Representation Learning for Attributed Multiplex Heterogeneous Network

Yukuo Cen[†], Xu Zou[†], Jianwei Zhang[‡], Hongxia Yang[‡*], Jingren Zhou[‡], Jie Tang[†*]

[†] Department of Computer Science and Technology, Tsinghua University
[‡] DAMO Academy, Alibaba Group
{cyk18,zoux18}@mails.tsinghua.edu.cn
{zhangjianwei.zjw,yang.yhx,jingren.zhou}@alibaba-inc.com
jietang@tsinghua.edu.cn

ABSTRACT

Network embedding (or graph embedding) has been widely used in many real-world applications. However, existing methods mainly focus on networks with single-typed nodes/edges and cannot scale well to handle large networks. Many real-world networks consist of billions of nodes and edges of multiple types, and each node is associated with different attributes. In this paper, we formalize the problem of embedding learning for the *Attributed Multiplex Heterogeneous Network* and propose a unified framework to address this problem. The framework supports both *transductive* and *inductive* learning. We also give the theoretical analysis of the proposed framework, showing its connection with previous works and proving its better expressiveness. We conduct systematical evaluations for the proposed framework on four different genres of challenging datasets: Amazon, YouTube, Twitter, and Alibaba[1]. Experimental results demonstrate that with the learned embeddings from the proposed framework, we can achieve statistically significant improvements (e.g., 5.99-28.23% lift by F1 scores; $p \ll 0.01$, $t-$test) over previous state-of-the-art methods for link prediction. The framework has also been successfully deployed on the recommendation system of a worldwide leading e-commerce company, Alibaba Group. Results of the offline A/B tests on product recommendation further confirm the effectiveness and efficiency of the framework in practice.

CCS CONCEPTS

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

KEYWORDS

Network embedding; Multiplex network; Heterogeneous network

[*]Hongxia Yang and Jie Tang are the corresponding authors.
[1]Code is available at https://github.com/cenyk1230/GATNE.

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330964

ACM Reference Format:
Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou and Jie Tang. 2019. Representation Learning for Attributed Multiplex Heterogeneous Network. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3292500.3330964

1 INTRODUCTION

Network embedding [4], or network representation learning, is a promising method to project nodes in a network to a low-dimensional continuous space while preserving network structure and inherent properties. It has attracted tremendous attention recently due to significant progress in downstream network learning tasks such as node classification [1], link prediction [39], and community detection [8]. DeepWalk [27], LINE [35], and node2vec [10] are pioneering works that introduce deep learning techniques into network analysis to learn node embeddings. NetMF [29] gives a theoretical analysis of equivalence for the different network embedding algorithms, and later NetSMF [28] gives a scalable solution via sparsification. Nevertheless, they were designed to handle only the homogeneous network with single-typed nodes and edges. More recently, PTE [34], metapath2vec [7], and HERec [31] are proposed for heterogeneous networks. However, real-world network-structured applications, such as e-commerce, are much more complicated, comprising not only multi-typed nodes and/or edges but also a rich set of attributes. Due to its significant importance and challenging requirements, there have been tremendous attempts in the literature to investigate embedding learning for complex networks. Depending on the network topology (homogeneous or heterogeneous) and attributed property (with or without attributes), we categorize six different types of networks and summarize their relative comprehensive developments, respectively, in Table 1. These six categories include *HO*mogeneous *N*etwork (or HON), *A*ttributed *HO*mogeneous *N*etwork (or AHON), *HE*terogeneous *N*etwork (or HEN), *A*ttributed *HE*terogeneous *N*etwork (or AHEN), *M*ultiplex *HE*terogeneous *N*etwork (or MHEN), and *A*ttributed *M*ultiplex *HE*terogeneous *N*etwork (or AMHEN). As can be seen, among them the AMHEN has been least studied.

In this paper, we focus on embedding learning for AMHENs, where different types of nodes might be linked with multiple different types of edges, and each node is associated with a set of different attributes. This is common in many online applications. For example, in the four datasets that we are working with, there

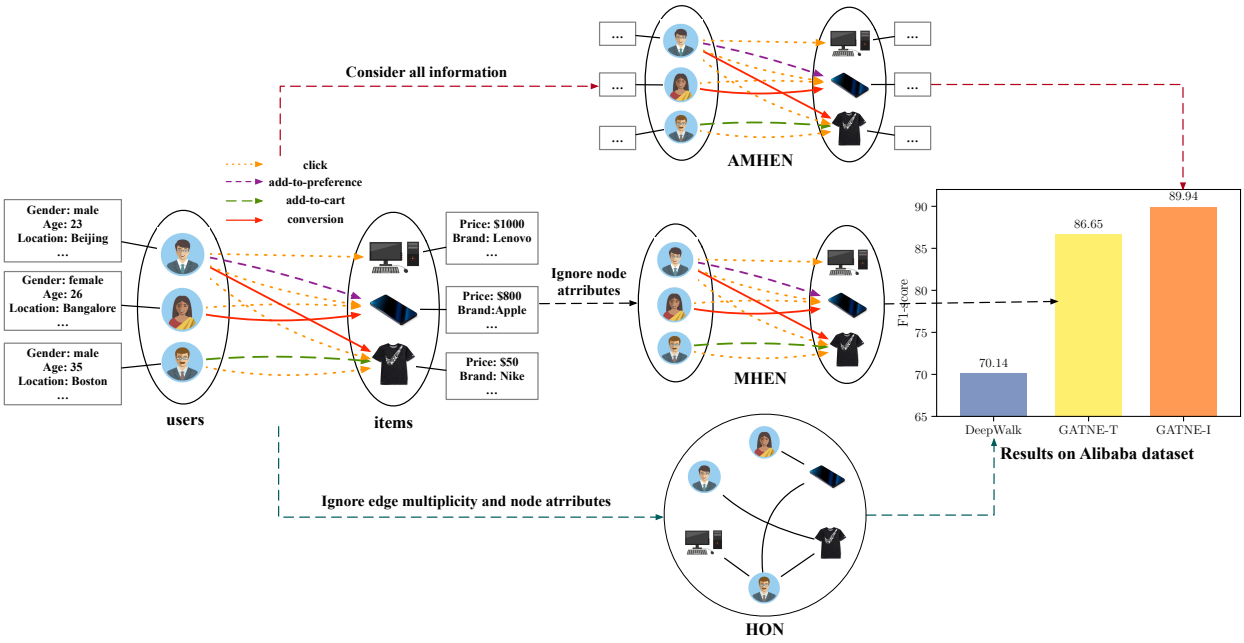


Figure 1: The left illustrates an example of an attributed multiplex heterogeneous network. Users in the left part of the figure are associated with attributes, including gender, age, and location. Similarly, items in the left part of the figure include attributes such as price and brand. The edge types between users and items are from four interactions, including *click*, *add-to-preference*, *add-to-cart* and *conversion*. The three subfigures in the middle represent three different ways of setting up the graphs, including HON, MHEN, and AMHEN from the bottom to the top. The right part shows the performance improvement of the proposed models over DeepWalk on the Alibaba dataset. As can be seen, *GATNE*-I achieves a +28.23% performance lift compared to DeepWalk.

are 20.3% (Twitter), 21.6% (YouTube), 15.1% (Amazon) and 16.3% (Alibaba) of the linked node pairs having more than one type of edges respectively. As an instance, in an e-commerce system, users may have several types of interactions with items, such as click, conversion, add-to-cart, add-to-preference. Figure 1 illustrates such an example. Obviously, "users" and "items" have intrinsically different properties and shall not be treated equally. Moreover, different user-item interactions imply different levels of interests and should be treated differently. Otherwise, the system cannot precisely capture the user's behavioral patterns and preferences and would be insufficient for practical use.

Not merely because of the heterogeneity and multiplicity, in practice, dealing with AMHEN poses several unique challenges:

- *Multiplex Edges.* Each node pair may have multiple different types of relationships. It is important to be able to borrow strengths from different relationships and learn unified embeddings.
- *Partial Observations.* The real networked data are usually partially observed. For example, a long-tailed customer may only present few interactions with some products. Most existing network embedding methods focus on the transductive settings, and cannot handle the long-tailed or cold-start problems.
- *Scalability.* Real networks usually have billions of nodes and tens or hundreds of billions of edges [40]. It is important to develop learning algorithms that can scale well to large networks.

