Learning Node Embeddings in Interaction Graphs

Yao Zhang
Fudan University
Shanghai, China
zhang_yao15@fudan.edu.cn

Yun Xiong
Fudan University
Shanghai, China
yunx@fudan.edu.cn

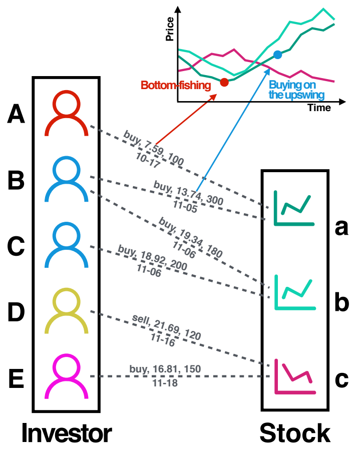
Xiangnan Kong
Worcester Polytechnic Institute
Worcester, MA, USA
xkong@wpi.edu

Yangyong Zhu
Fudan University
Shanghai, China
yyzhu@fudan.edu.cn

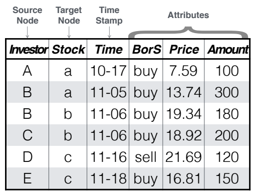
ABSTRACT

Node embedding techniques have gained prominence since they produce continuous and low-dimensional features, which are effective for various tasks. Most existing approaches learn node embeddings by exploring the structure of networks and are mainly focused on static non-attributed graphs. However, many real-world applications, such as stock markets and public review websites, involve bipartite graphs with dynamic and attributed edges, called attributed interaction graphs. Different from conventional graph data, attributed interaction graphs involve two kinds of entities (*e.g.* investors/stocks and users/businesses) and edges of temporal interactions with attributes (*e.g.* transactions and reviews). In this paper, we study the problem of node embedding in attributed interaction graphs. Learning embeddings in interaction graphs is highly challenging due to the dynamics and heterogeneous attributes of edges. Different from conventional static graphs, in attributed interaction graphs, each edge can have totally different meanings when the interaction is at different times or associated with different attributes. We propose a deep node embedding method called IGE (Interaction Graph Embedding). IGE is composed of three neural networks: an encoding network is proposed to transform attributes into a fixed-length vector to deal with the heterogeneity of attributes; then encoded attribute vectors interact with nodes multiplicatively in two coupled prediction networks that investigate the temporal dependency by treating incident edges of a node as the analogy of a sentence in word embedding methods. The encoding network can be specifically designed for different datasets as long as it is differentiable, in which case it can be trained together with prediction networks by back-propagation. We evaluate our proposed method and various comparing methods on four real-world datasets. The experimental results prove the effectiveness of the learned embeddings by IGE on both node clustering and classification tasks.

CIKM'17 , November 6–10, 2017, Singapore, Singapore
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-4918-5/17/11...$15.00
https://doi.org/10.1145/3132847.3132918

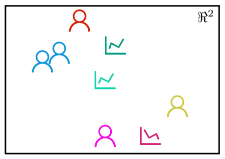


(a) Attributed Interaction Graph



Source Node	Target Node	Time Stamp	Attributes		
Investor	***Stock***	***Time***	***BorS***	***Price***	***Amount***
A	a	10-17	buy	7.59	100
B	a	11-05	buy	13.74	300
B	b	11-06	buy	19.34	180
C	b	11-06	buy	18.92	200
D	c	11-16	sell	21.69	120
E	c	11-18	buy	16.81	150

(b) Attributed Interactions (Edges)



(c) Node Embeddings in $\Re^2$

Figure 1: An example of attributed interaction graphs in stock market. (a) shows an attributed interaction graph and (b) shows its edges. Edges are dynamic and attributed. Investors with different colors have different investment strategies. (c) shows the embeddings of nodes, which can distinguish investors with different strategies and stocks with different trends.

