of Pi* is famous for its director Ang Lee. Thus each entity of neighborhood is weighted by dependent scores $\pi_{r,e}^u$, where $u$ denotes a different user; $r,e$ denotes the relation $r$ from entity $e$ to neighboring entity $e$ (the formulation of the scoring method is given below). We aggregate the weighted neighboring entity embeddings and generate the final user-oriented neighborhood information $n$ as

$$n = \sum_{e \in N(v)} \pi_{r,e}^u e$$

$$\pi_{r,e}^u = \pi(v,e) = \frac{\exp(\pi_{r,e}^u)}{\sum_{e' \in N(v)} \exp(\pi_{r,e'}^u)}$$

To calculate the neighbor’s dependent score $\pi_{r,e}^u$, we first concatenate relation $r \in \mathbb{R}^3$, item representation $v \in \mathbb{R}^3$, and user embedding $u \in \mathbb{R}^3$, and then transform these to generate the final user-oriented score $\pi_{r,e}^u$ as

$$\pi_{r,e}^u = W_r(\text{concat}([u, r, v])) + b_r,$$

where $W_r \in \mathbb{R}^{3 \times 3}$ and $b_r \in \mathbb{R}$ are trainable parameters.

4.1.2 User-Oriented Entity Projection. To further increase user-entity interaction, we propose a user-oriented entity projection module. For different users, KG entities should have different information to characterize their properties. For instance, in a movie recommendation, the user’s impression of actor Will Smith varies from person to person. Someone may think of him as a comedian due to the film *Aladdin*, while others may think of him as an action actor due to the film *Bad Boys*. Therefore, the entity projection mechanism refines the entity embeddings by projecting each entity onto user perspective $u$, where the projecting function can be either linear or non-linear:

$$\hat{e} = W_e(e + u) + b_e$$

$$\hat{e} = \sigma(W_e(e + u) + b_e)$$

where $W_e$ and $b_e$ are trainable parameters and $\sigma$ is the non-linear activation.

Thus the user-oriented entity projection module can be seen as an early layer which increases user-entity interactions. Then, the user-oriented relation attention module aggregates the neighboring information in a user-specific way.

4.1.3 KG-Enhanced User Representation. To enhance the quality of user-oriented information received from previous sections, we enrich user representations constructed according to KG entities which incorporates user click information [24]. For example, if a user watched *I, Robot*, we find *I, Robot* is acted by Will Smith, who also acts in *Men in Black* and *The Pursuit of Happyness*. Capturing user preference information from the KG relies on consulting all relevant entities in KG and the connections between entities help us to find the potential user interests. The extraction of user preference also fits the proposed user-oriented modules; in the user’s mind, the icon of a famous actor is defined not only by the movies they have watched but also by the movies in the KG that the user is potentially interested in. In our example, if the user has potential interests in Will Smith, the modules would quickly focus on other films he acted. In sum, the relevant KG entities model the user representation and by KG-enhanced user representation, the user-oriented information is enhanced as well. The overall process is shown in Figure 2 and Algorithm 1.

Algorithm 1: KG-Enhanced User Representation

1. **KGUR** ($u$):
2. 
3. 
4. $o^u_0 \leftarrow \sum_{i \in S_u} o_0^i$;
5. $o^u = \text{concat}([o^u_0; o^u_1; \ldots; o^u_l])$;
6. $u = W_o o_u + b_o$;
7. return $u$.

Preference Set We first initialize the preference set. For user $u$, the set of items that the user has interacted with, $V_u = \{v|y_{uv} = 1\}$, is treated as the starting point in $G$, which is then explored along the relations to construct the preference set $S_u$ as

$$E^p_u = \{(h, r, t) \in G \land h \in E^p_{u-1}\}, p = 1, 2, \ldots, l_p$$

where $E^0_u = V_u$; $E^p_u$ records the $p$ hop entities linked from entities at previous $p - 1$ hop.

$$S^p_u = \{(h, r, t) | (h, r, t) \in G \land h \in E^p_{u-1}\}, p = 1, 2, \ldots, l_p$$

In Section 5.4.2, this design helps MVIN to focus on entities which the user may show interest in given the items that the user has interacted with.
Figure 1: MVIN framework, which enhances item representations through user-entity and entity-entity interaction. For user-entity interaction, it contains (a) user-oriented relation attention $UO(r)$ and entity projection modules $UO(e)$ to collect KG entity information from a user perspective. For entity-entity interaction, the mixing layer allows MVIN not only to (a) aggregate high-order connectivity information but also to (b) mix layer-wise GCN information.

