Deep Adversarial Completion for Sparse Heterogeneous Information Network Embedding

Kai Zhao, Ting Bai*†
Beijing University of Posts and Telecommunications
Beijing, China
{kaizhao,baiting}@bupt.edu.cn

Bin Wu, Bai Wang
Beijing University of Posts and Telecommunications
Beijing, China
{wubin,wangbai}@bupt.edu.cn

Youjie Zhang, Yuanyu Yang
Beijing University of Posts and Telecommunications
Beijing, China
{molo_j,yangyy}@bupt.edu.cn

Jian-Yun Nie
University of Montreal
Canada
nie@iro.umontreal.ca

ABSTRACT

Heterogeneous information network (HIN) contains multiple types of entities and relations. Most of existing HIN embedding methods learn the semantic information based on the heterogeneous structures between different entities, which are implicitly assumed to be complete. However, in real world, it is common that some relations are partially observed due to privacy or other reasons, resulting in a sparse network, in which the structure may be incomplete, and the "unseen" links may also be positive due to the missing relations in data collection. To address this problem, we propose a novel and principled approach: a **M**ulti-**V**iew **A**dversarial **C**ompletion **M**odel (**MV-ACM**). Each relation space is characterized in a single viewpoint, enabling us to use the topological structural information in each view. Based on the multi-view architecture, an adversarial learning process is utilized to learn the reciprocity (*i.e.,* complementary information) between different relations: In the generator, MV-ACM generates the complementary views by computing the similarity of the semantic representation of the same node in different views; while in the discriminator, MV-ACM discriminates whether the view is complementary by the topological structural similarity. Then we update the node's semantic representation by aggregating neighborhoods information from the syncretic views. We conduct systematical experiments[1] on six real-world networks from varied domains: AMiner, PPI, YouTube, Twitter, Amazon and Alibaba. Empirical results show that MV-ACM significantly outperforms the state-of-the-art approaches for both link prediction and node classification tasks.

CCS CONCEPTS

• Mathematics of computing → network algorithms; **• Computing methodologies → Learning latent representations**.

*Both authors contributed equally to this research.
†Corresponding author
[1] Code is available at https://github.com/1400012831/MV-ACM.

WWW '20, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380134

KEYWORDS

Heterogeneous Information Network, Adversarial Learning, Relational Completion

ACM Reference Format:
Kai Zhao, Ting Bai, Bin Wu, Bai Wang, Youjie Zhang, Yuanyu Yang, and Jian-Yun Nie. 2020. Deep Adversarial Completion for Sparse Heterogeneous Information Network Embedding. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3366423.3380134

1 INTRODUCTION

In recent years, heterogeneous information network (HIN), which contains multiple types of entities and relations, has attracted extensive attention in both academia and industry [6, 9, 29, 32]. The basic way to model the information of networks is network embedding (*i.e.,* network representation learning). It aims to represent nodes into dense representations with semantic and structure information of the networks, and shows significant performance in many downstream mining tasks, *e.g.,* community detection, link prediction and node classification [3, 13, 28, 44]. The majority of existing network embedding methods for HIN are based on the heterogeneous structure information in the single-view network [5, 9, 29]. Recently, some adversarial based methods [17, 41] generate the "unlinked" nodes as the negative samples, enabling to learn semantic-preserving representations in a robust manner.

However, in real world, it is commonly seen that some relations are partially observed due to privacy sensitive or other reasons, resulting in the sparsity problem in network. Therefore, the "unlinked" nodes may be also positive due to the missing relations in data collection. For example, Fig. 1 (a) shows an example of HIN in e-commerce systems. The HIN are composed with two kinds of nodes (*i.e.,* users and items) and three kinds of relations (*i.e.,* friendship, purchasing and browsing). For a new user or who with few purchase records, the linkages are highly sparse in the single-view network of purchasing relation (shown in Fig. 1 (b)), which will lead to the failure of learning in this semantic relation space. Fortunately, we can usually find the users may have many other types of interactions, such as the friendship and browsing behaviour. In this case, the topological structural information, *e.g.,* the linked nodes in friendship and browsing relation spaces, are

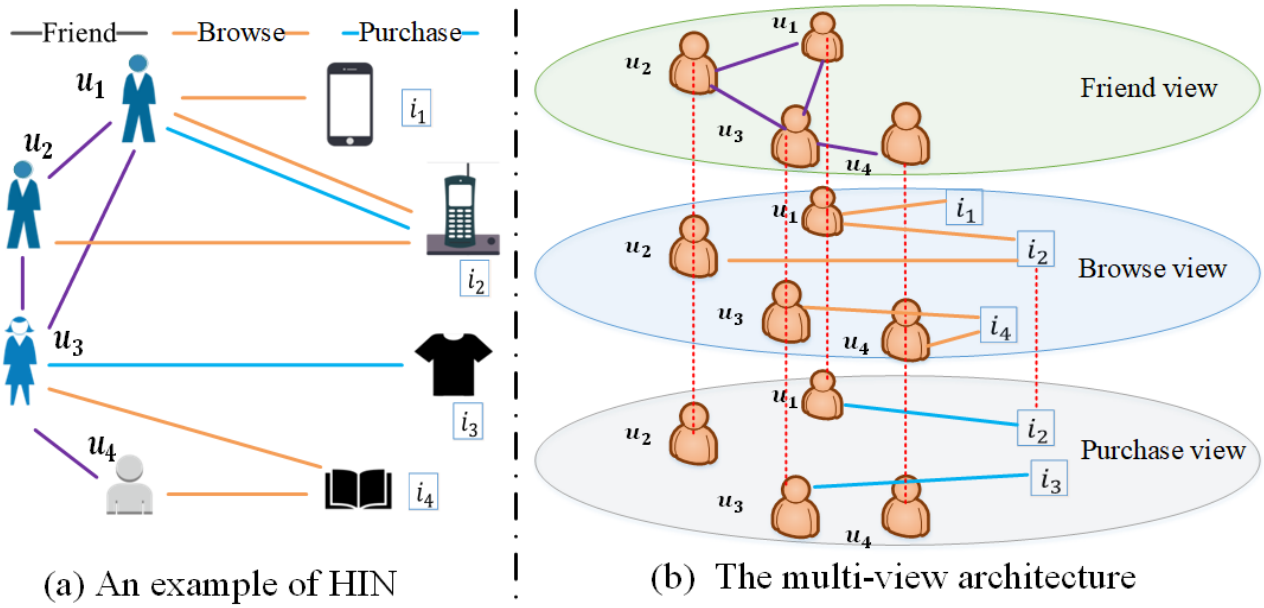

Figure 1: (a) an illustrative example of the network with both entities and relations heterogeneity. There exist two node types, *i.e.,* user and item; three edge types, *i.e.,* friendship, browse and purchase. And the different interactions between users or items may show the complementary information. (b) the corresponding multi-view network architecture, which facilitates our study on exploring the reciprocity between relations.

really promising to make a complement for learning the semantic of nodes in the purchasing space, as well as enhancing the whole network embedding.

Distinguishing such reciprocity (*i.e.,* complementary information) between relations is promising for providing more useful information. And it motivates us to address the data sparsity problem for robust and informative HIN representation learning. However, it is challenging and we find few of studies have explored this problem in literature of HIN learning. The **first challenge** is distinguishing the complementary information from different semantic spaces of HIN. In reality, different relations may have disparate topological structures, as well as the different semantic meanings for nodes. So it is challenging to distinguish the complementary information from such complex semantic spaces of relations. Many studies (*e.g.,* [9, 17, 29]) explore learning and preserving such heterogeneity in HIN, but few of them stress to distinguish the reciprocity influence between different relation spaces. Other network embedding methods [19, 42] somehow address such relation relevance problems, but they are task-specific, which need hand-engineering based on expert knowledge. The **second challenge** is maintaining the semantics of different relations in the original HIN while incorporating the reciprocity between complementary relations. The heterogeneity is an intrinsic property of complex HIN with multiple types of nodes and edges. Apart from the reciprocity, different types of nodes and edges usually fall in different semantic spaces. So it is important to simultaneously maintain such heterogeneous property while incorporating the complementary relations.