CCS CONCEPTS

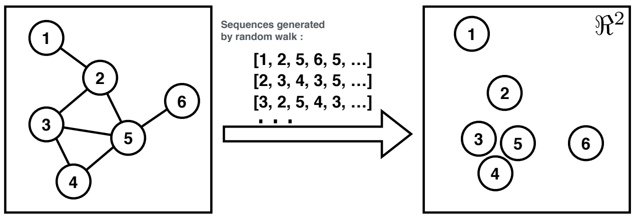
• **Computing methodologies** → **Learning latent representations**; • **Information systems** → Data mining;

KEYWORDS

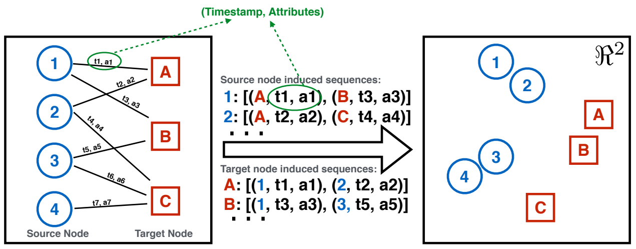
Graph Mining; Representation Learning; Attributed Interaction Graph; Node Embedding

1 INTRODUCTION

With the power of representing relations among entities, graphs are natural ways to organize data in many real-world applications. Examples include social networks and bibliographical networks. Graph analysis has been attracting much attention in recent years [5, 6, 10]. Most learning tasks on graphs, such as node classification or clustering, require a set of features of nodes. Node embedding



(a) Node Embedding in Static Graphs [8, 21]


(Timestamp, Attributes)
Source node induced sequences:
1: [(A, t1, a1), (B, t3, a3)]
2: [(A, t2, a2), (C, t4, a4)]
. . .
Target node induced sequences:
A: [(1, t1, a1), (2, t2, a2)]
B: [(1, t3, a3), (3, t5, a5)]
. . .
Source Node
Target Node
ℜ²

(b) Node Embedding in Attributed Interaction Graphs [This paper]

Figure 2: The comparison of node embedding problem in conventional graph data and in attributed interaction graphs. (a) Previous node embedding methods take a static homogeneous graph as input and simulate random walks starting from different nodes to generate sequences. Then a word embedding model is trained to get node embeddings. (b) Node embedding in attributed interaction graphs requires learning embeddings of source and target nodes simultaneously with a dynamic and attributed bipartite graph as input. IGE, our proposal, uses each node's sorted incident edges as *tagged* sequences. A source node's sequence only contains targets nodes, and vice versa.

problem has thus become popular in recent years. Many approaches, such as DeepWalk [21], LINE [25] and node2vec [8], have been proposed.

Previous node embedding methods mainly focus on exploring the structure of static graphs with non-attributed edges. However, many real-world applications involve data in the form of dynamic bipartite graphs with attributed edges. Here, we call them attributed interaction graphs. In attributed interaction graphs, an edge represents an interaction between two heterogeneous entities (nodes), and attributes of the edge represent the content of the interaction. For example, in the stock market, in order to profile different users' activities, we log each transaction in the form of *(investor, stock, time, buy_or_sell, price, amount)* as shown in Figures 1(a) and 1(b). In crowd-sourced review websites, like Yelp, users leave comments on business entities. Here, an attributed interaction is a tuple of *(user, business, time, comment)*, where the comment may contain photos in addition to a plain text review. Similarly in network security, we record interactions between programs and web servers for analysis. Here each interaction includes a set of attributes, such as program's name, target IP, connection type, *etc.*

In this paper, we study the problem of node embedding in attributed interaction graphs. The target is to learn the latent embeddings of nodes by considering the attributes and dynamics of the edges in the graph. Learning node embeddings in attributed interaction graphs is highly challenging and different from node embedding problems in conventional graph data:

• **Dynamic edges**: Unlike static graphs, each edge in interaction graphs represents a temporal event. An edge may have different meanings at different times. Consider that an investor tends to buy on the upswing while another investor likes bottom fishing, like investors A and B in Figure 1. They bought the same stocks, but at different times, which clearly reflects their different investment strategies.