Figure 2: KG-enhanced user representation in MVIN. At hop $p$, user preference set $S^p_u$ is propagated to generate user preference responses $o^p_u$, after which all hops of user preference responses are integrated to generate the KG-enhanced user representation $u$.

where $S^1_u$ is the preference set at hop 0. Note that $E^p_u$ is only tail entities and $S^p_u$ is the set of knowledge triples, $p$ represents the hop(s), and $l_p$ is the number of preference hops.

Preference Propagation The KG-enhanced user representation is constructed by user preference responses $o_u$ generated by propagating preference set $S^u_u$.

First, we define at hop 0 the user preference responses $o_u^0$ which is calculated from the user-clicked items $h_i \in S^1_u$; taking into account different items representations $v$ assigns different degrees of impact to the user preference response:

$$o_u^0 = \sum_{h_i \in S^1_u} a_i h_i$$

$$a_i = \text{softmax}(W_u[h_i, v])$$

where $W_u$ is a trainable parameter.

Second, at hop $p$, where $p > 0$, user preference responses $o_u^p$ are computed as the sum of the tails weighted by the corresponding relevance probabilities $k_i$ as

$$o_u^p = \sum_{(h_i, r_i, t_i) \in S^p_u} k_i t_i$$

$$k_i = \text{softmax}(v^T R_i h_i)$$

where $h_i \in \mathbb{R}^s$, $R_i \in \mathbb{R}^{s \times s}$, $t_i \in \mathbb{R}^s$, and $v \in \mathbb{R}^s$ are the embeddings of heads $h_i$, relations $r_i$, tails $t_i$, and item $v$. The relation space embedding $R$ helps to calculate the relevance of item representation $v$ and entity representation $h$.

After integrating all user preference responses $o_u^p$, we generate the final preference embedding of user $u \in \mathbb{R}^s$ as

$$o_u = \text{concat}(o_u^0, o_u^1, \ldots, o_u^l)$$

$$u = W_o o_u + b_o$$

where $W_o$ and $b_o$ are trainable parameters.

4.2 Entity-Entity Interaction

In entity-entity interaction, we propose layer mixing and focus on capturing high-order connectivity and mixing layer-wise information. We introduce these two aspects in terms of depth and width, respectively; the overall process, combined with the method mentioned in Section 4.1, is shown in Figure 1 and Algorithm 2.

For depth, we integrate user-oriented information obtained as described in Section 4.1, yielding high-order connectivity information to generate entity $\hat{v}^d_w$ and neighborhood information $\hat{n}^d_w$, followed
The objective function is the cross-entropy loss, $P$ is a negative sampling distribution, and $N^u$ is the number of negative samples for user $u$; $N^u = |\{v : y_{uv} = 1\}|$, and $P$ follows a uniform distribution. The second term is the L2 regularizer.

### 4.3 Learning Algorithm

The formal description of the above training step is presented in Algorithm 3. For a given user-item pair $(u, v)$ (line 2), we first generate the user representation $u$ (line 7) and item representation $v'$ (line 8), which are used to compute the click probability $\hat{y}_{uv}$ as

$$\hat{y}_{uv} = \sigma'(u^T v')$$

where $\sigma'$ is the sigmoid function.

To optimize MVIN, we use negative sampling [16] during training. The objective function is

$$L = \sum_{u \in U} \left( \sum_{v_{uv} = 1} J(y_{uv}, \hat{y}_{uv}) - \sum_{1 \leq l \leq l_d} \mathbb{E}_{v_i \sim P(v)} J(y_{uv_i}, \hat{y}_{uv_i}) \right) + \lambda \|F\|_2^2$$

(17)