Based on the above observations, in this paper, we propose a novel and principled approach: a Multi-View Adversarial Completion Model (MV-ACM) for the HIN representation learning. In MV-ACM, each relation space is characterized in a single viewpoint, enabling us to use the topological structural information in each view to distinguish the reciprocity between relations. Based on the multi-view architecture, an adversarial learning process is utilized to learn the complementary neighborhoods in other views. The basic idea is that the more similar the network structure is, the more similar the semantic representations of the same node in different views. In particular, in generator, MV-ACM generates the complementary view by computing the similarity of the semantic representation of the same node in different views. While in discriminator, MV-ACM discriminate whether the view is complementary by the topological structural similarity. Then we update the node's semantic representation by aggregating neighborhood information from the syncretic views. To address the data sparsity problem, two tricks are specifically designed in our MV-ACM: (1) the inner-view aggregation of a node in generator, that is to say, we utilize the collaborative neighborhood information to make a complement for view-specific information, (2) a "soft samples" strategy, which means the positive nodes are sampled not only from the direct linkages in HIN, it can also be the nodes with similar topological structure in other views through our cross-view updating. Both of them can alleviate the data sparsity problem, as well as enhancing the representation learning of HIN.

Our contributions in this paper are summarized as follows:

- We emphasize the usefulness of the complementary information between different relation spaces, to alleviate the

Table 1: Notations

Notation	Explanation
G	a network
$\mathcal{V}, \mathcal{E}$	the sets of nodes/edges respectively
$\mathcal{O}, \mathcal{R}$	the sets of node/edge types respectively
$\mathcal{X}$	$\{x_i \mid v_i \in \mathcal{V}\}$ the set of nodes attributes
v_i, x_i	a node and its feature respectively
o, r	a node/edge type respectively
G_r	a view
$e_{i,j,r}$	an edge between v_i and v_j in view r
$\mathcal{N}_{i,r}$	neighbors of node v_i in view r
$\mathbf{v}_{i,r}$	the learned representation of node v_i in view r
d	the dimension of node representations
$\mathcal{G}, \mathcal{D}$	the generator/discriminator

data sparsity problem in HIN, so as to enhance the HIN embedding.

- We propose a novel multi-view adversarial completion model (MV-ACM). With semantic similarity generator and structural similarity discriminator, MV-ACM has the ability to incorporate the complementary structure information, as well as maintain the original heterogeneous semantics of network.
- We empirically evaluate our model on six real-world networks on two representative downstream tasks, *i.e.,* link prediction and node classification tasks. Experimental results show that our MV-ACM significantly outperforms the state-of-the-art baselines, especially for the sparse network.

2 PRELIMINARY

In this section, we formalize our problem and then introduce the basic background knowledge on generative adversarial learning. The important notations are summarized in Table 1.

2.1 Formalization

We first define the heterogeneous information network, the multi-view network architecture and the view as follows:

Definition 1. ***The Multi-view Network Architecture.*** *Given a heterogeneous information network (HIN) [29, 33]* $G = (\mathcal{V}, \mathcal{E}, \mathcal{O}, \mathcal{R}, \mathcal{X})$ *where* $|\mathcal{R}| \geq 2$*, we can always reorganize it as* $\{G_r \mid r \in \mathcal{R}\}$*, where* $G_r = (\mathcal{V}, \mathcal{E}_r, \mathcal{O})$*, and* $\mathcal{E}_r$ *is the set of all edges from edge type* $r \in \mathcal{R}$*. Thus we have* $\mathcal{E} = \bigcup_{r \in \mathcal{R}} \mathcal{E}_r$ *and* $G = \bigcup_{r \in \mathcal{R}} G_r$*. By this way,* $\{G_r \mid r \in \mathcal{R}\}$ *with* $\{\mathcal{X}\}$ *is a multi-view network architecture, and* G_r *is an information network with homogeneous edge type,* i.e., *a view.*

Figure 1 shows an illustrative example of heterogeneous information network with such complexities, and the corresponding multi-view network architecture. The HIN are composed with two kinds of nodes (*i.e.,* $\mathcal{O} = \{user, item\}$) and three kinds of relations (*i.e.,* $\mathcal{R} = \{friendship, purchasing, browsing\}$). Since there may exist multiple types of edges between the same pair of two nodes v_i and v_j, $e_{i,j}$ will cause ambiguity. To avoid such ambiguity, $e_{i,j,r}$ is used when referring to an certain edge, where r is the respective edge type.

Now we can formalize the problem for the **M**ulti-**V**iew based **H**eterogeneous **I**nformation **N**etwork **E**mbedding (**MV-HINE**):

Definition 2. ***MV-HINE.*** *Given* $G = (\mathcal{V}, \mathcal{E}, \mathcal{O}, \mathcal{R}, \mathcal{X})$*, we aim to learn the low-dimensional representation* $\mathbf{v}_{i,r}$ *of each node* v_i *in each view* r*,* $f_r : \mathcal{V} \rightarrow \mathbb{R}^d$ *for* $r \in \mathcal{R}$*, where* $d \ll |\mathcal{V}|$*. And the learned representations should preserve the rich semantics of both heterogeneous node types and edges types.*

2.2 Generative Adversarial Learning

Our model MV-ACM is based on the generative adversarial learning [12], which could be denoted as the theoretical game between generator $\mathcal{G}$ and discriminator $\mathcal{D}$. It can be formally defined as the minimax optimization problem:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \quad \Big\{ \mathbb{E}_{X \sim P_{data}} [log \mathcal{D}(X)] + \mathbb{E}_{Z \sim P_{noise}} \big[log\big(1 - \mathcal{D}(\mathcal{G}(Z))\big) \big] \Big\} \tag{1}$$

3 THE OVERALL FRAMEWORK

Firstly, we introduce the overall framework of our MV-ACM by addressing the two major challenges mentioned in Section 1, and leave the implementation details in Subsection 4.

1) Distinguish the complementary information from different semantic spaces of HIN. As shown in Figure 2, there are two major components competing with each other in our model, *i.e.,* the generator $\mathcal{G}$ and the discriminator $\mathcal{D}$. In order to distinguish the reciprocity between different views, our $\mathcal{G}$ and $\mathcal{D}$ interact as follows: Given a node v_i in a specific view r, the generator $\mathcal{G}(\cdot|r, v_i; \theta_{\mathcal{G}})$ tries to generate the most complementary views to, 1) fool the discriminator, 2) complete information for v_i in r; whereas the discriminator $\mathcal{D}(r', r, v_i; \theta_{\mathcal{D}})$ tries to discriminate whether the view r' is complementary for v_i in r, using the samples from true distribution. During the process that $\mathcal{G}$ and $\mathcal{D}$ play the minimax game, they receive positive reinforcement from each other. Finally, our generator $\mathcal{G}$ can learns the corresponding underlying reciprocity between heterogeneous relations.

2) Maintain the semantics of different relations in the original HIN while incorporating the reciprocity between complementary relations. As shown in Figue 2, after generating the complementary views, we update the view-semantic representation of the node v_i in view r, by incorporating the neighborhoods information from these fusioned views. After incorporating the complementary relations, we put another constraint on the updated representation, which is computed from the meta-path-based heterogeneous skip-gram model [9]. Benefit from the rich information reflected by meta-paths, MV-ACM could preserve the heterogeneity in both node types and relation types, and maintain the network structures of the original heterogeneous relations.

4 METHODOLOGY

In this section, we introduce our MV-ACM model in details.