• **Heterogeneous attributes on edges**: The attributes of an edge include different types of attributes: besides traditional categorical or numerical attributes, texts or images are also involved in some cases. Attributes interact closely with edges, and a unified model is needed to cope with these heterogeneous attributes. For example, in Figure 1, edges under different attributes may have opposite meanings, like buying and selling a stock. These phenomenons are hard to capture by only exploring the graph structure.

To learn meaningful embeddings for interaction graphs like Figure 1(c), we could investigate the temporal dependency of edges instead of the graph structure. Since nodes and edges can be seen as entities and interaction events respectively, all edges connecting with a same node are relevant events of the corresponding entity. Like the observation in word embeddings that words in the similar contexts might be similar, an entity's relevant events are likely to have some dependency. For instance, in stock dataset, edges of an investor can be treated as his transaction history. Determined by his investment strategy, the investor may short-sell a certain stock to hedge the risk after buying another stock. There may be more than one stock that can hedge the risk, and their embeddings should be similar. Stock entities are likely to be similar in similar "contexts", *i.e.*, previous and following transactions. From this point of view, an investor's sequence of events can be viewed as a "sentence" of transactions. Similarly, there are also "stock sentences", though they are not as meaningful as "investor sentences".

To tackle the above challenges, we propose a multiplicative neural model IGE (Interaction Graph Embedding) that takes an attributed interaction graph as input and produces latent embeddings of nodes (see Figure 1(c)). Our model is composed of two coupled prediction networks and an attribute encoding network. Similar to Skip-gram model [19, 20], given an edge, the objective of our prediction networks is to maximize classification of the node of another edge sampled from the same "sentence", *i.e.*, edges that share the same node. The edge is sampled according to the difference of timestamps of edges. A temporally closer edge is more likely to be selected. This solves the challenge of dynamic edges.

From Figure 2 we can see the difference in node embedding tasks in static graphs and attributed interaction graphs. We also can note that both our method and DeepWalk/node2vec use the idea of sequences of nodes, but they are quite different. Previous methods take a homogeneous graph as input and generate sequences by random walks, while our method is specialized in interaction graphs and directly uses the sequences of incident edges as *tagged* sequences.

To address the second challenge, we introduce an attributes encoding network. The encoding network can transform a tuple of attributes to a vector, and attribute vectors and node embeddings interact multiplicatively then.

The main contributions of our paper are as follows:

- We study the problem of learning node embeddings in attributed interaction graphs, and propose a multiplicative neural model IGE to learn embeddings.
- Our model consists of an attributes encoding network and two prediction networks. Three networks are trained simultaneously by back-propagation. Negative sampling method [20] is applied in training prediction networks for acceleration.
- We train IGE on four real-world datasets and evaluate the learned embeddings for both clustering and classification tasks. The experimental results show that the embeddings learned by IGE are effective and task-independent.

2 PROBLEM DEFINITION

In this section, we introduce some related concepts and notions, and then define the problem.

Definition 2.1 (Attributed Interaction Graph). An attributed interaction graph $G = (X, Y, E)$ is a special form of bipartite multigraph, where X and Y are two disjoint sets of nodes, and E is the set of edges. Each edge $e \in E$ can be represented as a tuple $e = (x, y, t, \mathbf{a})$, where $x \in X$, $y \in Y$, t denotes the timestamp and $\mathbf{a} = (a_1, \ldots, a_m)$ is the tuple of heterogeneous attributes of the edge. $a_i \in A_i$ is an instance of the i-th attribute.

To make the description clear, we call nodes in X source nodes and those in Y target nodes throughout the paper. An interaction, as the name implies, is a process by which two entities interact with each other. Though we name them source or target, these two types of nodes are in an equal position in interaction graphs.