For Section 4.1.1, we uniformly sample a fixed-size set of neighbors $N'(v)$ for each entity $v$, where $N'(v) \defeq \{e_i \sim N(v)\}$ and $N(v)$ denotes those entities directly connected to $v$, where $|N'(v)| = K_p$ and $K_m$ is the sampling size of the item neighborhoods and can be modified.\(^5\) Also, we do not compute the next-order entity representations for all entities $e \in G$, as shown in line 6 of Algorithm 2, and we sample only a minimal number of entities to calculate the final entity embedding $v'$. Per Section 4.1.3, at hop $p$ we sample user preferences $S^p_u = \{K_m\}$ to maintain a fixed number of relevant entities, where $|S^p_u| = K_m$ and $K_m$ is the fixed neighbor sample size, which can be modified.\(^3\)

### 4.3.2 Stage-wise Training.

To solve the potential issue that the fixed-size sampling strategy may put limitation on the use of all entities, recently stage-wise training has been proposed to collect more entity-relation from KG to approach the panoramic view of the whole neighborhood [22]. Specifically, in each stage, stage-wise training would resample another set of entities to allow MVIN to collect more entity information from KG. The whole algorithm of stage-wise training is shown in the Algorithm 3 (Line 11).

### Algorithm 3: MVIN Learning

**Input:** Interaction matrix $Y$, knowledge graph $G(E, R)$;  
**Output:** Prediction function $F(u, v|\Theta, G)$;

1. **Regular Training:**
   1. Initialize all parameters;
   2. Calculate preference set $S_u$ for each user $u$;
   3. Map neighborhood sample $N'(v)$ for each node $v$;
   4. while MVIN has not converged do
      1. for $(u, v)$ in $Y$ do
         1. $u \leftarrow \text{KGUR}(u)$;
         2. $v' \leftarrow \text{MixLayer}(u, v)$;
         3. Calculate predicted probability $\hat{y}_{uv} = f(u, v')$;
         4. Update parameters by gradient descent;
      end do;
   end while;
   5. Stage-wise Training:
      1. Initialize all parameters;
      2. Load previous embedding of $G(E, R)$;
      3. Re-sample $S_u$ and $N'(v)$ according to Eq. (3)–(4);
      4. Calculate Eq. (5)–(10);
      5. Save best embedding of $G(E, R)$;

### 4.3.3 Time Complexity Analysis.

Per batch, the time cost for MVIN mainly comes from generating KG-enhanced user representation and the mixing layer. The user representation generation has a computational complexity of $O(l_p K_m s^2)$ to calculate the relevance probability $\hat{e}_i$ for total of $l_p$ layers. The mixing layer has a computational complexity of $O(K_p^2 l_d + l_d s^2)$ to aggregate through the deep layer $l_d$ and wide layer $l_w$. The overall training complexity of MVIN is thus $O(l_p K_m s^2 + K_n^2 l_d s^2)$.

\(^4\)In Section 5.4.3, this is shown to help MVIN to improve results given large neighbor sampling sizes.

\(^5\)We discuss the performance changes when $K_p$ and $K_m$ vary.
Compared with other GNN-based recommendation models such as RippleNet, KGMCN, and KGAT, MVIN achieves a comparable computation complexity level. Below, we set their layers to $l$ and the sampling number to $K$ for simplicity. The computational complexities of RippleNet and KGMCN are $O(lKs^2)$ and $O(Kl^2s^2)$ respectively. This is at the same level as ours because $l_w, l_d$ is a special case of $l$. However, for KGAT, without the sampling strategy, its attention embedding propagation would globally update the total entities in graph, and its computational complexity is $O(l|G|j^2)$.

We conducted experiments to compare the training speed of the proposed MVIN and others on an RTX-2080 GPU. Empirically, MVIN, RippleNet, KGMCN, and KGAT take around 6.5s, 5.8s, 3.7s, and 550s respectively to iterate all training user-item pairs in the Amazon-Book dataset. We see that MVIN has a time consumption comparable with RippleNet and KGMCN, but KGAT is inefficient because of the whole-graph updates.

5 EXPERIMENTS AND RESULTS

In this section, we introduce the datasets, baseline models, and experiment setup, followed by the results and discussion.