Table 1: The network types handled by different methods.

Network Type	Method	Heterogeneity		Attribute
		Node Type	Edge Type	
Homogeneous Network (HON)	DeepWalk [27] LINE [35] node2vec [10] NetMF [29] NetSMF [28]	Single	Single	/
Attributed Homogeneous Network (AHON)	TADW [41] LANE [16] AANE [15] SNE [20] DANE [9] ANRL [44]	Single	Single	Attributed
Heterogeneous Network (HEN)	PTE [34] metapath2vec [7] HERec [31]	Multi	Single	/
Attributed HEN (AHEN)	HNE [3]	Multi	Single	Attributed
Multiplex Heterogeneous Network (MHEN)	PMNE [22] MVE [30] MNE [43] mvn2vec [32]	Single	Multi	/
	GATNE-T	Multi	Multi	
Attributed MHEN (AMHEN)	*GATNE*-I	Multi	Multi	Attributed

To address the above challenges, we propose a novel approach to capture both rich attributed information and to utilize multiplex topological structures from different node types, namely General

Attributed Multiplex He**T**erogeneous **N**etwork **E**mbedding, or abbreviated as *GATNE*. The key features of *GATNE* are the following:

- We formally define the problem of attributed multiplex heterogeneous network embedding, which is a more general representation for real-world networks.
- *GATNE* supports both transductive and inductive embeddings learning for attributed multiplex heterogeneous networks. We also give the theoretical analysis to prove that our transductive model is a more general form than existing models (e.g., MNE [43]).
- Efficient and scalable learning algorithms for *GATNE* have been developed. Our learning algorithms are able to handle hundreds of million nodes and billions of edges efficiently.

We conduct extensive experiments to evaluate the proposed models on four different genres of datasets: Amazon, YouTube, Twitter, and Alibaba. Experimental results show that the proposed framework can achieve statistically significant improvements (~5.99-28.23% lift by F1 scores on Alibaba dataset; $p \ll 0.01$, $t-$test) over state-of-the-art methods. We have deployed the proposed model on Alibaba's distributed system and apply the method to Alibaba's recommendation engine. Offline A/B tests further confirm the effectiveness and efficiency of our proposed models.

2 RELATED WORK

In this section, we review related state-of-the-arts for network embedding, heterogeneous network embedding, multiplex heterogeneous network embedding, and attributed network embedding.

Network Embedding. Works in network embedding mainly consist of two categories, graph embedding (GE) and graph neural network (GNN). Representative works for GE include DeepWalk [27] which generates a corpus on graphs by random walk and then trains a skip-gram model on the corpus. LINE [35] learns node presentations on large-scale networks while preserving both first-order and second-order proximities. node2vec [10] designs a biased random walk procedure to efficiently explore diverse neighborhoods. NetMF [29] is a unified matrix factorization framework for theoretically understanding and improving DeepWalk and LINE. For popular works in GNN, GCN [19] incorporates neighbors' feature representations into the node feature representation using convolutional operations. GraphSAGE [11] provides an inductive approach to combine structural information with node features. It learns functional representations instead of direct embeddings for each node, which helps it work inductively on unobserved nodes during training.

Heterogeneous Network Embedding. Heterogeneous networks examine scenarios with nodes and/or edges of various types. Such networks are notoriously difficult to mine because of the bewildering combination of heterogeneous contents and structures. The creation of a multidimensional embedding of such data opens the door to the use of a wide variety of off-the-shelf mining techniques for multidimensional data. Despite the importance of this problem, limited efforts have been made on embedding a scalable network of dynamic and heterogeneous data. HNE [3] jointly considers the contents and topological structures in networks and represents different objects in heterogeneous networks to unified vector representations. PTE [34] constructs large-scale heterogeneous text network from labeled information and different levels of word co-occurrence information, which is then embedded into a low-dimensional space. metapath2vec [7] formalizes meta-path based random walk to construct the heterogeneous neighborhood of a node and then leverages a heterogeneous skip-gram model to perform node embeddings. HERec [31] uses a meta-path based random walk strategy to generate meaningful node sequences to learn network embeddings that are first transformed by a set of fusion functions and subsequently integrated into an extended matrix factorization (MF) model.

Multiplex Heterogeneous Network Embedding. Existing approaches usually study networks with a single type of proximity between nodes, which only captures a single view of a network. However, in reality, there usually exist multiple types of proximities between nodes, yielding networks with multiple views, or multiplex network embedding. PMNE [22] proposes three methods to project a multiplex network into a continuous vector space. MVE [30] embeds networks with multiple views in a single collaborated embedding using attention mechanism. MNE [43] uses one common embedding and several additional embeddings of each edge type for each node, which are jointly learned by a unified network embedding model. Mvn2vec [32] explores the feasibility to achieve better embedding results by simultaneously modeling preservation and collaboration to represent semantic meanings of edges in different views respectively.

Attributed Network Embedding. Attributed network embedding aims to seek for low-dimensional vector representations for nodes in a network, such that original network topological structure and node attribute proximity can be preserved in such representations. TADW [41] incorporates text features of vertices into network representation learning under the framework of matrix factorization. LANE [16] smoothly incorporates label information into the attributed network embedding while preserving their correlations. AANE [15] enables a joint learning process to be done in a distributed manner for accelerated attributed network embedding. SNE [20] proposes a generic framework for embedding social networks by capturing both the structural proximity and attribute proximity. DANE [9] can capture the high nonlinearity and preserve various proximities in both topological structure and node attributes. ANRL [44] uses a neighbor enhancement autoencoder to model the node attribute information and an attribute-aware skip-gram model based on the attribute encoder to capture the network structure.

3 PROBLEM DEFINITION

Denote a network $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of n nodes and $\mathcal{E}$ is a set of edges between nodes. Each edge $e_{ij} = (v_i, v_j)$ is associated with a weight $w_{ij} \geq 0$, indicating the strength of the relationship between v_i and v_j. In practice, the network could be either directed or undirected. If G is directed, we have $e_{ij} \neq e_{ji}$ and $w_{ij} \neq w_{ji}$; if G is undirected, we have $e_{ij} \equiv e_{ji}$ and $w_{ij} \equiv w_{ji}$. Notations are summarized in Table 2.

Table 2: Notations.

Notation	Description
G	the input network
$\mathcal{V}$, $\mathcal{E}$	the node/edge set of G
$\mathcal{O}$, $\mathcal{R}$	the node/edge type set of G
$\mathcal{A}$	the attribute set of G
n	the number of nodes
m	the number of edge types
r	an edge type
d	the dimension of base/overall embeddings
s	the dimension of edge embeddings
v	a node in the graph
$\mathcal{N}$	the neighborhood set of a node on an edge type
b, u, c, v	the base/edge/context/overall embedding of a node
h, g	transformation functions in our inductive approach
x	the attribute of a node

Definition 1 (Heterogeneous Network). *A **heterogeneous network** [3, 33] is a network $G = (\mathcal{V}, \mathcal{E})$ associated with a node type mapping function $\phi : \mathcal{V} \rightarrow \mathcal{O}$ and an edge type mapping function $\psi : \mathcal{E} \rightarrow \mathcal{R}$, where $\mathcal{O}$ and $\mathcal{R}$ represent the set of all node types and the set of all edge types, respectively. Each node $v \in \mathcal{V}$ belongs to a particular node type. Similarly, each edge $e \in \mathcal{E}$ is categorized into a specific edge type. If $|\mathcal{O}| + |\mathcal{R}| > 2$, the network is called **heterogeneous**; otherwise **homogeneous**.*

Notice that in a heterogeneous network, an edge e can no longer be denoted as e_{ij} since there may be multiple types of edges between node v_i and v_j. Under such situations, an edge is denoted as $e_{ij}^{(r)}$, where r corresponds to a certain edge type.