Definition 2.2 (Induced Edge List). Given an attributed interaction graph $G = (X, Y, E)$ and a source node $x \in X$, we say $s_x = (e_1, \ldots, e_n)$ is an edge list induced by the source node x, if every edge in s_x is incident to x. Similarly, we can define a target node y's induced edge list s'_y.

In stock dataset, X and Y denote the set of investors and the set of stocks respectively, and edges denote transactions. An investor's induced list can be seen as his transaction history, and a stock's induced list is comprised of all transactions related to the stock. For example, in Figure 1(b), Investor B's list contains the second and third events in the table, which are his historical transactions in this interaction graph.

Like building inverted lists, it is easy to get all nodes' induced lists by one pass through all edges.

Definition 2.3 (Node Embedding in Attributed Interaction Graphs). Given an attributed interaction graph $G = (X, Y, E)$, the goal of node embedding is to learn mapping functions $\phi : X \longrightarrow \mathfrak{R}^K$ and $\psi : Y \longrightarrow \mathfrak{R}^{K'}$. $\phi(x)$ is the learned K-dimensional representation of node $x \in X$ and $\psi(y)$ is the K'-dimensional representation of node $y \in Y$.

3 PROPOSED METHOD

In this section we describe our proposed method for node embedding in attributed interaction graphs. We first consider a simplified problem to demonstrate the basic idea. Next we introduce a multiplicative neural model to learn node embeddings for interaction graphs.

Though we discuss on bipartite graphs, our proposal is also suitable for other types of graphs, *e.g.*, directed graphs, as long as we can distinguish between source nodes and target nodes of edges.

3.1 A Simplified Model

Let $G = (X, Y, E)$ be a given interaction graph where edges are without attributes. Inspired by the Skip-gram model [19, 20] and Paragraph Vector [15], we formulate the embedding learning as a maximum likelihood problem.

Since the graph can be partitioned into several induced lists, we let the log-likelihood be the sum of log-probability of induced lists:

$$\mathcal{L}(\theta) = \alpha \sum_{x \in X} \log P(s_x; \theta) + (1 - \alpha) \sum_{y \in Y} \log P(s'_y; \theta), \tag{1}$$

where s_x is an induced list of source node $x \in X$, s'_y is an induced list of target node $y \in Y$, $\alpha \in [0, 1]$ is a hyper-parameter, and θ denotes all the model parameters. α is used to make a trade-off between the importance of source nodes induced lists and target nodes induced lists. For instance, investors' lists seem more meaningful than stocks' lists, then we put more emphasis on investors' side by tuning α.

Note that the objective function Eq.1 contains two similar terms, so we alternatively optimize them during training, and we now only focus on the first term to describe the model.

Given an edge list $s_x = (e_1, \ldots, e_n)$ induced by some source node x, where $e_i = (x, y_i, t_i)$, we formulate the log-probability of s_x as follows:

$$\log P(s_x; \theta) = \sum_i \frac{1}{Z_i} \sum_{j \neq i} e^{\frac{-|t_i - t_j|}{\tau}} \log P(y_j | x, y_i; \theta), \tag{2}$$