4.1 Discriminator

The goal of our discriminator $\mathcal{D}(r', r, v_i; \theta_{\mathcal{D}})$ is to discriminate whether the view r' is complementary for the given node $v_i \in \mathcal{V}$

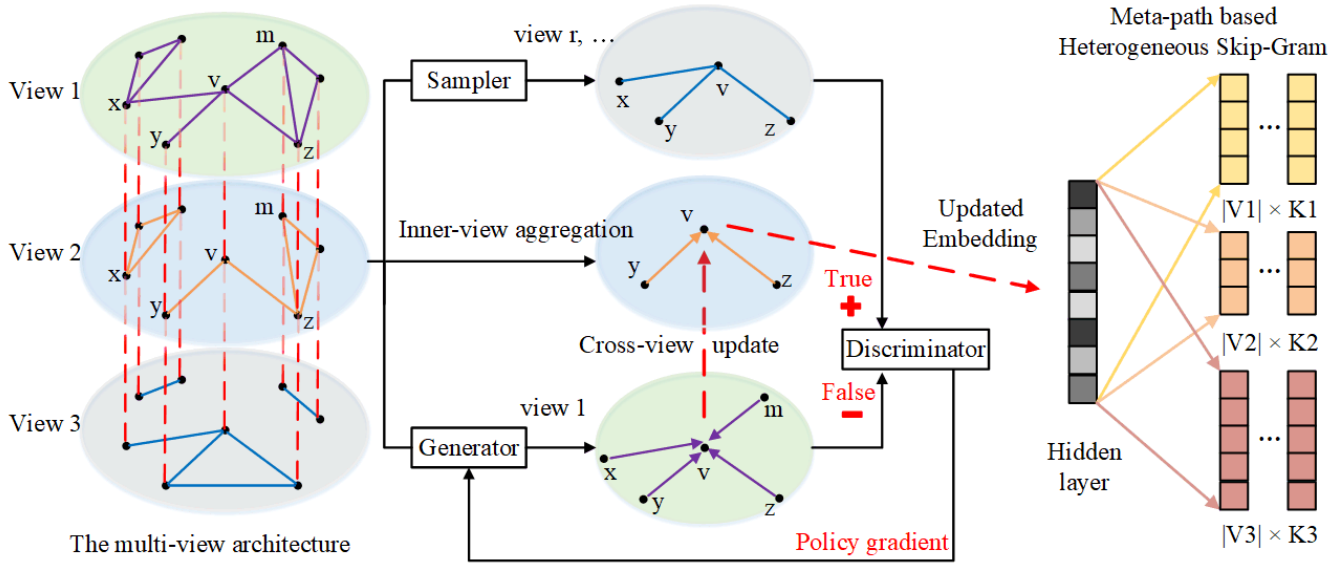

Figure 2: The overall framework of MV-ACM. It shows the learning progress for node v in view 2. Firstly, the view-semantic representations of v in different views are produced by inner-view aggregation respectively. Then the generator generates the complementary views by computing the similarity between these view-semantic representations; On the other hand, the discriminator tries to discriminate whether the view is complementary by using the samples from true underlying distribution. After generating the complementary view, taking 1 as an example, the representation of v in view 2 is updated by aggregating neighborhoods information from view 1 to incorporate the complementary relations. Lastly, the meta-path-based skip-gram is used to preserve the rich information of the HIN.

in the given view $r \in \mathcal{R}$, where the $\theta_{\mathcal{D}}$ represents the trainable parameters of $\mathcal{D}$. So the discriminator should be both globally view-aware and locally node-aware, and output a probability how the view r' is complementary for v_i and r. We present our discriminator as follows:

$$\mathcal{D}(r', r, v_i; \theta_{\mathcal{D}}) = cos\langle \mathbf{u}^{r'}_{\mathcal{D}} + \mathbf{W}^{r'}_{\mathcal{D}}\mathbf{n}^{i}_{\mathcal{D}}, \mathbf{u}^{r}_{\mathcal{D}} + \mathbf{W}^{r}_{\mathcal{D}}\mathbf{n}^{i}_{\mathcal{D}} \rangle \quad (2)$$

where $\mathbf{u}^{r'}_{\mathcal{D}}, \mathbf{u}^{r}_{\mathcal{D}} \in \mathbb{R}^{s\times 1}$ are the view-specific vectors of views r' and r respectively; $\mathbf{n}^{i}_{\mathcal{D}} \in \mathbb{R}^{s\times 1}$ is the node-specific vector of node v_i, with $s, t \ll |\mathcal{V}|$; and $\mathbf{W}^{r'}_{\mathcal{D}}, \mathbf{W}^{r}_{\mathcal{D}} \in \mathbb{R}^{s\times s}$ are the translation matrices between node-specific vector space and each view-specific vector space respectively. In this case, they are all the trainable parameters, *i.e.*, $\theta_{\mathcal{D}} = \{\mathbf{u}^{r}_{\mathcal{D}}, \mathbf{n}^{i}_{\mathcal{D}}, \mathbf{W}^{r}_{\mathcal{D}} | r \in \mathcal{R}, v_i \in \mathcal{V}\}$.

Following the minimax optimization defined in Equation (1), we should maximize the output log-probability when (r', r, v_i) is a true sample from the correct underlying distribution, and minimize the output log-probability when (r', r, v_i) is a false sample from the generator $\mathcal{G}(\cdot|r, v_i; \theta_{\mathcal{G}})$. So the trainable parameters $\theta_{\mathcal{D}}$ of our discriminator could be optimized by:

$$\sum_{v_i\in\mathcal{V}, r\in\mathcal{R}} \Big\{ \mathbb{E}_{X\sim P_{true}(\cdot|r,v_i)} [log\mathcal{D}(X, r, v_i)] + \mathbb{E}_{Z\sim\mathcal{G}(\cdot|r,v_i)} [log(1 - \mathcal{D}(Z, r, v_i))] \Big\} \quad (3)$$

Lastly, we introduce how to get the true sample (r', r, v_i), which indicates the view r' is complementary for v_i in r. The basic idea is that, the more similar the local structures in two views are, the more reasonable to believe there exist complementary information between these two views. So we utilize the structural similarity between two views to produce true samples. Let the $P_{true}(\cdot|v_i, r)$ denotes the true underlying connecting distribution of node v_i in view r, and we can estimate it as:

$$p(v_j|v_i, r) = \frac{e_{i,j,r}}{\sum_{v_k\in\mathcal{V}} e_{i,k,r}} \quad (4)$$

Then the locally topological structural similarity of node v_i between views r' and r can be calculated by the Jensen-Shannon distance between $P_{r',i} = P(\cdot|v_i, r')$ and $P_{r,i} = P(\cdot|v_i, r)$ as:

$$D_{JS}(P_{r',i}||P_{r,i}) = \frac{1}{2}[D_{KL}(P_{r,i}||M) + D_{KL}(P_{r',i}||M)] \quad (5)$$

where $M = \frac{P_{r,i}+P_{r',i}}{2}$, and D_{KL} is the Kullback-Leibler divergence:

$$D_{KL}(P||Q) = \sum_{v} P(v) log \frac{P(v)}{Q(v)} \quad (6)$$

Note that when the locally topological structures of node v_i between views r' and r are identical, $D_{JS}(P_{r',i}||P_{r,i}) = 0$; otherwise $D_{JS}(P_{r',i}||P_{r,i}) = 1$. So we get $S_{struc}(r', r|i) = 1 - D_{JS}(P_{r',i}||P_{r,i})$ as the locally topological structural similarity between views r' and r regarding node v_i. Finally, we can estimate $P_{true}(\cdot|r, v_i)$ and sample true views for discriminator according to the distribution:

$$P_{true}(r'|r, v_i) = \frac{S_{struc}(r', r|i)}{\sum_{r_k\in\mathcal{R}} S_{struc}(r_k, r, i)} \quad (7)$$

4.2 Generator

Firstly, we introduce the trainable parameters $\theta_{\mathcal{G}}$ of our generator $\mathcal{G}$: $\mathbf{n}_i \in \mathbb{R}^{d\times 1}$ is the node-specific vector for node v_i; $\mathbf{u}^0_{i,r} \in \mathbb{R}^{s\times 1}$ is the supplementary vector for node v_i in view r; $\mathbf{M}_r \in \mathbb{R}^{d\times s}, \mathbf{W}^k_r \in \mathbb{R}^{s\times s}$ are the translation matrices regarding view r, where the $d, s \ll |\mathcal{V}|$ and $1 \le k \le K$. K is a hyper parameter as the depth for inner-view aggregation (we will define this soon). Note that we have removed the subscript $\mathcal{G}$ from the notations for convenience.

4.2.1 Inner-view Aggregation. Following the works in graph neural network [14, 38], to capture the structural information, the k-th level ($1 \le k \le K$) supplementary vector $\mathbf{u}^k_{i,r} \in \mathbb{R}^{s\times 1}$ for node v_i in view r can be aggregated from the neighbors':

$$\mathbf{u}^k_{i,r} = aggregate(\{u^{k-1}_{j,r} | v_j \in \mathcal{N}_{i,r}\}) \tag{8}$$

where $\mathcal{N}_{i,r}$ is the set of neighbors (include v_i, *i.e.,* self-loop) associated with node v_i in view r. As suggested by GraphSAGE [14], the *aggregate* can have many forms, such as mean aggregator:

$$\mathbf{u}^k_{i,r} = \sigma(\mathbf{W}^k_r \cdot \mathbf{mean}(\{u^{k-1}_{j,r} | v_j \in \mathcal{N}_{i,r}\})) \tag{9}$$

or max-pooling aggregator:

$$\mathbf{u}^k_{i,r} = \mathbf{max}(\{\sigma(\mathbf{W}^k_r \cdot u^{k-1}_{j,r}) | v_j \in \mathcal{N}_{i,r}\}) \tag{10}$$

where $\sigma(x) = 1/(1 + exp(-x))$ is the sigmoid activation function. Note that we can conduct the aggregating progress in each view simultaneously. In this paper, mean aggregator is used to report our results as their performance is quite similar [14].