5.1 Datasets

In the evaluation, we utilized three real-world datasets: ML-1M, LFM-1b, and AZ-book which are publicly available [24, 28, 30]. We compared MVIN with models working on these datasets coupled with various KGs, which were built in different ways. For ML-1M, its KGs were built by Microsoft Satori where the confidence level was set to greater than 0.9. The KGs of LFM-1b and AZ-book were compared with various KGs, which were built in different ways. For ML-1M, its KGs were built by Microsoft Satori where the confidence level was set to greater than 0.9. The KGs of LFM-1b and AZ-book were built by title matching as described in [42]. The statistics of the three datasets are shown in Table 1, and their descriptions are as follows:

- **MovieLens-1M** A benchmark dataset for movie recommendations with approximately 1 million explicit ratings (ranging from 1 to 5) on a total of 2,445 items from 6,036 users.
- **LFM-1b 2015** A music dataset which records artists, albums, tracks, and users, as well as individual listening events and contains about 3 million explicit rating records on 15,471 items from 12,134 users.
- **Amazon-book** Records user preferences on book products. It records information about users, items, ratings, and event timestamps. This dataset contains about half a million explicit rating records on a total of 9,845 users from 7,000 users.

We transformed the ratings into binary feedback, where each entry was marked as 1 if the item had been rated by users; otherwise, it was marked as 0. The rating threshold of ML-1M was 4; that is, if the item was rated less than 4 by the user, the entry was set to 0. For LFM-1b and AZ-book, the entry was marked as 1 if user-item interaction was observed. To ensure dataset quality, we applied a q-core setting, i.e., we retained users and items with at least $g$ interactions. For AZ-book and LFM-1b, $g$ was set to 20.

5.2 Baseline Models

To evaluate the performance, we compared the proposed MVIN with the following baselines, CF-based (FM and NFM), regularization-based (CKE), path-based (MRec), and graph neural network-based (GC-MC, KGCN, RippleNet, and KGAT) methods.

- **FM** [18] A widely used factorization approach for modeling feature interaction. In our evaluations, we concatenated IDs of user, item, and related KG knowledge as input features.
- **NFM** [7] A factorization-based method which seamlessly combines the linearity and non-linearity of neural networks in modeling user-item interaction. Here, to enrich the representation of an item, we followed [7] and fed NFM with the embeddings of its connected entities on KG.
- **GC-MC** [23] A graph-based auto-encoder framework for matrix completion. GC-MC is a GCN-based recommendation model which encodes a user-item bipartite graph by graph convolutional matrix completion. We used implicit user-item interaction to create a user-item bipartite graph.
- **CKE** [38] A regularization-based method. CKE combines structural, textual, and visual knowledge and learns jointly for recommendation. We used structural knowledge and recommendation component as input.
- **MRec** [9] A co-attentive model which requires finer meta paths, which connect users and items, to learn context representation. The co-attention mechanism improves the representations for meta-path-based context, users, and items in a mutually enhancing way.
- **KGMCN** [28] Utilizes GCN to collect high-order neighborhood information from the KG. To find the neighborhood which the user may be more interested in, it uses the user representation to attend on different relations to calculate the weight of the neighborhood.
- **RippleNet** [24] A memory-network-like approach which represents the user by his or her related items. RippleNet uses all relevant entities in the KG to propagate the user’s representation for recommendation.
- **KGAT** [30] A GNN-based recommendation model equipped with a graph attention network. It uses a hybrid structure of the knowledge graph and user-item graph as a collaborative knowledge graph. KGAT employs an attention mechanism to discriminate the importance of neighbors and outperforms several state-of-the-art methods.

5.3 Experiments

5.3.1 Experimental Setup. For MVIN, $l_p = 2$, $l_w = 1$, $l_d = 2$, $K_m = 64$, $K_n = 8$, $\lambda = 1 \times 10^{-7}$ for ML-1M; $l_p = 1$, $l_w = 1$, $l_d = 2$, $K_m = 64$, $K_n = 4$, $\lambda = 5 \times 10^{-8}$ for LFM-1b; $l_p = 2$, $l_w = 2$, $l_d = 2$, $K_m = 16$, $K_n = 8$, $\lambda = 1 \times 10^{-7}$ for AZ-book; We set function $\sigma$ as ReLU. The embedding size was fixed to 16 for all models except 32 for KGAT because it stacks propagation layers for final output. For stage-wise training, average early stopping stage
we set the depth to 2 and layer size to [16, 16]. For RippleNet, we set the number of hops to 2 and sampling size to 64 for each dataset.