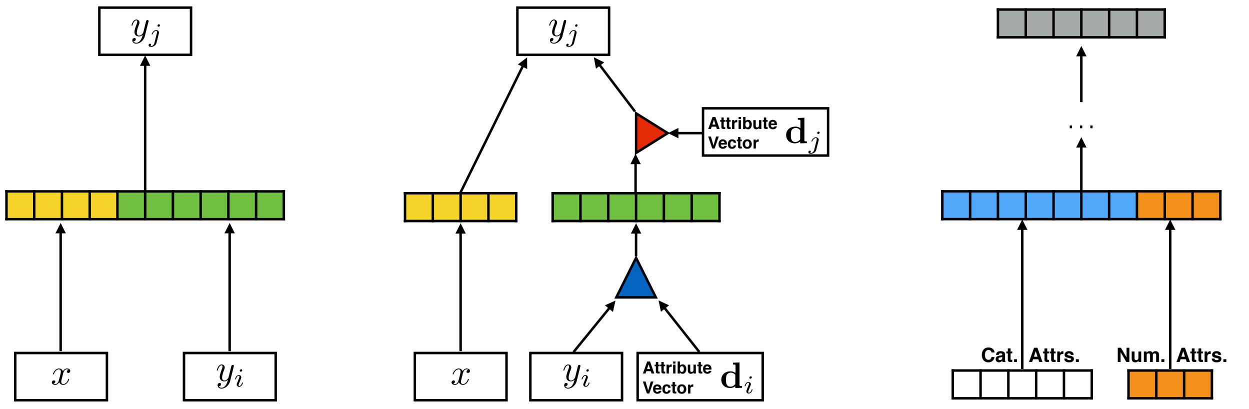
where $Z_i = \sum_{j \neq i} e^{\frac{-|t_i - t_j|}{\tau}}$ is a normalizing constant, and τ is a hyper-parameter. Now the log-probability of s_x becomes the weighted sum of log-probability of pairs of edges. If τ is small, weights will concentrate on the temporally close pairs. Conversely, the larger the τ is, the smoother the weights are. In this case, more long-term effects will be considered.

We use a neural network to represent $P(y_j | x, y_i; \theta)$. The structure is shown in Figure 3(a). We get the embeddings of x and y_i through embedding lookup tables $E_X \in \mathfrak{R}^{|X| \times K}$ and $E_Y \in \mathfrak{R}^{|Y| \times K'}$, and concatenate them as the input of softmax layer. The conditional probability of y_j being the k-th node is given by

$$\begin{aligned} & P(y_j = k | x, y_i; \theta) \\ = & \frac{\exp(U_X[k, :]^T \mathbf{v}_x + U_Y[k, :]^T \mathbf{v}_{y_i} + b)}{\sum_l \exp(U_X[l, :]^T \mathbf{v}_x + U_Y[l, :]^T \mathbf{v}_{y_i} + b)}, \end{aligned}$$

where $\mathbf{v}_x, \mathbf{v}_{y_i}$ are the embeddings of x and y_i, $U_X \in \mathfrak{R}^{|Y| \times K}$, $U_Y \in \mathfrak{R}^{|Y| \times K'}$ are weight matrices of softmax classifier, and $b \in \mathfrak{R}^{|Y|}$ is the bias term. $U[l, :]$ denotes the l-th row of matrix U.

Considering the size of Y, we apply negative sampling method [20] to training the softmax classifier efficiently and approximately.



y_j
x
y_i
y_j
Attribute Vector $\mathbf{d}_j$
x
y_i
Attribute Vector $\mathbf{d}_i$
...
Cat. Attrs.
Num. Attrs.

(a) Prediction network in the simplified model **(b) Prediction network in IGE** **(c) Attributes encoding network**

Figure 3: Network structure.

Other candidate sampling methods [9] are also possible. However, computing Eq.2 explicitly is still expensive. We instead compute

$$\log P(s_x;\theta) = \sum_i \frac{1}{|N(i)|} \sum_{j \in N(i)} \log P(y_j|x, y_i;\theta) \tag{3}$$

as an approximation, where $N(i) = \{i_1, \dots, i_c\}$ is the "context" of y_i. c is a pre-chosen hyper-parameter indicating the length of context, and i_k is selected randomly with the probability proportional to $e^{\frac{-|t_i - t_{i_k}|}{\tau}}$. $N(i)$ is a multiset, which means it allows duplicates of elements.