4.2.2 Cross-view Updating. We denote the last level supplementary vector $\mathbf{u}^K_{i,r}$ as the view-semantic representation; and calculate

$$S_{semantic}(r', r|i) = cos\langle \mathbf{u}^K_{i,r'}, \mathbf{u}^K_{i,r} \rangle \tag{11}$$

as the similarity between views r' and r regarding node v. Then the generator $\mathcal{G}$ can generate the complementary views for node v_i in view r according to the distribution:

$$\mathcal{G}(r'|r, v_i) = \frac{exp(S_{semantic}(r', r|i))}{\sum_{r_k \in \mathcal{R}} exp(S_{semantic}(r_k, r, i))} \tag{12}$$

Following the minimax optimization defined in Equation (1), we can have the first loss function $\mathcal{L}_1$ for $\mathcal{G}$:

$$\mathcal{L}_1 = \sum_{v_i \in \mathcal{V}, r \in \mathcal{R}} \left\{ \mathbb{E}_{Z \sim \mathcal{G}(\cdot|r, v_i)} [log(1 - \mathcal{D}(Z, r, v_i))] \right\} \tag{13}$$

which could be optimized by policy gradient [41].

Now we can update the view-semantic representation of node v_i in view r by incorporating the information from those complementary views as:

$$\mathbf{u}_{i,r} = \mathbf{u}^K_{i,r} + \sum_{r' \in \mathcal{R}} \mathcal{G}(r'|r, v_i) \mathbf{u}^K_{i,r'} \tag{14}$$

The basic idea is that, the more similar the same node's view-semantic representations in two views are, the more reasonable to incorporate the information between these two views.

Then we can get the overall target representation (see it in Definition 2) $\mathbf{v}_{i,r}$ of node v_i in view r as:

$$\mathbf{v}_{i,r} = \mathbf{n}_i + \mathbf{M}_r \cdot \mathbf{u}_{i,r} \tag{15}$$

Algorithm 1: MV-ACM

Input: $G = (\mathcal{V}, \mathcal{E}, \mathcal{O}, \mathcal{R}, \mathcal{X})$; meta-paths $\mathcal{P}$; dimension d and s; walks per node w ; walk length l negative samples per node L

Output: the representations $\mathbf{v}_{i,r}$ of each node v_i in each view r

```
Initialize all parameters for G and D
Generate walks W_r in each view r based on P_r
Preprocess Eq. (7)
while not converge do
    for Generator_steps do
        Generate views for each node v_i in each view r by Eq. (12).
        Incorporate the information from generated views by Eq. (14).
        Update G by minimizing Eq. (20).
    for Discriminator_steps do
        Sample true views for each node v_i in each view r by Eq. (7)
        Generate false views for each node v_i in each view r by Eq. (12)
        Update D by maximizing Eq. (3)
return the representations in Eq. (15)
```

4.2.3 Heterogeneous Skip-gram. In order to preserve the rich information of the heterogeneous network, we put another constraint on $\mathbf{v}_{i,r}$, using meta-path-based heterogeneous skip-gram model [9] (If there is only one node type, it is the same as skip-gram on random walks [23, 25]).

In this paper, we denote a meta-path begin with edge type r as: $\mathcal{P}_r : o_1 - r_1 - o_2 - r_2 \cdots - r_{l-1} - o_l$, where l is the meta-path length, $o_i \in \mathcal{O}$ is the node type, $r_i \in \mathcal{R}$ is the edge type and $r_1 = r$. Then we can generate nodes sequences $Path\_r$ by:

$$p(v_j | v^t_i; \mathcal{P}) = \begin{cases} \frac{1}{|\mathcal{N}_{i,r} \bigcap o_{t+1}|} & (v_i, v_j) \in \mathcal{E}_{r_t}, v_j \in o_{t+1} \\ 0 & (v_i, v_j) \in \mathcal{E}_{r_t}, v_j \notin o_{t+1} \\ 0 & (v_i, v_j) \notin \mathcal{E}_{r_t} \end{cases} \tag{16}$$

For each generated nodes sequence $Path\_r = (v_1, v_2, \cdots, v_l)$, the positive context of v_i is defined as $C_i = \{v_j | v_j \in P, |i - j| \le c\}$, where c is the window size. Then the skip-gram [23] model try to minimize the negative log-likelihood:

$$-logP(C_i | v_i) = \sum_{v_j \in C_i} -logP(v_j | v_i) \tag{17}$$

And the probability of v_j when given v_i is defined by the heterogeneous softmax function [9]:

$$P(v_j | v_i) = \frac{exp(\langle \mathbf{c}_j, \mathbf{v}_{i,r} \rangle)}{\sum_{v_k \in o_j} exp(\langle \mathbf{c}_k, \mathbf{v}_{i,r} \rangle)} \tag{18}$$

where o_j is the node type corresponding to node v_j, r is the edge type which $Path\_r$ begin with, $\mathbf{c}_j \in \mathbb{R}^{d\times 1}$ is the context vector of node v_j, and $\mathbf{v}_{i,r}$ is calculated by Equation (15). Following metapath2vec [9], we use the heterogeneous negative sampling to approximate Equation (17) and then get the second loss function $\mathcal{L}_2$

Table 2: The detailed statistics of the datasets with sparse views (marked by *)

Task	Dataset	N-types	N-numbers	Edges in Each View					
Classification	AMiner	2	178,385	author-paper 190,645	citation 3,912,854	similarity 1,831,850			
	PPI	1	15,005	neighborhood 37,210	fusion 1,302	occurrence 13,967	expression 638,876	experiment 227,876	database 125,310
Link Prediction	YouTube	1	2,000	friendship 7,797*	friends 300,450	subscriptions 424,510	subscribers 136,938	favorite 244,330	
	Twitter	1	40,000	friendship 1,015,026	re-tweet 7,742*	reply 1,136*	mention 4460*		
	Amazon	1	10,099	co-view 62,973	co-purchase 50,664				
	Alibaba	2	40,324	browse 102,777	add-to-cart 17,400*	preference 12,204*	purchase 17,206*		

for $\mathcal{G}$:

$$\mathcal{L}_2 = \sum_{v_i \in \mathcal{V}} \sum_{v_j \in C_i} \Big\{ - \ log\ \sigma(\langle \mathbf{c}_j, \mathbf{v}_{i,r} \rangle) - \sum_{v_k \in \mathbf{N}_j} log\ \sigma(-\langle \mathbf{c}_k, \mathbf{v}_{i,r} \rangle) \Big\} \tag{19}$$

where $\mathbf{N}_j$ is the negative samples correspond to a positive training node $v_j \in C_i$, which are randomly sampled from node type o_j of node v_j.

Lastly. We integrate the Equation (13) and (19) to get the final objective function for $\mathcal{G}$:

$$\min_{\mathcal{G}} \mathcal{L} = \lambda \mathcal{L}_1 + \mathcal{L}_2 \tag{20}$$

where $\lambda > 0$ is the weight to the losses.

4.3 Inductive Learning

We can extend our model effectively to deal with inductive learning [14], where there may exist unseen nodes after the training stage. And the extended model can handle the attributes as well. Following previous works [5, 14], in the extended model, we learn the trainable functions over the feature $\mathcal{X}$ of nodes, instead of training the parameters vectors of nodes directly. To be more specific, in generator and discriminator, $\mathbf{u}^0_{i,r}$, $\mathbf{n}_i$ and $\mathbf{n}^i_{\mathcal{D}}$ are not the trainable parameters, but can be learned by: $\mathbf{u}^0_{i,r} = f_r(x_i)$, $\mathbf{n}_i = g(x_i)$ and $\mathbf{n}^i_{\mathcal{D}} = g_{\mathcal{D}}(x_i)$, where x_i is the feature of v_i, $f_r(), g(), g_{\mathcal{D}}()$ are trainable functions respectively. And the rest of our model remains unchanged. The trainable functions can have different forms such as multi-layer perceptron [5].