Table 2: AUC and ACC results in CTR prediction on all datasets.

| Model    | ML-1M     | LFM-1b    | AZ-book   |
|----------|-----------|-----------|-----------|
|          | AUC       | ACC       | AUC       | ACC       | AUC       | ACC       |
| FM       | .9101 (-2.3%) | .8328 (-2.9%) | .9052 (-6.3%) | .8662 (-5.6%) | .7860 (-10.2%) | .7107 (-10.4%) |
| NFM      | .9167 (-1.6%) | .8420 (-1.8%) | .9301 (-3.7%) | .8825 (-3.2%) | .8206 (-6.2%) | .7474 (-5.8%) |
| CKE      | .9095 (-2.4%) | .8376 (-2.3%) | .9035 (-6.5%) | .8591 (-5.7%) | .8070 (-7.8%) | .7227 (-8.9%) |
| MRec     | .8970 (-3.7%) | .8262 (-3.6%) | .8920 (-7.6%) | .8428 (-7.5%) | .7925 (-9.4%) | .7217 (-9.1%) |
| KGNN     | .9093 (-2.4%) | .8338 (-2.7%) | .9171 (-5.0%) | .8664 (-4.9%) | .8043 (-8.1%) | .7291 (-8.1%) |
| RippleNet| .9208 (-1.2%) | .8435 (-1.6%) | .9421 (-2.5%) | .8887 (-2.5%) | .8234 (-5.9%) | .7486 (-5.7%) |
| KGAT     | .9222 (-1.2%) | .8489 (-1.0%) | .9384 (-2.8%) | .8771 (-3.7%) | .8555 (-2.2%) | .7793 (-1.8%) |
| GC-MC    | .9005 (-3.2%) | .8197 (-4.4%) | .9204 (-4.7%) | .8723 (-4.3%) | .8177 (-6.5%) | .7347 (-7.4%) |
| MVIN     | .9318* (%)  | .8573* (%)  | .9658* (%)  | .9112* (%)  | .8749* (%)  | .7935* (%)  |

Note: * indicates statistically significant improvements over the best baseline by an unpaired two-sample t-test with p-value = 0.01.

5.3.2 Experimental Results. Table 2 and Figure 3 are the results of MVIN and the baselines, respectively (FM, NFM, CKE, GC-MC, MRec, RippleNet, KGNN, KGAT), in click-through rate (CTR) prediction, i.e., taking a user-item pair as input and predicting the probability of the user engaging with the item. We adopt AUC and ACC, which are widely used in binary classification problems, to evaluate the performance of CTR prediction. For those of top-N recommendation, selecting $N$ items with highest predicted click probability for each user and choose Precision@$N$ to evaluate the recommended sets. We have the following observations:

- MVIN yields the best performance of all the datasets and achieves AUC performance gains of 1.2%, 2.5%, and 2.2% on ML-1M, LFM-1b, and AZ-book, respectively. Also, MVIN achieves outstanding performance in top-N recommendation, as shown in Figure 3.

- The two path-based baselines RippleNet and KGAT outperform the two CF-based methods FM and NFM, indicating that KG is helpful for recommendation. Furthermore, although RippleNet and KGAT achieve excellent performance, they still do not outperform MVIN. This is because RippleNet neither incorporates user click history items $h_i^t$ into user representation nor does it introduces high-order connectivities, and KGAT does not mix GCN layer information and not value user preferences when collecting KG information.

- For the other baselines KGNN and MRec, their relatively bad performance is attributed to their not fully utilizing information from user click items. In contrast, MVIN would enrich a user representation by user click items and all relevant entities in KG and then weighted the nearby entities and emphasize the most important ones. Also, KGNN only uses GCN in each layer, which does not allow contrast on neighborhood layers. Furthermore, MRec requires finer defined meta paths, which requires manual labor and domain knowledge.

- To our surprise, the CF-based NFM achieves good performance on LFM-1b and AZ-book, even outperforming the KG-aware baseline KGNN, and achieves results comparable to RippleNet. Upon investigation, we found that this is because we enriched its item representation by feeding the embeddings of its connected entities. In addition, NFM’s design involves modeling higher-order and non-linear feature interactions and thus better captures interactions between user and item embeddings. These observation conform to [30].