Notice that the network structure shown in Figure 3(a) is similar to PV-DM in [15] but there are some differences between the two models. In PV-DM, context is gained through a sliding window, and target word is predicted by its surrounding words with the help of paragraph vector. In our model, nodes are sampled and we use only one target node y_i to predict another target node y_j with the help of the source node x. Further, we have another similar network for second term in Eq.1 and use α to tune the weights of two networks.

Like what mentioned in [15], the embedding of x acts as a memory that remembers what is missing from the current context. In the stock dataset where x denotes an investor, an investor's embedding may contain information about his investment style, *e.g.*, active or passive style, that helps to predict the target stock.

3.2 Embedding Tensors

Now we discuss how to make use of attributes. In stock market, an investor may have different strategies when buying or selling a stock. It seems more sensible to have different embeddings under different situations, *i.e.*, conditional embeddings [13].

For this reason, we instead embed nodes as a tensor $\mathcal{T} \in \mathfrak{R}^{V \times K \times D}$, where V is the size of nodes set and D corresponds to the number of tensor slices. Given a tuple of attributes $\mathbf{a} = (a_1, \dots, a_m)$, and an attributes encoder f, we can get an attribute vector $\mathbf{d} = f(\mathbf{a}) \in \mathfrak{R}^D$. Then we can compute attributed-gated embeddings as $E^{\mathbf{d}} = \sum_{i=1}^{D} d_i \mathcal{T}[:,:,i]$, *i.e.*, a linear combination of slices weighted by each component d_i of $\mathbf{d}$. The details about encoder f is discussed in Section 3.4.

However, fully 3-way tensors are not practical because of enormous size. It is a common way [13, 24, 26] to factor the tensor by introducing three matrices $W^{fv} \in \mathfrak{R}^{F \times V}$, $W^{fd} \in \mathfrak{R}^{F \times D}$ and $W^{fk} \in \mathfrak{R}^{F \times K}$, and re-represent $E^{\mathbf{d}}$ by the equation

$$E^{\mathbf{d}} = (W^{fv})^T \cdot \operatorname{diag}(W^{fd}\mathbf{d}) \cdot W^{fk}, \tag{4}$$

where diag$(\cdot)$ denotes the matrix with its argument on the diagonal. These matrices are parametrized by a pre-chosen number of factors F. Eq.4 can be seen as the embeddings conditioned on $\mathbf{d}$, and we let

$$E = (W^{fv})^T W^{fk}$$

denote unconditional embeddings.

3.3 IGE: A Multiplicative Neural Model

We now show how to formulate $P(y_j|x, y_i, \mathbf{a}_i, \mathbf{a}_j;\theta)$, and substitute it for $P(y_j|x, y_i)$ term in Eq.2 to finish our IGE model. The structure is shown in Figure 3(b).

The key idea is still using the embeddings of x and y_i to predict y_j, but with the help of attributes $\mathbf{a}_i$ and $\mathbf{a}_j$.

At embedding layer, we have 6 matrices to learn: $W_X^{fv}, W_X^{fd}, W_X^{fk}$ and $W_Y^{fv}, W_Y^{fd}, W_Y^{fk}$. $\mathbf{v}_{y_i}^{\mathbf{d}_i}$, the embedding of y_i, is the corresponding row of attributed-gated embeddings $E_Y^{\mathbf{d}_i}$, where $\mathbf{d}_i = f(\mathbf{a}_i)$. Since $\mathbf{v}_x$, the embedding of x, is used as a memory and reflects the overall property, it should be independent of attributes. We use

$$E_X = (W_X^{fv})^T W_X^{fk}$$

as the unconditional matrix of embeddings of X.