Definition 2 (Attributed Network). *An **attributed network** [3, 16] is a network G endowed with an attribute representation $\mathcal{A}$, i.e., $G = (\mathcal{V}, \mathcal{E}, \mathcal{A})$. Each node $v_i \in \mathcal{V}$ is associated with some types of feature vectors. $\mathcal{A} = \{\mathbf{x}_i | v_i \in \mathcal{V}\}$ is the set of node features for all nodes, where $\mathbf{x}_i$ is the associated node feature of node v_i.*

Definition 3 (Attributed Multiplex Heterogeneous Network). *An **attributed multiplex heterogeneous network** is a network $G = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, $\mathcal{E} = \bigcup_{r \in \mathcal{R}} \mathcal{E}_r$, where $\mathcal{E}_r$ consists of all edges with edge type $r \in \mathcal{R}$, and $|\mathcal{R}| > 1$. We separate the network for every edge type or view $r \in \mathcal{R}$ as $G_r = (\mathcal{V}, \mathcal{E}_r, \mathcal{A})$.*

An example of AMHEN is illustrated in Figure 1, which contains 2 node types and 4 edge types. The two node types are *user* and *item* with different attributes. Given the above definitions, we can formally define our problem for representation learning on networks.

Problem 1 (AMHEN Embedding). *Given an AMHEN $G = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, the problem of **AMHEN embedding** is to give a unified low-dimensional space representation of each node v on every edge type r. The goal is to find a function $f_r : \mathcal{V} \rightarrow \mathbb{R}^d$ for every edge type r, where $d \ll |\mathcal{V}|$.*

4 METHODOLOGY

In this section, we first explain the proposed *GATNE* framework in the transductive context [19]. The resultant model is named as *GATNE*-T. We also give a theoretical discussion about the connection of *GATNE*-T with the newly trending models, e.g., MNE. To deal with the partial observation problem, we further extend the model to the inductive context [42] and present a new inductive model named as *GATNE*-I. For both models, we present efficient optimization algorithms.

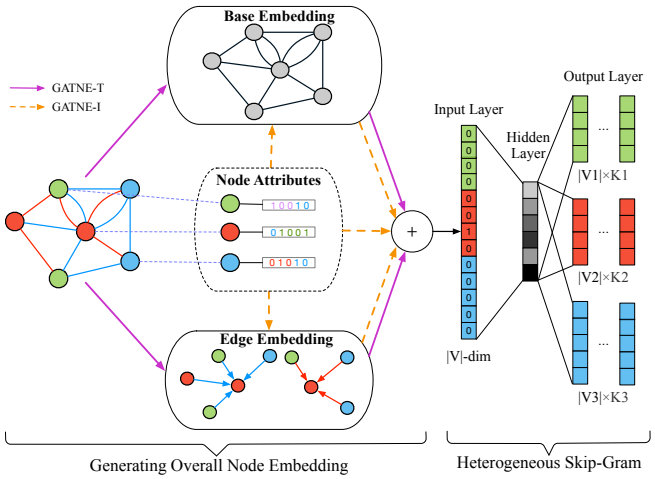


Figure 2: Illustration of *GATNE*-T and *GATNE*-I models. *GATNE*-T only uses network structure information while *GATNE*-I considers both structure information and node attributes. The output layer of heterogeneous skip-gram specifies one set of multinomial distributions for each node type in the neighborhood of the input node v. In this example, $\mathcal{V} = \mathcal{V}1 \cup \mathcal{V}2 \cup \mathcal{V}3$ and $K1, K2, K3$ specify the size of v's neighborhood on each node type respectively.

4.1 Transductive Model: *GATNE*-T

We begin with embedding learning for attributed multiplex heterogeneous networks in the transductive context, and present our model *GATNE*-T. More specifically, in *GATNE*-T, we split the overall embedding of a certain node v_i on each edge type r into two parts: base embedding, edge embedding as shown in Figure 2. The base embedding of node v_i is shared between different edge types. The k-th level edge embedding $\mathbf{u}_{i,r}^{(k)} \in \mathbb{R}^s, (1 \leq k \leq K)$ of node v_i on edge type r is aggregated from neighbors' edge embeddings as:

$$\mathbf{u}_{i,r}^{(k)} = aggregator(\{\mathbf{u}_{j,r}^{(k-1)}, \forall v_j \in \mathcal{N}_{i,r}\}), \quad (1)$$

where $\mathcal{N}_{i,r}$ is the neighbors of node v_i on edge type r. The initial edge embedding $\mathbf{u}_{i,r}^{(0)}$ for each node v_i and each edge type r is randomly initialized in our transductive model. Following GraphSAGE [11], the *aggregator* function can be a mean aggregator as:

$$\mathbf{u}_{i,r}^{(k)} = \sigma(\hat{\mathbf{W}}^{(k)} \cdot \text{mean}(\{\mathbf{u}_{j,r}^{(k-1)}, \forall v_j \in \mathcal{N}_{i,r}\})), \quad (2)$$

or other pooling aggregators such as max-pooling aggregator:

$$\mathbf{u}_{i,r}^{(k)} = \max(\{\sigma(\hat{\mathbf{W}}_{pool}^{(k)} \mathbf{u}_{j,r}^{(k-1)} + \hat{\mathbf{b}}_{pool}^{(k)}), \forall v_j \in \mathcal{N}_{i,r}\}), \quad (3)$$

where σ is an activation function. We denote the K-th level edge embedding $\mathbf{u}_{i,r}^{(K)}$ as edge embedding $\mathbf{u}_{i,r}$, and concatenate all the

edge embeddings for node v_i as $\mathbf{U}_i$ with size s-by-m, where s is the dimension of edge-embeddings:

$$\mathbf{U}_i = (\mathbf{u}_{i,1}, \mathbf{u}_{i,2}, \ldots, \mathbf{u}_{i,m}). \tag{4}$$

We use self-attention mechanism [21] to compute the coefficients $\mathbf{a}_{i,r} \in \mathbb{R}^m$ of linear combination of vectors in $\mathbf{U}_i$ on edge type r as:

$$\mathbf{a}_{i,r} = \text{softmax}(\mathbf{w}_r^T \tanh(\mathbf{W}_r \mathbf{U}_i))^T, \tag{5}$$

where $\mathbf{w}_r$ and $\mathbf{W}_r$ are trainable parameters for edge type r with size d_a and $d_a \times s$ respectively and the superscript T denotes the transposition of the vector or the matrix. Thus, the overall embedding of node v_i for edge type r is:

$$\mathbf{v}_{i,r} = \mathbf{b}_i + \alpha_r \mathbf{M}_r^T \mathbf{U}_i \mathbf{a}_{i,r}, \tag{6}$$

where $\mathbf{b}_i$ is the base embedding for node v_i, α_r is a hyper-parameter denoting the importance of edge embeddings towards the overall embedding and $\mathbf{M}_r \in \mathbb{R}^{s \times d}$ is a trainable transformation matrix.

Connection with Previous Work. We choose MNE [43], a recent representative work for MHEN, as the base model for multiplex heterogeneous networks to discuss the connection between our proposed model and previous work. In *GATNE*-T, we use the attention mechanism to capture the influential factors between different edge types. We theoretically prove that our transductive model is a more general form of MNE and improves the expressiveness. For MNE, the overall embedding $\tilde{\mathbf{v}}_{i,r}$ of node v_i on edge type $r \in \mathcal{R}$ is

$$\tilde{\mathbf{v}}_{i,r} = \mathbf{b}_i + \alpha_r \mathbf{X}_r^T \mathbf{o}_{i,r}, \tag{7}$$

where $\mathbf{X}_r$ is a edge-specific transformation matrix. And for *GATNE*-T, the overall embedding for node v_i on edge type r is:

$$\mathbf{v}_{i,r} = \mathbf{b}_i + \alpha_r \mathbf{M}_r^T \mathbf{U}_i \mathbf{a}_{i,r} = \mathbf{b}_i + \alpha_r \mathbf{M}_r^T \sum_{p=1}^{m} \lambda_p \mathbf{u}_{i,p}, \tag{8}$$

where λ_p denotes the p-th element of $\mathbf{a}_{i,r}$ and is computed as:

$$\lambda_p = \frac{\exp(\mathbf{w}_r^T \tanh(\mathbf{W}_r \mathbf{u}_{i,p}))}{\sum_t \exp(\mathbf{w}_r^T \tanh(\mathbf{W}_r \mathbf{u}_{i,t}))}. \tag{9}$$