- The regularization-based CKE is outperformed by NFM. CKE does not make full use of the KG because it is only

number is 7, 7, 5 for ML-1M, LFM-1b and AZ-book, respectively. For all models, the hyperparameters were determined by optimizing AUC on a validation set. For all models, the learning rate $\eta$ and regularization weight were selected from $[2 \times 10^{-2}, 1 \times 10^{-2}, 5 \times 10^{-3}, 5 \times 10^{-4}, 2 \times 10^{-4}]$ and from $[1 \times 10^{-4}, 1 \times 10^{-5}, 2 \times 10^{-5}, 2 \times 10^{-7}, 1 \times 10^{-7}, 5 \times 10^{-8}]$, respectively. For MRec, to define several types of meta paths, we manually selected user-item-attribute item meta paths for each dataset and set the hidden layers as in [9]. For KGAT, we set the depth to 2 and layer size to [16, 16]. For RippleNet, we set the number of hops to 2 and the sampling size to 64 for each dataset. For KGNN, we set the number of hops to 2, 2, 1 and sampling size to 4, 8, 8 for ML-1M, AZ-book, and LFM-1b, respectively. Other hyperparameters were optimized according to validation result.

Figure 3: Precision@N results in top-N recommendation.
We compare the performance of MVIN and MVIN. We conducted an ablation study to verify the effectiveness of the user and item representations, it achieves comparably weak performance as it only utilizes a user-item bipartite graph and ignores semantic information between KG entities.

5.4 Study of MVIN

We also provide an in-depth exploration of the entity view.

5.4.1 User-Oriented Information. The ablation study results are shown in Table 3. After removing the proposed user-oriented relation attention UO(r) and user-oriented entity projection UO(e) modules, MVINw/oUO(r) and MVINw/oUO(e) perform worse than MVIN in all datasets. Thus considering user preferences when aggregating entities and relations in KG improves recommendation results.

Figure 4: Attention visualization. We compare the attention weights between a) MVINw/oUO(k) and b) MVIN. Results show that when information on user-interacted items is provided, (b) MVIN pays more attention to Kate Atkinson, the author which the user may be interested in.

5.4.2 KG-enhanced User-Oriented Information. To enhance user-oriented information, we enrich the user representation using KG information as a pre-processing step. Here, we denote MVIN without KG-enhanced user-oriented information as MVINw/oUO(k). We compare the performance of MVIN and MVINw/oUO(k). Table 3 shows that the former outperforms the latter by a large margin, which confirms that KG-enhanced user representation improves user-oriented information.

Moreover, we conducted a case study to understand the effect of KG-enhanced user-oriented information incorporated with user-entity interaction. Given the attention weights learned by MVINw/oUO(k) in Figure 4(a), user u324 puts only slightly more value on the author of Life After Life. However, Figure 4(b) shows that MVIN puts much more attention on the author when information on user-interacted items is provided. Furthermore, in Figure 4(b), we find user u324’s interacted items—v734 (One Good Turn), v941 (Behind the Scenes at the Museum), and v6410 (Case Histories)—are all written by Kate Atkinson. This demonstrates that MVIN outperforms MVINw/oUO(k) because it captures the most important view that user u324 sees: item Life After Life, a book by Kate Atkinson.

5.4.3 Mixing layer-wise GCN information. In the mixing layer, the wide part ML(w) allows MVIN to represent general layer-wise neighborhood mixing. To study the effect of ML(w), we remove the wide part from MVIN, denoted as MVINw/oML(w). Table 3 shows a drop in performance, suggesting that the mixing of features from different distances improves recommendation performance.

Figure 5: AUC improvement from MVINw/oML(w) to MVIN for different neighborhood sampling sizes Kn, where the preference set size Km is set to 16. With a large Kn, the performance gap between MVIN and MVINw/oML(w) increases, indicating the indispensability of ML(w) for large Kn.

5.4.4 Mixing layer-wise GCN information (at high neighbor sampling size Kn). It has shown that in homogeneous graphs the benefit of the mixing layer depends on the homophily level. In MVIN, the mixing layer works in KGS, i.e., heterogeneous graphs; we also investigate its effect (ML(w)) at different sampling sizes of neighborhood nodes Kn. With a large Kn, entities in KG connect to more different entities, which is similar to a low homophily level. Figure 5 shows that ML(w) is effective in heterogeneous graphs. In addition, Kn increases the performance gap between MVIN and MVINw/oML(w). We conclude that the mixing layer not only improves MVIN performance but is indispensable for large Kn.