4.4 The Optimization and Time Complexity

We present the adversarial optimization process in Algorithm 1. The time complexity of generating walks and preprocessing is $O(wl|\mathcal{V}||\mathcal{R}| + |\mathcal{E}|)$; the time complexity of training is $O(|\mathcal{V}||\mathcal{R}|Ld)$ per epoch, where w is walks per node, l is walk length, L is negative samples per node, d is the dimension of representations, $|\mathcal{V}|$ is the number of nodes, $|\mathcal{E}|$ is the number of edges, $|\mathcal{R}|$ is the number of views (*i.e.,* edge types). Not considering the constant terms, the time complexity is the same as other scalable models, *i.e.,* linear to the network scale $O(|\mathcal{V}||\mathcal{R}| + |\mathcal{E}|)$.

5 EXPERIMENTS

Link prediction and node classification is widely used to evaluate the quality of the nodes representations, as they plays a fundamental role in many real-world tasks, *e.g.,* the recommendation [19, 29] and identity mining [13, 28]. So in this section, we empirically evaluate MV-ACM on six real-world networks from various domains with these tasks. We describe the datasets and the competitor baselines first. Then we present the quantitative results, and analyze our model with more details.

5.1 Experimental Setup

5.1.1 Datasets. We select a series of benchmark datasets [5, 16, 27] from various domains, *i.e.,* academic networks, biological networks, social networks and e-commerce networks. The detailed statistics of these networks are shown in Table 2. We will also specify the sparse views according to the data and report the link prediction result on these views additionally to evaluate our models with respect to the sparsity problems.

1. AMiner. AMiner is an academic research dataset[2] [34] which contains authors, papers and conferences information. We extract a subset to build a three-view network, with two node types (*i.e., author* and *paper*) and three edge types (*i.e., author-paper, paper-paper*: *citation* and *paper-paper*: *key terms similarity*). The similarity is calculated using the TF-IDF cosine value of key terms, and the five-nearest neighbors are kept in this view. We choose eight different research areas and label the authors according to the representative conferences they submitted to. Note that each author may belong to more than one research areas, so this is for a multi-label classification.

2. PPI. The STRING PPI dataset[3] [31] contains protein-protein interaction information. We choose Homo Sapiens organism and build a six-view network, with one node type (*i.e., protein*) and six edge types (*i.e., gene neighborhood, gene fusion, gene co-occurrence, co-expression, experimental relation* and *database relation*). And each protein has features[4] [20] that are composed of positional gene sets, motif gene sets and immunologic gene sets. We label the proteins

[2] https://www.aminer.cn
[3] https://string-db.org
[4] http://software.broadinstitute.org/gsea/msigdb

Table 3: Quantitative results on link prediction

Model	YouTube						Twitter						Alibaba					
	All Views			Sparse Views			All Views			Sparse Views			All Views			Sparse Views		
	ROC	PR	F1	ROC	PR	F1	ROC	PR	F1	ROC	PR	F1	ROC	PR	F1	ROC	PR	F1
LINE	67.42	66.25	65.45	68.14	67.22	66.03	58.78	58.11	55.07	56.14	56.07	54.86	54.06	54.71	53.34	53.12	53.87	52.74
DeepWalk	71.58	70.34	66.12	72.84	71.69	67.75	64.17	65.53	60.81	60.21	61.35	60.14	56.86	57.21	54.30	55.24	56.07	53.49
Node2vec	72.41	71.24	66.85	73.81	71.88	67.68	64.43	65.61	61.77	60.67	61.46	60.07	59.41	59.53	57.04	56.80	56.34	54.97
GraphGAN	72.76	71.46	67.07	73.61	72.42	68.17	64.77	65.74	61.83	60.52	61.49	60.24	60.52	60.81	57.22	56.81	56.67	55.09
ANRL	75.45	73.94	70.31	76.42	74.44	71.57	65.34	65.01	62.07	60.84	61.41	61.29	57.11	57.63	54.38	55.61	56.14	53.84
Metapath2vec	71.57	70.51	66.29	72.83	71.81	67.85	64.05	65.51	61.41	60.51	61.37	60.43	58.34	58.87	55.41	56.43	56.14	54.88
HeGAN	73.21	72.64	67.54	74.64	73.81	68.07	64.98	65.91	62.17	60.84	61.81	61.57	60.83	61.08	57.81	57.24	56.69	55.41
MVE	70.47	70.19	65.74	70.89	70.83	66.71	67.41	67.34	63.72	63.57	63.42	63.04	60.17	60.52	56.97	56.70	56.43	55.07
MNE	82.27	81.21	75.13	83.51	81.77	75.53	86.14	86.37	81.72	81.97	81.14	80.47	62.74	62.89	59.13	58.64	58.60	56.71
GATNE-T	84.01	81.20	75.84	84.22	83.14	76.45	86.85	86.41	81.98	82.23	81.97	81.15	64.15	64.48	60.18	59.74	59.88	57.46
GATNE-I	83.94	81.17	75.81	83.47	82.76	76.04	86.67	86.38	81.64	82.74	82.46	80.92	63.97	64.01	59.94	59.37	59.81	57.31
MV-ACM	**87.30**	**85.54**	**79.46**	**90.27**	**90.84**	**82.65**	**88.96**	**89.05**	**84.01**	**86.82**	**86.34**	**84.89**	**65.89**	**66.75**	**61.77**	**64.46**	**65.51**	**60.78**
Gain [%]	**3.92** ↑	**5.34** ↑	**4.77** ↑	**7.18** ↑	**9.26** ↑	**8.11** ↑	**2.43** ↑	**3.06** ↑	**2.48** ↑	**4.93** ↑	**4.71** ↑	**4.61** ↑	**2.71** ↑	**3.52** ↑	**2.64** ↑	**7.90** ↑	**9.40** ↑	**5.78** ↑

according to the hallmark gene sets[4] which can represent biological states. This is for a multi-label classification with 50 different labels.

3. YouTube. This dataset[5] [36] contains the different interactions information between users from video website YouTube. A subset is used to build a five-view network [5], with one node type (*i.e., user*) and five edge types (*i.e., the friendship, shared friends, shared subscriptions, shared subscribers* and *shared favorite videos*, which denotes the two users are friends, or they have common friends or common subscriptions etc). The friendship between users is treated as the sparse view in our experiments.

4. Twitter. This dataset[6] [8] has been built to monitor the spreading processes about the announcement of the Higgs boson on Twitter between 1st and 7th, July 2012. We extract a subset to build a four-view network, with one node type (*i.e., user*) and four edge types (*i.e., the follower-friendship, re-tweet, reply* and *mention*). The re-tweet, reply and mention between users are treated as the sparse views in our experiments.

5. Amazon. This dataset[7] [15] contains the information and reviews of products from Amazon. The products from electronics category are used to build a two-view network [5], with one node type (*i.e., product*) and two edge types (*i.e., the co-viewed relation* and *co-purchased relation*). As there are no sparse views in this dataset, we conduct link prediction for it by hiding different percentages of edges (detail is in Section 5.2).

6. Alibaba. This dataset[8] contains users' shopping logs on the e-commerce platform of Alibaba. We extract a subset to build a four-view network, with two node types (*i.e., user* and *item*) and four edge types (*i.e.,* user-item: *browse, add-to-cart, add-to-preference* and *purchase*). The add-to-cart, add-to-preference and purchase between user and item are treated as the sparse views in our experiments.

5.1.2 Baselines. We compare our model with the state-of-the-art methods, which fall into three main groups as shown in Table 4. More information can be found in Section 6.

1. Homogeneous Methods. This group include: LINE (1st+2nd forms) [33], DeepWalk [25], Node2vec [13], GraphGAN [41], ANRL [49] and DGI [39]. As these methods can only handle the homogeneous network, we train them on each view separately without discriminating node types. Besides, there is no trivial way for DGI [39] to predict link probability, so we only report its results on node classification tasks.