Theorem 4.1. *For any $r \in \mathcal{R}$, there exist parameters $\mathbf{w}_r$ and $\mathbf{W}_r$, such that for any $\mathbf{o}_{i,1}, \ldots, \mathbf{o}_{i,m}$, and corresponding matrix $\mathbf{X}_r$, with dimension s for each $\mathbf{o}_{i,j}$ and size $s \times d$ for $\mathbf{X}_r$, there exist $\mathbf{u}_{i,1}, \ldots, \mathbf{u}_{i,m}$, and corresponding matrix $\mathbf{M}_r$, with dimension $s + m$ for each $\mathbf{u}_{i,j}$ and size $(s + m) \times d$ for $\mathbf{M}_r$, that satisfy $\mathbf{v}_{i,r} \approx \tilde{\mathbf{v}}_{i,r}$.*

Proof. For any t, let $\mathbf{u}_{i,t} = (\mathbf{o}_{i,t}^T, \mathbf{u}_{i,t}'^T)^T$ concatenated by two vectors, where the first s dimension is $\mathbf{o}_{i,t}$, and $\mathbf{u}'_{i,t}$ is an m-dimensional vector. Let $u'_{i,t,k}$ denote the k-th dimension of $\mathbf{u}'_{i,t}$, then take $u'_{i,t,k} = \delta_{tk}$, where δ is the Kronecker delta function as

$$\delta_{ij} = 1, i = j; \delta_{ij} = 0, i \neq j. \tag{10}$$

Let $\mathbf{W}_r$ be all zero, except for the element on row 1 and column $s+r$ with a large enough number M; therefore $\tanh(\mathbf{W}_r \mathbf{u}_{i,p})$ becomes a vector with its 1^{st} dimension approximately being δ_{rp}, and other dimensions being 0. Take $\mathbf{w}_r$ as a vector with its 1^{st} dimension being M, then $\exp(\mathbf{w}_r^T \tanh(\mathbf{W}_r \mathbf{u}_{i,p})) \approx \exp(M\delta_{rp})$, so $\lambda_p \approx \delta_{rp}$. Finally we take $\mathbf{M}_r$ being $\mathbf{X}_r$ at its first $s \times d$ dimension, and 0 on the following $m \times d$ dimension, and we can get $\mathbf{v}_{i,r} \approx \tilde{\mathbf{v}}_{i,r}$. □

Thus the parameter space of MNE is almost included by our model's parameter space and we can conclude that *GATNE*-T is a generalization of MNE, if edge embeddings can be trained directly. However, in our model, the edge embedding $\mathbf{u}$ is generated from single or multiple layers of aggregation. We give more discussions about the aggregation case.

Effects of Aggregation. In the *GATNE*-T model, the edge embedding $\mathbf{u}^{(k)}$ is computed by aggregating the edge embedding $\mathbf{u}^{(k-1)}$ of its neighbors as:

$$\mathbf{u}_{i,r}^{(k)} = \sigma(\hat{\mathbf{W}}^{(k)} \cdot \text{mean}(\{\mathbf{u}_{j,r}^{(k-1)}, v_j \in \mathcal{N}_{i,r}\})). \tag{11}$$

The mean aggregator is basically a matrix multiplication,

$$\text{mean}_k(v_i) = \text{mean}(\{\mathbf{u}_{j,r}^{(k-1)}, v_j \in \mathcal{N}_{i,r}\}) = (\mathbf{U}_r^{(k-1)} \mathbf{N}_r)_i, \tag{12}$$

where $\mathbf{N}_r$ is the neighborhood matrix on edge type r, $\mathbf{U}_r^{(k)} = (\mathbf{u}_{1,r}^{(k)}, ..., \mathbf{u}_{n,r}^{(k)})$ is the k-th level edge embedding of all nodes in the graph on edge type r, and $(\mathbf{U}_r^{(k-1)} \mathbf{N}_r)_i$ denotes the i^{th} column of $\mathbf{U}_r^{(k-1)} \mathbf{N}_r$. $\mathbf{N}_r$ can be a normalized adjacency matrix. The mean operator of Equation (11) can be weighted and the neighborhood matrix $\mathcal{N}_{i,r}$ can be sampled. Take $\hat{\mathbf{W}}^{(k)} = \mathbf{I}$, where $\mathbf{I}$ is an identity matrix, then $\mathbf{u}_{i,r}^{(k)} = \sigma(\text{mean}_k(v_i))$. If $\mathbf{N}_r$ is of full rank, then for any $\mathbf{O}_r = (\mathbf{o}_{1,r}, \ldots, \mathbf{o}_{n,r})$, there exists $\mathbf{U}_r^{(k-1)}$ such that $\mathbf{U}_r^{(k-1)} \mathbf{N}_r = \mathbf{O}_r$.

If the activation function σ is an automorphism, i.e., $\sigma : \mathbb{R} \to \mathbb{R}$ and $\mathbf{N}_r$ is of full rank, we can use the construction method described in Theorem 4.1 to construct $\mathbf{u}_{i,r}$ and the above method to construct each level of edge embeddings $\mathbf{u}_{i,r}^{(K-1)}, ..., \mathbf{u}_{i,r}^{(0)}$ subsequently. Therefore, our model is still a more general form that can generalize the MNE model, when all the neighborhood matrices $\mathbf{N}_r$ and the activation function σ are invertible in all levels of aggregation.

4.2 Inductive Model: *GATNE*-I

The limitation of *GATNE*-T is that it cannot handle unobserved data. However, in many real-world applications, the networked data is often partially observed [42]. We then extend our model to the inductive context and present a new model named *GATNE*-I. The model is also illustrated in Figure 2. Specifically, we define the base embedding $\mathbf{b}_i$ as a parameterized function of v_i's attribute $\mathbf{x}_i$ as $\mathbf{b}_i = \mathbf{h}_z(\mathbf{x}_i)$, where $\mathbf{h}_z$ is a transformation function and $z = \phi(i)$ is node v_i's corresponding node type. Notice that nodes with different types may have different dimensions of their attributes $\mathbf{x}_i$. The transformation function $\mathbf{h}_z$ can have different forms such as a multi-layer perceptron [26]. Similarly, the initial edge embeddings $\mathbf{u}_{i,r}^{(0)}$ for node i on edge type r should be also parameterized as the function of attributes $\mathbf{x}_i$ as $\mathbf{u}_{i,r}^{(0)} = \mathbf{g}_{z,r}(\mathbf{x}_i)$, where $\mathbf{g}_{z,r}$ is also a transformation function that transforms the feature to an edge embedding of node v_i on the edge type r and z is v_i's corresponding node type. To be more specific, for the inductive model, we also add an extra attribute term to the overall embedding of node v_i on type r:

$$\mathbf{v}_{i,r} = \mathbf{h}_z(\mathbf{x}_i) + \alpha_r \mathbf{M}_r^T \mathbf{U}_i \mathbf{a}_{i,r} + \beta_r \mathbf{D}_z^T \mathbf{x}_i, \tag{13}$$

where β_r is a coefficient and $\mathbf{D}_z$ is a feature transformation matrix on v_i's corresponding node type z. The difference between our

Algorithm 1: *GATNE*

Input: network $G = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, embedding dimension d, edge embedding dimension s, learning rate η, negative samples L, coefficient α, β.
Output: overall embeddings $\mathbf{v}_{i,r}$ for all nodes on every edge type r

```
Initialize all the model parameters θ.
Generate random walks on each edge type r as P_r.
Generate training samples {(v_i, v_j, r)} from random walks P_r on each edge type r.
while not converged do
    foreach (v_i, v_j, r) ∈ training samples do
        Calculate v_{i,r} using Equation (6) or (13)
        Sample L negative samples and calculate objective function E using Equation (17)
        Update model parameters θ by ∂E/∂θ.
```

transductive and inductive model mainly lies on how the base embedding $\mathbf{b}_i$ and initial edge embeddings $\mathbf{u}_{i,r}^{(0)}$ are generated. In our transductive model, the base embedding $\mathbf{b}_i$ and initial edge embedding $\mathbf{u}_{i,r}^{(0)}$ are directly trained for each node based on the network structure, and the transductive model cannot handle nodes that are not seen during training. As for our inductive model, instead of training $\mathbf{b}_i$ and $\mathbf{u}_{i,r}^{(0)}$ directly for each node, we train transformation functions $\mathbf{h}_z$ and $\mathbf{g}_{z,r}$ that transforms the raw feature $\mathbf{x}_i$ to $\mathbf{b}_i$ and $\mathbf{u}_{i,r}^{(0)}$, which works for nodes that did not appear during training as long as they have corresponding raw features.