5.4.5 High-order connectivity information. In addition to the wide part in the mixing layer, the proposed deep part allows MVIN to aggregate high-order connectivity information. Figure 3 shows that after removing the mixing layer (ML), MVINw/oML performs poorly compared to MVIN, demonstrating the significance of high-order connectivity. This observation is consistent with [27, 30, 36].

5.4.6 Stage-wise Training. Removing the stage-wise training (SW) shown by MVINw/oSW deteriorates performance, showing that stage-wise training helps MVIN achieve better performance by collecting more entity relations from the KG to approximate a

The homophily level indicates the likelihood of a node forming a connection to a neighbor with the same label.
Table 3: MVIN ablation study results. We evaluate using AUC in CTR prediction on all datasets and show the effect of the proposed methods. User-oriented modules contain entity projection (UO(e)), relation attention (UO(r)), and KG-enhanced user-oriented information (UO(k)). Mixing layer has deep (ML(d)) and wide (ML(w)) parts. Stage-wise training (SW) is used as well. * indicates statistically significant improvements by an unpaired two-sample t-test with p-value = 0.01.

| Ablation component | UO(e) | UO(r) | UO(k) | ML(w) | ML(d) | SW | AUC | LFM-1b | AZ-book |
|---------------------|-------|-------|-------|-------|-------|----|-----|--------|---------|
| N/A                 | ✔     | ✔     | ✔     | ✔     | ✔     | ✔  | .9318 | .9658  | .8739   |
| w/o UO(e)           | ✔     | ✔     | ✔     | ✔     | ✔     | ✔  | .9299 (-0.2%) | .9617 (-0.4%)* | .8672 (-0.8%)* |
| w/o UO(r)           | ✔     | ✔     | ✔     | ✔     | ✔     | ✔  | .9305 (-0.1%) | .9638 (-0.2%) | .8705 (-0.4%)* |
| w/o UO(k)           | ✔     | ✔     | ✔     | ✔     | ✔     | ✔  | .9247 (-0.7%)* | .9598 (-0.6%)* | .8573 (-1.8%)* |
| w/o ML(w)           | ✔     | ✔     | ✔     | ✔     | ✔     | ✔  | .9289 (-0.3%)* | .9621 (-0.4%)* | .8683 (-0.6%)* |
| w/o ML              | ✔     | ✔     | ✔     | ✔     | ✔     | ✔  | .9283 (-0.4%)* | .9613 (-0.5%)* | .8637 (-1.2%)* |
| w/o SW              | ✔     | ✔     | ✔     | ✔     | ✔     | ✔  | .9276 (-0.5%)* | .9567 (-0.9%)* | .8642 (-1.1%)* |

Table 5: AUC of MVIN with different preference set size $K_m$ and neighbor sampling size $K_n$.

| $K_m$ size ($K_n = 4$) | 4   | 8   | 16  | 32  | 64  |
|------------------------|-----|-----|-----|-----|-----|
| ML-1M                  | .9210 | .9247 | .9255 | .9269 | .9276 |
| LFM-1b                 | .9299 | .9368 | .9433 | .9498 | .9567 |
| AZ-book                | .8508 | .8613 | .8616 | .8642 | .8631 |

| $K_n$ size ($K_m = 16$) | 4   | 8   | 16  | 32  | 64  |
|------------------------|-----|-----|-----|-----|-----|
| ML-1M                  | .9246 | .9254 | .9258 | .9264 | .9252 |
| LFM-1b                 | .9427 | .9430 | .9433 | .9429 | .9415 |
| AZ-book                | .8590 | .8601 | .8594 | .8610 | .8593 |

Table 5: AUC of MVIN with different number of $l_p$, $l_w$, and $l_d$ hops, where $K_m$ is set to 16. For the Propagation layer, 0 hops denotes that only the user-clicked items $h^1_i$ are utilized.