2. Heterogeneous Methods. This group include: Metapath2vec [9] and HeGAN [17].

3. Muti-view Methods. This group include: MVE [27], MNE [46], GATNE-T and GATNE-I [5], as well as our MV-ACM.

Table 4: Comparisons between representative models

Model	Network Type			inductive
	Heterogeneous	Multi-view	Attribute	
DeepWalk [25] LINE [33] GraphGAN [41]	×	×	×	×
DANE [10] ANRL [49]	×	×	√	×
GraphSAGE [14] DGI [39]	×	×	√	√
Metapath2vec [9] HERec [29] HeGAN [17]	√	×	×	×
HNE [6]	√	×	√	×
MVE [27] MNE [46]	*	√	×	×
MINES [22]	√	√	×	×
GATNE [5] **MV-ACM**	√	√	√	√

∗ They don't consider different node types.

5.1.3 Parameter Settings. The embedding dimensions d are 128 for all models. The parameters dimensions s, t in our generator and discriminator are both 20, and the inner-view aggregation layer K is 1. The number of walks per node is 40, the length of walks is 10, the window size is 5 and the number of negative samples is 5 for all models which need them. For the meta-path based models: in AMiner, the meta-path is author-paper-author and paper-author-paper; in Alibaba, the meta-path is user-item-user and item-user-item. For all models, the validation set is used for hyper-parameters tuning (*i.e.,* the learning rate from $\{0.0001, 0.001\}$, the batch-size from $\{128, 256, 512\}$, the search bias p, q from $\{0.5, 1, 2\}$, the weight to the

[5] http://socialcomputing.asu.edu/datasets/YouTube
[6] https://snap.stanford.edu/data/higgs-twitter.html
[7] http://jmcauley.ucsd.edu/data/amazon
[8] https://tianchi.aliyun.com/dataset/dataDetail?dataId=42

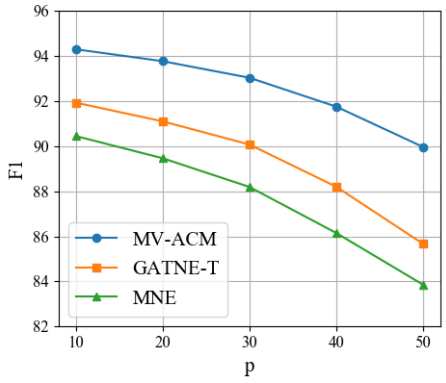

Figure 3: Link prediction results (F1-score) on Amazon dataset with $(5+p)$% edges missing, to the most comparable methods GATNE-T and MNE. We can find that the gains over feat are more significant (from 2.6% to 5.0%) as the sparse problems go more serious (from 15% to 55%).

losses λ from $\{0.05, 0.1, 0.5, 1\}$ and the early stopping epoch; others are set as in their original papers), and the test set is used to evaluate the performance. We only turn our MV-ACM model to inductive learning for PPI, which is the only dataset with attributes, and the trainable functions ($f_r, g, g_{\mathcal{D}}$, see in Section 4.3) are simple linear transformations. For ANRL, DGI and GATNE-I which need attributes, we pre-train DeepWalk and take the embeddings ($d = 128$) as nodes attributes for them, on all datasets except PPI. We implement our model MV-ACM[1] by Tensorflow-1.12[9]. All the experiments are conducted on a Linux server with two Intel Xeon E5 CPU and one GTX 2080Ti GPU.

5.2 Link Prediction

We randomly hide 5%/p% (p is the varying percentages of different test sets) of edges from the original network as the positive links and randomly sample disconnected node pairs with the equivalent number as the negative links to get the validation/test sets, for each edge type (*i.e.,* each edge type has $(5+p)$% missing edges). For YouTube, Twitter and Alibaba datasets, the p is set to 10; for Amazon dataset the p is set from {10, 20, 30, 40, 50}. We train each model on the residual network and obtain the node representations. Then we calculate the cosine similarity of the learned representations to predict the link probability. We take the commonly used ROC-AUC (the area under the ROC curve), PR-AUC (the area under the PR curve) and F1-score as the evaluation criteria [5, 35].

We present the quantitative results in Table 3 (the gains is compared with runners-up) and Figure 3. And we can have the following observations:

- MV-ACM consistently outperforms state-of-the-art baselines from all groups on the four datasets.
- Heterogeneous methods only outperform homogeneous methods on Alibaba dataset, where there exists two node types and they can preserve these information. But for both YouTube and Twitter, where there exist only one node type but with different relation types, these heterogeneous methods degenerate into homogeneous methods, as they can't handle different relations between the same pair of two nodes the same with homogeneous methods.
- Muti-view methods outperform the other groups (except for one case, as MVE only preserve first-order information), which indicates the importance and usefulness of learning such different relations between the same pair of two nodes. So the multi-view architecture are basically more superior for modeling real-word complex HIN.
- As shown in Figure 3, the more missing data are, the more significant improvements MV-ACM makes. Our model can deal with the sparse problems better than the others.
- Meanwhile, the most significant improvement can be found with YouTube dataset, but it is not surprising, as the shared friends and the shared favorite *et al.* relations are complementary to the friendship for most users. Besides, MV-ACM outperforms other muti-view based methods not only in sparse views, but also in all views. These observations together indicate that MV-ACM has the ability to incorporate these complementary structural information, as well as maintain the original semantics of the HIN.

Table 5: Quantitative results on node classification

Model	AMiner		PPI	
	Macro-F1	Micro-F1	Macro-F1	Micro-F1
LINE	60.12	60.51	43.81	46.57
DeepWalk	62.17	52.33	44.17	46.55
Node2vec	65.44	65.90	45.31	47.92
GraphGAN	63.71	63.88	44.52	47.10
ANRL	63.21	63.47	49.56	42.93
DGI	64.81	65.34	54.38	57.14
Metapath2vec	65.10	65.49	44.01	46.79
HeGAN	65.87	65.96	44.82	46.91
MVE	63.50	63.57	46.82	49.34
MNE	67.22	67.47	50.87	54.03
GATNE-T	70.62	70.90	52.48	55.22
GATNE-I	70.17	70.24	55.61	58.47
MV-ARCM	**72.89****	**73.51****	**58.72****	**61.90****

**: significantly outperforms the runner-up based on paired t-test at the significance level of 0.01.

5.3 Node Classification

We train each model and take the representations of nodes as the features for the multi-label classification task. For these models which can't handle the attributes in PPI dataset, we concatenate the representations and raw attributes as final features following [5, 38]. We randomly sample 10%/80% nodes to get the validation/test sets, and then train the one-vs-rest logistic regression classifier with L2 regularization [13, 27] on the remaining 10% nodes. We repeat 5 times and evaluate the average performance for each model. We take the commonly used Macro-F1 and Micro-F1 as the evaluation criteria [13, 35].

We present the quantitative results in Table 5. And we can have some similar observations as in link prediction task. Besides, ANRL

[9] https://www.tensorflow.org/

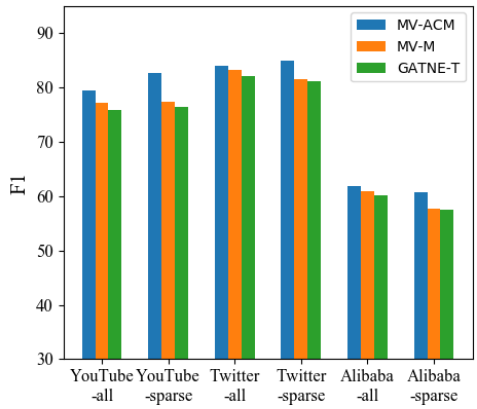

Figure 4: The importance of our adversarial learning.

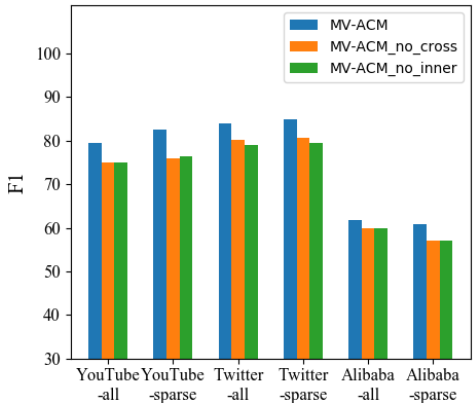

Figure 5: The importance of inner-view and cross-view aggregation.

and DGI outperform other homogeneous methods on PPI more significantly, as they can model these informative attributes associated to nodes in PPI datasets. It can be also seen that MV-ACM consistently outperforms all baselines on the two datasets with statistical significance. These results on node classification tasks indicate that MV-ACM can preserve more robust and semantic information for the real-word networks with both entities and relations heterogeneity.