4.3 Model Optimization

We discuss how to learn the proposed transductive and inductive models. Following [10, 27, 35], we use random walk to generate node sequences and then perform skip-gram [24, 25] over the node sequences to learn embeddings. Since each view of the input network is heterogeneous, we use meta-path-based random walks [7]. To be specific, given a view r of the network, i.e., $G_r = (\mathcal{V}, \mathcal{E}_r, \mathcal{A})$ and a meta-path scheme $\mathcal{T} : \mathcal{V}_1 \rightarrow \mathcal{V}_2 \rightarrow \cdots \mathcal{V}_t \cdots \rightarrow \mathcal{V}_l$, where l is the length of the meta-path scheme, the transition probability at step t is defined as follows:

$$p(v_j|v_i, \mathcal{T}) = \begin{cases} \frac{1}{|\mathcal{N}_{i,r} \cap \mathcal{V}_{t+1}|} & (v_i, v_j) \in \mathcal{E}_r, v_j \in \mathcal{V}_{t+1}, \\ 0 & (v_i, v_j) \in \mathcal{E}_r, v_j \notin \mathcal{V}_{t+1}, \\ 0 & (v_i, v_j) \notin \mathcal{E}_r, \end{cases} \quad (14)$$

where $v_i \in \mathcal{V}_t$ and $\mathcal{N}_{i,r}$ denotes the neighborhood of node v_i on edge type r. The flow of the walker is conditioned on the predefined meta path $\mathcal{T}$. The meta-path-based random walk strategy ensures that the semantic relationships between different types of nodes can be properly incorporated into skip-gram model [7]. Supposing the random walk with length l on edge type r follows a path $P = (v_{p_1}, \ldots, v_{p_l})$ such that $(v_{p_{t-1}}, v_{p_t}) \in \mathcal{E}_r (t = 2 \ldots l)$, denote v_{p_t}'s context as $C = \{v_{p_k} | v_{p_k} \in P, |k-t| \leq c, t \neq k\}$, where c is the radius of the window size.

Table 3: Statistics of Datasets.

Dataset	# nodes	# edges	# n-types	# e-types
Amazon	10,166	148,865	1	2
YouTube	2,000	1,310,617	1	5
Twitter	10,000	331,899	1	4
Alibaba-S	6,163	17,865	2	4
Alibaba	41,991,048	571,892,183	2	4

Thus, given a node v_i with its context C of a path, our objective is to minimize the following negative log-likelihood:

$$-\log P_\theta\left(\{v_j|v_j \in C\}|v_i\right) = \sum_{v_j \in C} -\log P_\theta(v_j|v_i), \quad (15)$$

where θ denotes all the parameters. Following metapath2vec [7] we use the heterogeneous softmax function which is normalized with respect to the node type of node v_j. Specifically, the probability of v_j given v_i is defined as:

$$P_\theta(v_j|v_i) = \frac{\exp(\mathbf{c}_j^T \cdot \mathbf{v}_{i,r})}{\sum_{k \in \mathcal{V}_t} \exp(\mathbf{c}_k^T \cdot \mathbf{v}_{i,r})}, \quad (16)$$

where $v_j \in \mathcal{V}_t$, $\mathbf{c}_k$ is the context embedding of node v_k and $\mathbf{v}_i$ is the overall embedding of node v_i for edge type r.

Finally, we use heterogeneous negative sampling to approximate the objective function $-\log P_\theta(v_j|v_i)$ for each node pair (v_i, v_j) as:

$$E = -\log \sigma(\mathbf{c}_j^T \cdot \mathbf{v}_{i,r}) - \sum_{l=1}^{L} \mathbb{E}_{v_k \sim P_t(v)}[\log \sigma(-\mathbf{c}_k^T \cdot \mathbf{v}_{i,r})], \quad (17)$$

where $\sigma(x) = 1/(1+\exp(-x))$ is the sigmoid function, L is the number of negative samples correspond to a positive training sample, and v_k is randomly drawn from a noise distribution $P_t(v)$ defined on node v_j's corresponding node set $\mathcal{V}_t$.

We summarize our algorithm in Algorithm 1. The time complexity of our random walk based algorithm is $O(nmdL)$ where n is the number of nodes, m is the number of edge types, d is overall embedding size, L is the number of negative samples per training sample ($L \geq 1$). The memory complexity of our algorithm is $O(n(d + m \times s))$ with s being the size of edge embedding.

5 EXPERIMENTS

In this section, we first introduce the details of four evaluation datasets and the competitor algorithms. We focus on the link prediction task to evaluate the performances of our proposed methods compared to other state-of-the-art methods. Parameter sensitivity, convergence, and scalability are then discussed. Finally, we report the results of offline A/B tests of our method on Alibaba's recommendation system.

5.1 Datasets

We work on three public datasets and the Alibaba dataset for the link prediction task. Amazon Product Dataset[2] [13, 23] includes product metadata and links between products; YouTube dataset[3] [36, 38]

[2] http://jmcauley.ucsd.edu/data/amazon/
[3] http://socialcomputing.asu.edu/datasets/YouTube

consists of various types of interactions; Twitter dataset[4] [6] also contains various types of links. Alibaba dataset has two node types, user and item (or product), and includes four types of interactions between users and items. Since some of the baselines cannot scale to the whole graph, we evaluate performances on sampled datasets. The statistics of these four sampled datasets are summarized in Table 3. Notice that *n-types* and *e-types* in the table denote node types and edge types, respectively.

Amazon. In our experiments, we only use the product metadata of *Electronics* category, including the product attributes and co-viewing, co-purchasing links between products. The product attributes include the price, sales-rank, brand, and category.

YouTube. YouTube dataset is a multiplex bidirectional network dataset that consists of five types of interactions between 15,088 YouTube users. The types of edges include contact, shared friends, shared subscription, shared subscriber, and shared favorite videos between users.

Twitter. Twitter dataset is about tweets related to the discovery of the Higgs boson between 1st and 7th, July 2012. It is made up of four directional relationships between more than 450,000 Twitter users. The relationships are re-tweet, reply, mention, and friendship/follower relationships between Twitter users.

Alibaba. Alibaba dataset consists of four types of interactions including click, add-to-preference, add-to-cart, and conversion between two types of nodes, user and item. The sampled Alibaba dataset is denoted as **Alibaba-S**. We also provide the evaluation of the whole dataset on Alibaba's distributed cloud platform; the full dataset is denoted as **Alibaba**.

5.2 Competitors

We categorize our competitors into the following four groups. The overall embedding size is set to 200 for all methods. The specific hyper-parameter settings for different methods are listed in the Appendix.

Network Embedding Methods. The compared methods include DeepWalk [27], LINE [35], and node2vec [10]. As these methods can only deal with HON, we feed separate graphs with different edge types to them and obtain different node embeddings for each separate graph.

Heterogeneous Network Embedding Methods. We focus on the representative work metapath2vec [7], which is designed to deal with the node heterogeneity. When there is only one node type in the network, metapath2vec degrades to DeepWalk. For Alibaba dataset, the meta-path schemes are set to $U-I-U$ and $I-U-I$, where U and I denote *User* and *Item* respectively.

Multiplex Heterogeneous Network Embedding Methods. The compared methods include PMNE [22], MVE [30], MNE [43]. We denote the three methods of PMNE as PMNE(n), PMNE(r) and PMNE(c) respectively. MVE uses collaborated context embeddings and applies an attention mechanism to view-specific embedding. MNE uses one common embedding and several additional embeddings for each edge type, which are jointly learned by a unified network embedding model.

Attributed Network Embedding Methods. The compared method is ANRL [44]. ANRL uses a neighbor enhancement auto-encoder to model the node attribute information and an attribute-aware skip-gram model based on the attribute encoder to capture the network structure.