| $l_p$ hops | 0   | 1   | 2   | 3   |
|------------|-----|-----|-----|-----|
| ML-1M      | .9257 | .9262 | .9233 | .9244 |
| LFM-1b     | .9317 | .9438 | .9429 | .9415 |
| AZ-book    | .8557 | .8576 | .8555 | .8572 |
| $l_w$ hops | 0   | 1   | 2   | 3   |
| ML-1M      | n/a | .9261 | .9267 | .9262 |
| LFM-1b     | n/a | .9438 | .9445 | .9447 |
| AZ-book    | n/a | .8568 | .8611 | .8618 |
| $l_d$ hops | 0   | 1   | 2   | 3   |
| ML-1M      | n/a | .9261 | .9269 | .9250 |
| LFM-1b     | n/a | .9438 | .9441 | .9440 |
| AZ-book    | n/a | .8552 | .8621 | .8613 |

Table 6: AUC of MVIN with different embedding size $s$.

| s | 4   | 8   | 16  | 32  | 64  | 128 |
|---|-----|-----|-----|-----|-----|-----|
| ML-1M | .9037 | .9217 | .9259 | .9279 | .9250 | .9247 |
| LFM-1b | .9247 | .9468 | .9538 | .9574 | .9562 | .9538 |
| AZ-book | .8353 | .8471 | .8616 | .8664 | .8598 | .8539 |

5.5 Parameter Sensitivity

Below, we investigate the parameter sensitivity in MVIN.

Preference set sample size $K_m$. Table 4 shows that the performance of MVIN improves when $K_m$ is set to a larger value, with the exception of AZ-book. MVIN achieves the best performance on AZ-book when $K_m$ is set to 32, which we attribute to its low number of user-interacted items, as shown in Table 1. That is, when there are few user-interacted items, a small $K_m$ still allows MVIN to find enough information to represent the user.

Neighborhood entity sample size $K_n$. The influence of the size of neighborhood nodes is shown in Table 4. MVIN achieves the best performance when this is set to 16 or 32, perhaps due to the noise introduced when $K_n$ is too large.

Number of preference hops $l_p$. The impact of $l_p$ is shown in Table 5. We conducted experiments with $l_p$ set to 0, that is, we only use user-clicked items $h^1_i$ to calculate user representation. The results show that when $l_p$ hop is set to 1, MVIN achieves the best performance, whereas again larger values of $l_p$ result in less relevant entities and thus more noise, consistent with [24].

Number of wide hops $l_w$ and deep hops $l_d$. Table 5 shows the effect of varying the number of the wide hops $l_w$ and deep hops $l_d$. MVIN achieves better performance when the number of hops is set to 2 over 1, suggesting that increasing the hops enables the modeling of high-order connectivity and hence enhances the performance. However, the performance drops when the number of hops becomes even larger, i.e., 3, suggesting that considering second-order relations among entities is sufficient, consistent with [20, 28].

Dimension of embedding size $s$. The results when varying the embedding size are shown in Table 6. Increasing $s$ initially boosts the performance as a larger $s$ contains more useful information of users and entities, whereas setting $s$ too large leads to overfitting.

6 CONCLUSION

We propose MVIN, a GNN-based recommendation model which improves representations of items from both the user view and the entity view. Given both user- and entity-view features, MVIN gathers personalized knowledge information in the KG (user view) and further considers the difference among layers (entity view) to ultimately
[39] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative Knowledge Base Embedding for Recommender Systems. In Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’16). ACM, New York, NY, USA, 353–362. https://doi.org/10.1145/2939672.2939673

[40] Yongfeng Zhang, Qingyao Ai, Xu Chen, and Pengfei Wang. 2018. Learning over Knowledge-Base Embeddings for Recommendation. ArXiv abs/1803.08540 (2018).

[41] Huan Zhao, Qianming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. 2017. Meta-Graph Based Recommendation Fusion over Heterogeneous Information Networks. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (Halifax, NS, Canada) (KDD ‘17). ACM, New York, NY, USA, 635–644. https://doi.org/10.1145/3097983.3098063

[42] Wayne Xin Zhao, Gaole He, Hong-Jian Dou, Jin Huang, Siqi Ouyang, and Ji-Rong Wen. 2018. KB4Rec: A Dataset for Linking Knowledge Bases with Recommender Systems. CoRR abs/1807.11141 (2018). arXiv:1807.11141 http://arxiv.org/abs/1807.11141

[43] Lei Zheng, Vahid Noroozi, and Philip S. Yu. 2017. Joint Deep Modeling of Users and Items Using Reviews for Recommendation. CoRR abs/1701.04783 (2017). arXiv:1701.04783 http://arxiv.org/abs/1701.04783