At softmax layer, $\mathbf{v}_x$ makes its on decision since it acts as a memory, but $\mathbf{v}_{y_i}^{\mathbf{d}_i}$ considers the impact of $\mathbf{a}_j$. We use the same trick again: we introduce three matrices $U_Y^{ft} \in \mathfrak{R}^{F' \times |Y|}$, $U_Y^{fk} \in \mathfrak{R}^{F' \times K'}$ and $U_Y^{fd} \in \mathfrak{R}^{F' \times D}$, and use

$$U_Y^{\mathbf{d}_j} = (U_Y^{ft})^T \cdot \operatorname{diag}(U_Y^{fd}\mathbf{d}_j) \cdot U_Y^{fk}$$

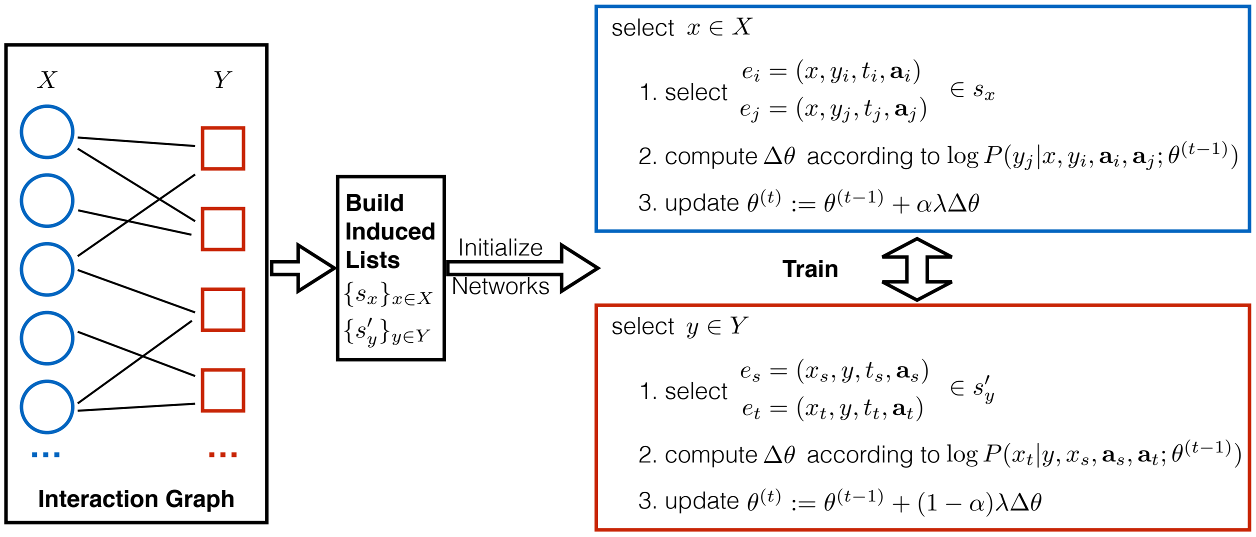

X
Y
Interaction Graph
Build Induced Lists
$\{s_x\}_{x\in X}$
$\{s'_y\}_{y\in Y}$
Initialize Networks
select $x \in X$
1. select $e_i = (x, y_i, t_i, \mathbf{a}_i)$, $e_j = (x, y_j, t_j, \mathbf{a}_j)$ $\in s_x$
2. compute $\Delta\theta$ according to $\log P(y_j|x, y_i, \mathbf{a}_i, \mathbf{a}_j; \theta^{(t-1)})$
3. update $\theta^{(t)} := \theta^{(t-1)} + \alpha\lambda\Delta\theta$
Train
select $y \in Y$
1. select $e_s = (x_s, y, t_s, \mathbf{a}_s)$, $e_t = (x_t, y, t_t, \mathbf{a}_t)$ $\in s'_y$
2. compute $\Delta\theta$ according to $\log P(x_t|y, x_s, \mathbf{a}_s, \mathbf{a}_t; \theta^{(t-1)})$
3. update $\theta^{(t)} := \theta^{(t-1)} + (1-\alpha)\lambda\Delta\theta$