Our Methods. Our proposed methods include *GATNE*-T and *GATNE*-I. *GATNE*-T considers the network structure and uses base embeddings and edge embeddings to capture the influential factors between different edge types. *GATNE*-I considers both the network structure and the node attributes, and learns an inductive transformation function instead of learning base embeddings and meta embeddings for each node directly. For Alibaba dataset, we use the same meta-path schemes as metapath2vec. For some datasets without node attributes, we also generate node features for them. Due to the size of the Alibaba dataset with more than 40 million nodes and 500 million edges and the scalabilities of the other competitors, we only compare our *GATNE* model with DeepWalk, MVE, and MNE. Specific implementations can be found in the Appendix.

5.3 Performance Analysis

Link prediction is a common task in both academia and industry. For academia, it is widely used to evaluate the quality of network embeddings obtained by different methods. In the industry, link prediction is a very demanding task since in real-world scenarios we are usually facing graphs with partial links, especially for e-commerce companies that rely on the links between their users and items for recommendations. We hide a set of edges/non-edges from the original graph and train on the remaining graph. Following [2, 18], we create a validation/test set that contains 5%/10% randomly selected positive edges respectively with the equivalent number of randomly selected negative edges for each edge type. The validation set is used for hyper-parameter tuning and early stopping. The test set is used to evaluate the performance and is only run once under the tuned hyper-parameter. We use some commonly used evaluation criteria, i.e., the area under the ROC curve (ROC-AUC) [12] and the PR curve (PR-AUC) [5] in our experiments. We also use F1 score as the other metric for evaluation. To avoid the thresholding effect [37], we assume that the number of hidden edges in the test set is given [27, 29, 37]. All of these metrics are uniformly averaged among the selected edge types.

Quantitative Results. The experimental results of three public datasets and Alibaba-S are shown in Table 4. *GATNE* outperforms all sorts of baselines in the various datasets. *GATNE*-T obtains better performance than *GATNE*-I on Amazon dataset as the node attributes are limited. The node attributes of Alibaba dataset are abundant so that *GATNE*-I obtains the best performance. ANRL is very sensitive to the weak node attributes and obtains the worst result on Amazon dataset. The different node attributes of users and items also limit the performance of ANRL on Alibaba-S dataset. On YouTube and Twitter datasets, *GATNE*-I performs similarly to *GATNE*-T as the node attributes of these two datasets are the node embeddings of DeepWalk, which are generated by the network

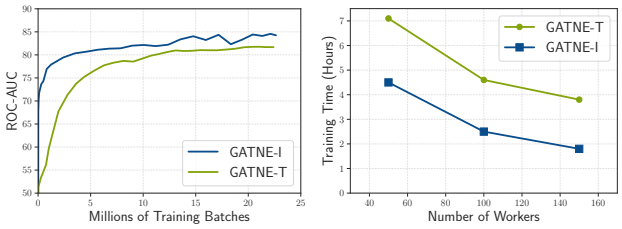
[4] https://snap.stanford.edu/data/higgs-twitter.html

Table 4: Performance comparison of different methods on four datasets.

	Amazon			YouTube			Twitter			Alibaba-S		
	ROC-AUC	PR-AUC	F1	ROC-AUC	PR-AUC	F1	ROC-AUC	PR-AUC	F1	ROC-AUC	PR-AUC	F1
DeepWalk	94.20	94.03	87.38	71.11	70.04	65.52	69.42	72.58	62.68	59.39	60.62	56.10
node2vec	94.47	94.30	87.88	71.21	70.32	65.36	69.90	73.04	63.12	62.26	63.40	58.49
LINE	81.45	74.97	76.35	64.24	63.25	62.35	62.29	60.88	58.18	53.97	54.65	52.85
metapath2vec	94.15	94.01	87.48	70.98	70.02	65.34	69.35	72.61	62.70	60.94	61.40	58.25
ANRL	71.68	70.30	67.72	75.93	73.21	70.65	70.04	67.16	64.69	58.17	55.94	56.22
PMNE(n)	95.59	95.48	89.37	65.06	63.59	60.85	69.48	72.66	62.88	62.23	63.35	58.74
PMNE(r)	88.38	88.56	79.67	70.61	69.82	65.39	62.91	67.85	56.13	55.29	57.49	53.65
PMNE(c)	93.55	93.46	86.42	68.63	68.22	63.54	67.04	70.23	60.84	51.57	51.78	51.44
MVE	92.98	93.05	87.80	70.39	70.10	65.10	72.62	73.47	67.04	60.24	60.51	57.08
MNE	90.28	91.74	83.25	82.30	82.18	75.03	91.37	91.65	84.32	62.79	63.82	58.74
GATNE-T	**97.44**	**97.05**	**92.87**	**84.61**	81.93	**76.83**	**92.30**	91.77	**84.96**	66.71	67.55	62.48
GATNE-I	96.25	94.77	91.36	84.47	**82.32**	**76.83**	92.04	**91.95**	84.38	**70.87**	**71.65**	**65.54**

Table 5: The experimental results on Alibaba dataset.

	ROC-AUC	PR-AUC	F1
DeepWalk	65.58	78.13	70.14
MVE	66.32	80.12	72.14
MNE	79.60	93.01	84.86
GATNE-T	81.02	93.39	86.65
GATNE-I	**84.20**	**95.04**	**89.94**

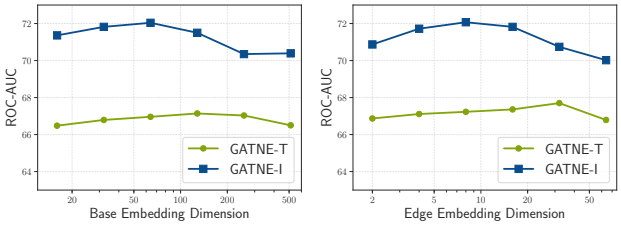


(a) Convergence (b) Scalability

Figure 3: (a) The convergence curve for *GATNE*-T and *GATNE*-I models on Alibaba dataset. The inductive model converges faster and achieves better performance than the transductive model. (b) The training time decreases as the number of workers increases. *GATNE*-I takes less training time to converge compared with *GATNE*-T.

structure. Table 5 lists the experimental results of Alibaba dataset. *GATNE* scales very well and achieves state-of-the-art performance on Alibaba dataset, with 2.18% performance lift in PR-AUC, 5.78% in ROC-AUC, and 5.99% in F1-score, compared with best results from previous state-of-the-art algorithms. The *GATNE*-I performs better than *GATNE*-T model in the large-scale dataset, suggesting that the inductive approach works better on large-scale attributed multiplex heterogeneous networks, which is usually the case in real-world situations.

Convergence Analysis. We analyze the convergence properties of our proposed models on Alibaba dataset. The results, as shown in Figure 3(a), demonstrate that *GATNE*-I converges faster and



(a) Base embedding dimension (b) Edge embedding dimension

Figure 4: The performance of *GATNE*-T and *GATNE*-I on Alibaba-S when changing base/edge embedding dimensions exponentially.

achieves better performance than *GATNE*-T on extremely large-scale real-world datasets.

Scalability Analysis. We investigate the scalability of *GATNE* that has been deployed on multiple workers for optimization. Figure 3(b) shows the speedup w.r.t. the number of workers on the Alibaba dataset. The figure shows that *GATNE* is quite scalable on the distributed platform, as the training time decreases significantly when we add up the number of workers, and finally, the inductive model takes less than 2 hours to converge with 150 distributed workers. We also find that *GATNE*-I's training speed increases almost linearly as the number of workers increases but less than 150. While *GATNE*-T converges slower and its training speed is about to reach a limit when the number of workers being larger than 100. Besides the state-of-the-art performance, *GATNE* is also scalable enough to be adopted in practice.

Parameter Sensitivity. We investigate the sensitivity of different hyper-parameters in *GATNE*, including overall embedding dimension d and edge embedding dimension s. Figure 4 illustrates the performance of *GATNE* when altering the base embedding dimension d or edge embedding dimension s from the default setting ($d = 200, s = 10$). We can conclude that the performance of *GATNE* is relatively stable within a large range of base/edge embedding dimensions, and the performance drops when the base/edge embedding dimension is either too small or too large.