5.4 Model Analysis

In this subsection, we will analyze our model from three aspects: the effectiveness of adversarial learning, the effectiveness of cross-view and inner-view aggregation, model convergence and scalability.

5.4.1 The Effectiveness of Adversarial Learning. Firstly, we focus on our adversarial learning, and evaluate its ability to distinguish the complementary information. To be more specific, we denote a variant of our model as MV-M, by optimizing generator only with loss function $\mathcal{L}_2$ in Equation (19). So the only difference is that MV-M aggregates information from all views without adversarial learning. Their performance on link prediction tasks is shown in Figure 4, from which we can conclude that: (1) there do exist complementary

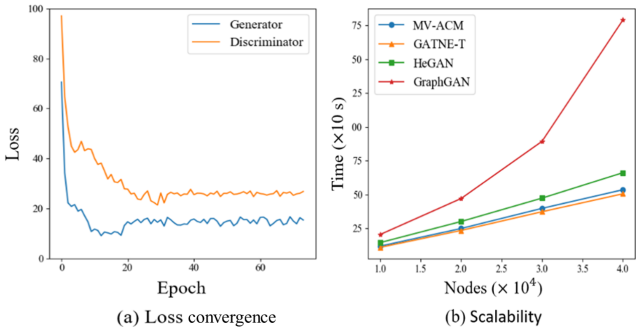

Figure 6: Convergence and scalability evaluations on AMiner dataset. (a) The learning curves. (b) The training time increases in linear scale *w.r.t* the number of nodes.

information between different relations as aggregating information from other views gets better performance compared with GATNE-T, (2) however, different relations are not always complementary, as aggregating information from all views gets worse performance compared with MV-ACM. So our adversarial learning is able to distinguish the complementary information out from others, which is crucial for completion.

5.4.2 The Effectiveness of Cross-view and Inner-view Aggregation. Then we focus on the incorporating process. We denote two variant of our model as MV-ACM_no_inner and MV-ACM_no_cross, by removing the inner-view aggregating (Eq. (8)) and and cross-view updating (Eq. (14)) respectively. Note that they are different from MV-M as they utilize the adversarial learning to distinguish the complementary information. The results on link prediction tasks are shown in Figure 5, and we can see that: MV-ACM_no_cross performs almost the same as MV-ACM_no_inner, and they are both worse than MV-ACM. Therefore, the complementary information from other relations is as important as the structural information from neighborhoods within one relation. They can both contribute to learn more robust representations for HIN.

5.4.3 Model Convergence and Scalability. We report the learning curves of MV-ACM on AMiner in Figure 6 (a). We can see that the generator and the discriminator decrease the losses of both by the minimax game, and converge well with about 20 epochs. We also sample 4 subsets of AMiner with the number of nodes from 10 thousand to 40 thousand, and report the training time compared with GATNE-T, HeGAN and GraphGAN in Figure 6 (b). We can see that the training time of MV-ACM increases in linear scale *w.r.t* the size of network. And MV-ACM is superior to most existing scalable methods, except for GATNE-T as it has no adversarial process.

6 RELATED WORK

Our work is mostly related to network embedding and adversarial learning. The related work are summarized as follows.

6.1 Network Embedding

We review this related work in network embedding, heterogeneous network embedding and multi-view network embedding. We summarize some representative models in Table 4 corresponding to their capacity,

6.1.1 Network Embedding. Network embedding, which aims to represent nodes into dense vectors in low-dimensional spaces, has shown significant effectiveness in various data mining tasks [3, 28, 29, 44]. It can be divided into two categories, unsupervised network embedding (NE) and graph neural network (GNN) [5].

NE is also related to spectral clustering and matrix factorization [1, 2, 26], which has been carefully studied since a long time ago [37]. Facing the daunting challenges of enormous scales and inspired by the deep learning techniques [11, 23], A number of pioneering approaches in NE have been proposed based on solving structure-preserving optimization problem. DeepWalk [25], LINE [33] and Node2vec [13] use different sampling strategies to preserve the topological structures: depth-first neighborhoods, breadth-first neighborhoods and a balance between them respectively; GraRep [4], NEU [43], ProNE [47] and AROPE [48] propose to preserve the high-order adjacency proximity; Struc2vec [28] propose to preserve the structural identity based on degrees; SDNE [40] uses deep autoencoder to replace the linear transformation; NetSMF [26] shows that these approaches are approximately factorizing some respective matrices, so they are equivalent in essence; DANE [10] and ANRL [49] uses autoencoder to preserve attributes proximity. Most recently, Deep Graph Infomax (DGI) [39] introduces maximizing mutual information into network analysis.

Meanwhile, approaches in GNN are proposed for network learning in a task-specific supervised setting primarily. GCN [18] uses convolutional operations while GAT [38] uses attention mechanisms to aggregate neighbors' attributes into the nodes representations. GraphSAGE [14] learns functional representations in an inductive manner, by performing some specific aggregators over neighbors.

Though empirically efficient and effective, all these approaches above can handle only the homogeneous network, *i.e.,* only one single node type and one single edge type.

6.1.2 Heterogeneous Network Embedding. Heterogeneous networks are proposed to model nodes and edges of various types [9, 29]. PTE [32] preserves proximity between different node types. Metapath2vec [9] uses meta-path based random walk and a heterogeneous skip-gram to learn different semantics. HERec [29] transforms semantic meanings by a set of functions and proposes extended matrix factorization. HNE [6] uses deep architectures to preserve attributes information and learns different nodes to unified vector space, however it can't scale well to large networks. Though empirically effective, these studies above try to learn and preserve the distinctive semantics of edges and nodes [17], they rarely speak of the multiple relations between the same pair of two nodes [5], *i.e.,* the multi-view network. Thus these approaches ignore the reciprocity of different relation types.

6.1.3 Multi-view Network Embedding. In reality, we can usually find many types of relations or interactions between the same pair of two nodes. Recently, some preliminary models have been proposed to handle the multi-view property. PMNE [21] utilize Node2vec to learn node representations into unified space. While MINES [22], MVE [27], MNE [46] and mvn2vec [30] suggests to preserve the semantics of different views respectively. To be more specific, MVE [27] preserve the first-order proximity and then use attention mechanism in a task-specific supervised ways; MNE [46] uses a common representation to bridge all views indirectly; while mvn2vec [30] preserves all views' information with collaboration directly. Inspired by GraphSAGE [14], GATNE [5] proposes a general model to handle heterogeneous multi-view network, and aggregate information in each view-specific neighborhood.

6.2 Adversarial Learning

Generative adversarial networks (GAN) [12] have received extensive attention, since it demonstrates superior success in various applications [11, 12]. By designing the minimax theoretical game, generator and discriminator compete to improve each other, and learn the underlying data distribution in unsupervised setting. Inspired by GAN, some works [24, 45] take a fixed prior distribution as regularization for network embedding, like ANE [7] *et al.* . GraphGAN [41] proposes a graph softmax and unifies generative and discriminative thinking for network embedding. HeGAN [17] proposes relation-aware generator and discriminator to capture the rich semantics for HIN, and sample the fake nodes from a continuous distribution efficiently.

However, after reviewing the related works in network embedding and adversarial learning, we find that they mainly focus on preserving the distinctive semantics of different entities and/or relations, and few of them stress the sparsity problems and explore the reciprocity. They don't distinguish whether there exists complementary information across vies, which can enhance the network representation learning. Meanwhile, some other methods, proposed to handle the sparsity problems, are task-specific or domain-specific, which need hand-engineering based on expert knowledge, *e.g.,* SHINE [42] and CMAP [19].

7 CONCLUSION

In this paper, we emphasize the sparsity problems for real-world complex HIN embedding, and show the importance and usefulness of the complementary information between different views for completion. We propose a novel and principled model MV-ACM based on adversarial learning, to distinguish and incorporate such complementary information from different semantic space of HIN. Then MV-ACM can update the node's semantic representation by aggregating neighborhood information from different views. Systematical experiments on six real-world networks shows the prior performance of MV-ACM on two downstream tasks compared with the state-of-the-art baselines. In the future, we will take more attributes of the relations into consideration.

ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Program of China under Grant 2018YFC0831500, the National Natural Science Foundation of China (NSFC) under Grant 61972047, and the NSFC-General Technology Basic Research Joint Funds under Grant U1936220.

REFERENCES

[1] Amr Ahmed, Nino Shervashidze, Shravan M. Narayanamurthy, Vanja Josifovski, and Alexander J. Smola. 2013. Distributed large-scale natural graph factorization. In *WWW, 2013*. 37–48.