5.4 Offline A/B Tests

We deploy our inductive model *GATNE*-I on Alibaba's distributed cloud platform for its recommendation system. The training dataset has about 100 million users and 10 million items, with 10 billion interactions between them per day. We use the model to generate embedding vectors for users and items. For every user, we use K nearest neighbor (KNN) with Euclidean distance to calculate the top-N items that the user is most likely to click. The experimental goal is to maximize top-N hit-rate. Under the framework of A/B tests, we conduct an offline test on *GATNE*-I, MNE, and DeepWalk. The results demonstrate that *GATNE*-I improves hit-rate by **3.26%** and **24.26%** compared to MNE and DeepWalk, respectively.

6 CONCLUSION

In our paper, we formalized the attributed multiplex heterogeneous network embedding problem and proposed *GATNE* to solve it with both transductive and inductive settings. We split the overall node embedding of *GATNE*-I into three parts: base embedding, edge embedding, and attribute embedding. The base embedding and attribute embedding are shared among edges of different types, while the edge embedding is computed by aggregation of neighborhood information with the self-attention mechanism. Our proposed methods achieve significantly better performances compared to previous state-of-the-art methods on link prediction tasks across multiple challenging datasets. The approach has been successfully deployed and evaluated on Alibaba's recommendation system with excellent scalability and effectiveness.

ACKNOWLEDGMENTS

We thank Qibin Chen, Ming Ding, Chang Zhou, and Xiaonan Fang for their comments. The work is supported by the NSFC for Distinguished Young Scholar (61825602), NSFC (61836013), and a research fund supported by Alibaba Group.

REFERENCES

[1] Smriti Bhagat, Graham Cormode, and S Muthukrishnan. 2011. Node classification in social networks. In *Social network data analytics*. Springer, 115–148.
[2] Aleksandar Bojchevski and Stephan GÃijnnemann. 2018. Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. In *ICLR'18*.
[3] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C Aggarwal, and Thomas S Huang. 2015. Heterogeneous network embedding via deep architectures. In *KDD'15*. ACM, 119–128.
[4] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. 2018. A survey on network embedding. *TKDE* (2018).
[5] Jesse Davis and Mark Goadrich. 2006. The relationship between Precision-Recall and ROC curves. In *ICML'06*. ACM, 233–240.
[6] Manlio De Domenico, Antonio Lima, Paul Mougel, and Mirco Musolesi. 2013. The anatomy of a scientific rumor. *Scientific reports* 3 (2013), 2980.
[7] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD'17*. ACM, 135–144.
[8] Santo Fortunato. 2010. Community detection in graphs. *Physics reports* 486, 3-5 (2010), 75–174.
[9] Hongchang Gao and Heng Huang. 2018. Deep Attributed Network Embedding.. In *IJCAI'18*. 3364–3370.
[10] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD'16*. ACM, 855–864.
[11] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NIPS'17*. 1024–1034.
[12] James A Hanley and Barbara J McNeil. 1982. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology* 143, 1 (1982), 29–36.
[13] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW'16*. 507–517.
[14] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[15] Xiao Huang, Jundong Li, and Xia Hu. 2017. Accelerated attributed network embedding. In *SDM'17*. SIAM, 633–641.
[16] Xiao Huang, Jundong Li, and Xia Hu. 2017. Label informed attributed network embedding. In *WSDM'17*. ACM, 731–739.
[17] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[18] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).
[19] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR'17*.
[20] Lizi Liao, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. 2018. Attributed social network embedding. *TKDE* 30, 12 (2018), 2257–2270.
[21] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *ICLR'17*.
[22] Weiyi Liu, Pin-Yu Chen, Sailung Yeung, Toyotaro Suzumura, and Lingli Chen. 2017. Principled multilayer network embedding. In *ICDMW'17*. IEEE, 134–141.
[23] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR'15*. ACM, 43–52.
[24] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *ICLR'13*.
[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS'13*. 3111–3119.
[26] Sankar K Pal and Sushmita Mitra. 1992. Multilayer Perceptron, Fuzzy Sets, Classifiaction. (1992).
[27] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD'14*. ACM, 701–710.
[28] 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 *WWW'19*.
[29] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *WSDM'18*. ACM, 459–467.
[30] 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 *CIKM'17*. ACM, 1767–1776.
[31] Chuan Shi, Binbin Hu, Xin Zhao, and Philip Yu. 2018. Heterogeneous Information Network Embedding for Recommendation. *TKDE* (2018).
[32] Yu Shi, Fangqiu Han, Xinran He, Carl Yang, Jie Luo, and Jiawei Han. 2018. mvn2vec: Preservation and Collaboration in Multi-View Network Embedding. *arXiv preprint arXiv:1801.06597* (2018).
[33] Yizhou Sun, Brandon Norick, Jiawei Han, Xifeng Yan, Philip S Yu, and Xiao Yu. 2013. Pathselclus: Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. *TKDD* 7, 3 (2013), 11.
[34] Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *KDD'15*. ACM, 1165–1174.
[35] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW'15*. 1067–1077.
[36] Lei Tang and Huan Liu. 2009. Uncovering cross-dimension group structures in multi-dimensional networks. In *SDM workshop on Analysis of Dynamic Networks*. ACM, 568–575.
[37] Lei Tang, Suju Rajan, and Vijay K Narayanan. 2009. Large scale multi-label classification via metalabeler. In *WWW'09*. ACM, 211–220.
[38] Lei Tang, Xufei Wang, and Huan Liu. 2009. Uncovering groups via heterogeneous interaction analysis. In *ICDM'09*. IEEE, 503–512.
[39] Ben Taskar, Ming-Fai Wong, Pieter Abbeel, and Daphne Koller. 2004. Link prediction in relational data. In *NIPS'04*. 659–666.
[40] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. 2018. Billion-scale Commodity Embedding for E-commerce Recommendation in Alibaba. *KDD'18*, 839–848.
[41] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. 2015. Network representation learning with rich text information.. In *IJCAI'15*. 2111–2117.
[42] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *ICML'16*. 40–48.
[43] Hongming Zhang, Liwei Qiu, Lingling Yi, and Yangqiu Song. 2018. Scalable Multiplex Network Embedding. In *IJCAI'18*. 3082–3088.
[44] 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'18*. 3155–3161.

A APPENDIX

In the appendix, we first give the implementation notes of our proposed models. The detailed descriptions of datasets and the parameter configurations of all methods are then given. Finally, we discuss the questions about fair comparison and our future work.

A.1 Implementation Notes

Running Environment. The experiments in this paper can be divided into two parts. One is conducted on four datasets using a single Linux server with 4 Intel(R) Xeon(R) Platinum 8163 CPU @ 2.50GHz, 512G RAM and 8 NVIDIA Tesla V100-SXM2-16GB. The codes of our proposed models in this part are implemented with TensorFlow[5] 1.12 in Python 3.6. The other part is conducted on the full Alibaba dataset using Alibaba's distributed cloud platform which contains thousands of workers. Every two workers share an NVIDIA Tesla P100 GPU with 16GB memory. Our proposed models are implemented with TensorFlow 1.4 in Python 2.7 in this part.

Implementation Details. Our codes used by single Linux server can be split into three parts: random walk, model training and evaluation. The random walk part is implemented referring to the corresponding part of DeepWalk[6] and metapath2vec[7]. The training part of the model is implemented referring to the word2vec part of TensorFlow tutorials[8]. The evaluation part uses some metric functions from scikit-learn[9] including *roc_auc_score*, *f1_score*, *precision_recall_curve*, *auc*. Our model parameters are updated and optimized by stochastic gradient descent with Adam updating rule [17]. The distributed version of our proposed models is implemented based on the coding rules of Alibaba's distributed cloud platform in order to maximize the distribution efficiency. High-level APIs, such as *tf.estimator* and *tf.data*, are used for the higher coefficient of utilization of computation resources in the Alibaba's distributed cloud platform.

Function Selection. Many different aggregator functions in Equation (1), such as the mean aggregator (Cf. Equation (2)) or pooling aggregator (Cf. Equation (3)), achieve similar performance in our experiments. Mean aggregator is finally used to be reported in the quantitative experiments in our model. We use the linear transformation function as the parameterized function of attributes $\mathbf{h}_z$ and $\mathbf{g}_{z,r}$ in Equation (13) of our inductive model *GATNE*-I.