[2] Mikhail Belkin and Partha Niyogi. 2001. Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. In *NIPS, 2001*. 585–591.

[3] Carter T Butts. 2009. Revisiting the foundations of network analysis. *Science* 325, 5939 (2009), 414–416.

[4] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. GraRep: Learning Graph Representations with Global Structural Information. In *ACM CIKM, 2015*. 891–900.

[5] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation Learning for Attributed Multiplex Heterogeneous Network. In *ACM SIGKDD, 2019*. 1358–1368. https://doi.org/10.1145/3292500.3330964

[6] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C. Aggarwal, and Thomas S. Huang. 2015. Heterogeneous Network Embedding via Deep Architectures. In *ACM SIGKDD, 2015*. 119–128.

[7] Quanyu Dai, Qiang Li, Jian Tang, and Dan Wang. 2018. Adversarial Network Embedding. In *AAAI, 2018*. 2167–2174.

[8] M. De Domenico, A. Lima, P. Mougel, and M. Musolesi. 2013. The Anatomy of a Scientific Rumor. *Scientific Reports* 3, 10 (2013), 65–65.

[9] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *ACM SIGKDD, 2017*. 135–144.

[10] Hongchang Gao and Heng Huang. 2018. Deep Attributed Network Embedding. In *IJCAI, 2018*. 3364–3370.

[11] Ian J. Goodfellow, Yoshua Bengio, and Aaron C. Courville. 2016. *Deep Learning*. MIT Press.

[12] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *NIPS, 2014*. 2672–2680.

[13] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *ACM SIGKDD, 2016*. 855–864.

[14] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NIPS, 2017*. 1024–1034.

[15] Ruining He and Julian J. McAuley. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *Proceedings of the 25th International Conference on World Wide Web, WWW, 2016*. 507–517. https://doi.org/10.1145/2872427.2883037

[16] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*. 173–182.

[17] Binbin Hu, Yuan Fang, and Chuan Shi. 2019. Adversarial Learning on Heterogeneous Information Networks. In *ACM SIGKDD, 2019*. 120–129.

[18] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR, 2017*.

[19] Adit Krishnan, Ashish Sharma, and Hari Sundaram. 2018. Insights from the Long-Tail: Learning Latent Representations of Online User Behavior in the Presence of Skew and Sparsity. In *ACM CIKM, 2018*. 297–306.

[20] Arthur Liberzon, Aravind Subramanian, Reid Pinchback, Helga Thorvaldsdóttir, Pablo Tamayo, and Jill P. Mesirov. 2011. Molecular signatures database (MSigDB) 3.0. *Bioinformatics* 27, 12 (2011), 1739–1740.

[21] Weiyi Liu, Pin-Yu Chen, Sailung Yeung, Toyotaro Suzumura, and Lingli Chen. 2017. Principled Multilayer Network Embedding. In *IEEE ICDM Workshops, 2017*. 134–141.

[22] Yao Ma, Zhaochun Ren, Ziheng Jiang, Jiliang Tang, and Dawei Yin. 2018. Multi-Dimensional Network Embedding with Hierarchical Structure. In *ACM WSDM, 2018*. 387–395.

[23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS, 2013*. 3111–3119.

[24] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. 2018. Adversarially Regularized Graph Autoencoder for Graph Embedding. In *IJCAI, 2018*. 2609–2615.

[25] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *ACM SIGKDD, 2014*. 701–710.

[26] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Chi Wang, Kuansan Wang, and Jie Tang. 2019. NetSMF: Large-Scale Network Embedding as Sparse Matrix Factorization. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*. 1509–1520.

[27] Meng Qu, Jian Tang, Jingbo Shang, Xiang Ren, Ming Zhang, and Jiawei Han. 2017. An Attention-based Collaboration Framework for Multi-View Network Representation Learning. In *ACM CIKM, 2017*. 1767–1776.

[28] Leonardo Filipe Rodrigues Ribeiro, Pedro H. P. Saverese, and Daniel R. Figueiredo. 2017. struc2vec: Learning Node Representations from Structural Identity. In *ACM SIGKDD, 2017*. 385–394.

[29] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and Philip S. Yu. 2019. Heterogeneous Information Network Embedding for Recommendation. *IEEE TKDE* 31, 2 (2019), 357–370.

[30] Yu Shi, Fangqiu Han, Xinran He, Carl Yang, Jie Luo, and Jiawei Han. 2018. mvn2vec: Preservation and Collaboration in Multi-View Network Embedding. *CoRR* (2018). arXiv:1801.06597

[31] Damian Szklarczyk, Annika L. Gable, David Lyon, Alexander Junge, Stefan Wyder, Jaime Huerta-Cepas, Milan Simonovic, Nadezhda T. Doncheva, John H. Morris, Peer Bork, Lars Juhl Jensen, and Christian von Mering. 2019. STRING v11: protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic Acids Research* 47, Database-Issue (2019), D607–D613.

[32] Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. PTE: Predictive Text Embedding through Large-scale Heterogeneous Text Networks. In *ACM SIGKDD, 2015*. 1165–1174.

[33] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*. 1067–1077. https://doi.org/10.1145/2736277.2741093

[34] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. ArnetMiner: extraction and mining of academic social networks. In *ACM SIGKDD, 2008*. 990–998.

[35] Lei Tang, Suju Rajan, and Vijay K. Narayanan. 2009. Large scale multi-label classification via metalabeler. In *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*. 211–220. https://doi.org/10.1145/1526709.1526738

[36] Lei Tang, Xufei Wang, and Huan Liu. 2009. Uncoverning Groups via Heterogeneous Interaction Analysis. In *ICDM, 2009*. 503–512.

[37] J B Tenenbaum, De Silva, V, and J C Langford. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science* 290, 5500 (2000), 2319–2323.

[38] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR, 2018*.

[39] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2019. Deep Graph Infomax. In *ICLR, 2019*.

[40] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural Deep Network Embedding. In *ACM SIGKDD, 2016*. 1225–1234.

[41] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. GraphGAN: Graph Representation Learning With Generative Adversarial Nets. In *AAAI, 2018*. 2508–2515.

[42] Hongwei Wang, Fuzheng Zhang, Min Hou, Xing Xie, Minyi Guo, and Qi Liu. 2018. SHINE: Signed Heterogeneous Information Network Embedding for Sentiment Link Prediction. In *ACM WSDM, 2018*. 592–600.

[43] Cheng Yang, Maosong Sun, Zhiyuan Liu, and Cunchao Tu. 2017. Fast Network Embedding Enhancement via High Order Proximity Approximation. In *IJCAI, 2017*. 3894–3900.

[44] Shuang-Hong Yang, Bo Long, Alexander J. Smola, Narayanan Sadagopan, Zhaohui Zheng, and Hongyuan Zha. 2011. Like like alike: joint friendship and interest propagation in social networks. In *Proceedings of the 20th International Conference on World Wide Web, WWW, 2011*. 537–546. https://doi.org/10.1145/1963405.1963481

[45] Wenchao Yu, Cheng Zheng, Wei Cheng, Charu C. Aggarwal, Dongjin Song, Bo Zong, Haifeng Chen, and Wei Wang. 2018. Learning Deep Network Representations with Adversarially Regularized Autoencoders. In *ACM SIGKDD, 2018*. 2663–2671.

[46] Hongming Zhang, Liwei Qiu, Lingling Yi, and Yangqiu Song. 2018. Scalable Multiplex Network Embedding. In *IJCAI, 2018*. 3082–3088.

[47] Jie Zhang, Yuxiao Dong, Yan Wang, Jie Tang, and Ming Ding. 2019. ProNE: Fast and Scalable Network Representation Learning. In *IJCAI, 2019*. 4278–4284.

[48] Ziwei Zhang, Peng Cui, Xiao Wang, Jian Pei, Xuanrong Yao, and Wenwu Zhu. 2018. Arbitrary-Order Proximity Preserved Network Embedding. In *ACM SIGKDD, 2018*. 2778–2786.

[49] Zhen Zhang, Hongxia Yang, Jiajun Bu, Sheng Zhou, Pinggang Yu, Jianwei Zhang, Martin Ester, and Can Wang. 2018. ANRL: Attributed Network Representation Learning via Deep Neural Networks. In *IJCAI, 2018*. 3155–3161.