Parameter Configuration. Our base/overall embedding dimension d is set to 200 and the dimension of edge embedding s is set to 10. The number of walks for each node is set to 20 and the length of walks is set to 10. The window size is set to 5 for generating node contexts. The number of negative samples L for each positive training sample is set to 5. The number of maximum epochs is set to 50 and our models will early stop if ROC-AUC on the validation set does not improve in 1 training epoch. The coefficient α_r and β_r are all set to 1 for every edge type r. For Alibaba dataset, the node types include U and I representing *User* and *Item* respectively. The meta-path-schemes of our methods are set to $U-I-U$ and $I-U-I$.

We use the default setting of Adam optimizer in TensorFlow; the learning rate is set to $lr = 0.001$. For offline A/B test in section 5.4, we use $N = 50$.

Code and Dataset Releasing Details. The codes of our proposed models on the single Linux server (based on Tensorflow 1.12), together with our partition of the three public datasets and the Alibaba-S dataset are available.

A.2 Compared Methods

We give the detailed running configuration about all compared methods as follows. The embedding size is set to 200 for all methods. For random-walk based methods, the number of walks for each node is set to 20 and the length of walks is set to 10. The window size is set to 5 for generating node contexts. The number of negative samples for each training pairs is set to 5. The number of iterations for training the skip-gram model is set to 100. The code sources and other specific hyper-parameter settings of compared methods are explained below.

A.2.1 Network Embedding Methods.

- **DeepWalk** [27]. For public and Alibaba-S datasets, we use the codes from the corresponding author's GitHub[6]. For Alibaba dataset, we re-implemented DeepWalk on Alibaba distributed cloud platform.
- **LINE** [35]. The codes of LINE are from the corresponding author's GitHub[10]. We use the LINE(1st+2nd) as the overall embeddings. The embedding size is set to 100 both for first-order and second-order embeddings. The number of samples is set to 1000 million.
- **node2vec** [10]. The codes of node2vec are from the corresponding author's GitHub[11]. Node2vec adds two parameters to control the random walk process. The parameter p is set to 2 and the parameter q is set to 0.5 in our experiments.

A.2.2 Heterogeneous Network Embedding Methods.

- **metapath2vec** [7]. The codes provided by the corresponding author are only for specific datasets and could not directly generalize to other datasets. We re-implement metapath2vec for networks with arbitrary node types in Python based on the original C++ codes[12]. As the number of node types of three public datasets is one, metapath2vec degrades to DeepWalk in these three datasets. For Alibaba dataset, the node types include U and I representing *User* and *Item* respectively. The meta-path-schemes are set to $U-I-U$ and $I-U-I$.

A.2.3 Multiplex Heterogeneous Network Embedding Methods.

- **PMNE** [22]. PMNE proposes three different methods to apply node2vec on multiplex networks. We denote their network aggregation algorithm, result aggregation algorithm, and layer co-analysis algorithm as PMNE(n), PMNE(r), and PMNE(c) respectively in accord with the denotations of

[5] https://www.tensorflow.org/
[6] https://github.com/phanein/deepwalk
[7] https://ericdongyx.github.io/metapath2vec/m2v.html
[8] https://www.tensorflow.org/tutorials/representation/word2vec
[9] https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics
[10] https://github.com/tangjianpku/LINE
[11] https://github.com/aditya-grover/node2vec
[12] https://ericdongyx.github.io/metapath2vec/m2v.html

Table 6: Statistics of Original Datasets.

Dataset	# nodes	# edges	# n-types	# e-types
Amazon	312,320	7,500,100	1	4
YouTube	15,088	13,628,895	1	5
Twitter	456,626	15,367,315	1	4

MNE[43]. We use the codes from MNE's GitHub[13]. The probability of traversing layers of PMNE(c) is set to 0.5.

- **MVE** [30]. MVE uses collaborated context embeddings and applies an attention mechanism to view-specific embeddings. The code of MVE was received from the corresponding author by email. The embedding dimensions for each view is set to 200. The number of training samples for each epoch is set to 100 million and the number of epochs is set to 10. As for Alibaba dataset, we re-implemented this method on the Alibaba distributed cloud platform.
- **MNE** [43]. MNE uses one common embedding and several additional embeddings of each edge type for each node, which are jointly learned by a unified network embedding model. The additional embedding size for MNE is set to 10. We use the codes released by the corresponding author in the GitHub[13]. As for Alibaba dataset, we re-implemented it on the Alibaba distributed cloud platform.

A.2.4 Attributed Network Embedding Methods.

- **ANRL** [44]. We use the codes from Alibaba's GitHub[14]. As YouTube and Twitter datasets do not have node attributes, we generate node attributes for them. To be specific, we use the node embeddings (200 dimensions) of DeepWalk as the input node features on these datasets for ANRL. For Alibaba-S and Amazon dataset, we use raw features as attributes.

A.3 Datasets

Our experiment evaluates on five datasets, including four datasets and Alibaba dataset. Due to the limitation of memory and computation resources on a single Linux server, the four datasets are subgraphs sampled from the original datasets for training and evaluation. Table 6 shows the statistics of the original public datasets.

- **Amazon** is a dataset of product reviews and metadata from Amazon. In our experiments, we only use the product metadata, including the product attributes and co-viewing, co-purchasing links between products. The node type set of Amazon is $O = \{product\}$ and the edge type set of Amazon is $\mathcal{R} = \{also\_bought, also\_viewed\}$, which denotes two products are co-bought or co-viewed by the same user respectively. The products of Amazon are split into many categories. The number of products in all the categories is so large that we use the *Electronics* category of products for experimentation. The number of products in *Electronics* is still large for many algorithms; therefore, we extract a connected subgraph from the whole graph.
- **YouTube** is a multi-dimensional bidirectional network dataset consists of 5 types of interactions between 15,088 YouTube users. The types of edges include contact, shared friends, shared subscription, shared subscriber, and shared favorite videos between users. It is a multiplex network with $|O| = 1$ and $|\mathcal{R}| = 5$.
- **Twitter** is a dataset about tweets posted on Twitter about the discovery of the Higgs boson between 1st and 7th, July 2012. It is made up of 4 directional relationships between more than 450,000 Twitter users. The relationships are retweet, reply, mention, and friendship/follower relationship between Twitter users. It is a multiplex network with $|O| = 1$ and $|\mathcal{R}| = 4$.
- **Alibaba** consists of 4 types of interactions which including click, add-to-preference, add-to-cart, and conversion between two types of nodes, user and item. The node type set of Alibaba is $O = \{user, item\}$ and the size of the edge type set is $|\mathcal{R}| = 4$. The whole graph of Alibaba is so large that we cannot evaluate the performances of different methods on it by a single machine. We extract a subgraph from the whole graph for comparison with different methods, denoted as **Alibaba-S**. By the way, we also provide the evaluation of the whole graph on the Alibaba's distributed cloud platform, the full graph is denoted as **Alibaba**.

A.4 Discussion

As for research on network embedding, many people use link prediction or node classification tasks for evaluating the representation of network embeddings. However, although there are many commonly used public datasets, like Twitter or YouTube dataset, none of them provide a "standard" separation for train, validation, and test for different tasks. This causes different results on the same dataset for different evaluation separations so the results from previous papers cannot be directly used, and researchers have to re-implement and run all baselines themselves, reducing their attention on improving their model.

Here we appeal on researchers to provide the standardized dataset, which contains a standard separation of train, validation and test sets as well as the full dataset. Therefore researchers can evaluate their method based on a standard environment, and results across papers can be compared directly. This also helps to increase the reproducibility of research.

Future Work. Apart from the heterogeneity of networks, the dynamics of networks are also crucial to network representation learning. There are three ways to capture the dynamic information of networks. Firstly, we can add dynamic information into node attributes. For example, we can use methods like LSTM [14] to capture the dynamic activities of users. Secondly, the dynamic information, such as the timestamp of each interaction, can be considered as the attributes of edges. Thirdly, we may consider the several snapshots of networks representing the dynamic evolution of networks. We leave representation learning for the dynamic attributed multiplex heterogeneous network as our future work.

[13] https://github.com/HKUST-KnowComp/MNE
[14] https://github.com/cszhangzhen/ANRL