Figure 4: The pipeline of IGE. IGE takes an interaction graph as the input and builds all induced lists by one pass through the data. After initialization of networks, we train the coupled prediction networks alternatively. The encoding network also gets trained by back-propagation. In Step 1, edges are selected according to Eq. 3, and the expression of $\log P(\cdot)$ in Step 2 is Eq. 5. $\Delta\theta$ is the direction of decrease and λ is the step size.

as the weight matrix of softmax classifier with respect to y_i. Now the probability can be computed as follows:

$$P(y_j = k|x, y_i, \mathbf{a}_i, \mathbf{a}_j; \theta) = \frac{\exp(U_X[k,:]^T \mathbf{v}_x + U_Y^{\mathbf{d}_j}[k,:]^T \mathbf{v}_{y_i}^{\mathbf{d}_i} + b)}{\sum_l \exp(U_X[l,:]^T \mathbf{v}_x + U_Y^{\mathbf{d}_j}[l,:]^T \mathbf{v}_{y_i}^{\mathbf{d}_i} + b)}, \qquad (5)$$

where $U_X \in \Re^{|Y|\times K}$ is the weight matrix with respect to x, and $b \in \Re^{|Y|}$ is the bias term.

Recall that there are two terms in the objective function Eq.1, so there are two corresponding prediction networks. The network of second term uses x_s and y to predict x_t instead. We need to clarify that the weights at embedding layers are shared across two networks, and we use unconditional embeddings as the final embeddings.

3.4 Attributes Encoding Network

For consistency, we use another neural network to represent attributes encoder. The encoding network can be specifically designed for different datasets as long as it is differentiable, so that it can be trained together with prediction networks by back-propagation. For example, if attributes are sentences like reviews in Yelp, a recurrent neural network (RNN) is applicable. Or we could use a convolutional neural network (CNN) to encode images.

Here we illustrate the encoding network for mixes of categorical and numerical attributes. The structure is shown in Figure 3(c). Each categorical attribute in attributes tuple $\mathbf{a} = (a_1, \ldots, a_m)$ has its own embedding lookup table. Embeddings of all categorical attributes are averaged or concatenated. Then the merged embeddings and remaining numerical attributes are concatenated comprising the input of the next layer. The remaining part of network is a simple fully connected network. We choose rectified linear unit (ReLU) [7] as the activation function to ensure sparseness and positiveness.

Though multiplicative units enhance the power of representations, the product of parameters makes gradient descent learning difficult [24]. So here we prefer a relatively shallow network to weaken the effect of this phenomenon.

We train two prediction networks alternatively using Adam optimizer [11]. During training, all weights in the encoding network also get trained by back-propagation. We summarize the training procedure in Figure 4.

4 EXPERIMENTS

In this section we provide an overview of the datasets and methods that we use in our experiments, and then show the results.

4.1 Datasets

In order to evaluate the effectiveness, we test methods on four real-world datasets. An overview of the datasets is shown in Table 1.

- DBLP: The first dataset is a bibliographic network extracted from DBLP database[1]. Authors and conferences are treated as nodes, and edges represent publications. We use the corresponding paper's id as a categorical attribute and the bag-of-words representation of title as a set of numerical attributes of an edge. 2125 authors are labeled with one of the four domains[2].
- PPD: This dataset contains investing records in a month extracted from a P2P lending platform[3]. Investors and loans

[1]http://dblp.uni-trier.de
[2]Data mining, machine learning, database, and information retrieval.
[3]http://www.ppdai.com

Table 1: Overview of datasets. (Entities in bold indicates the labeled entities.)

	DBLP	PPD	Stock	Yelp
Source Entity	**Author**(35851)	Investor(9292)	Investor(22001)	User(724884)
Target Entity	Conference(20)	**Loan**(4501)	**Stock**(939)	**Business**(15726)
# Edges	104325	223190	66890	2465173
# Cat. Attrs.	1	0	1	Various
# Num. Attrs.	698	6	3	0

Table 2: Summary of comparing methods.

	Structure of graph	Temporal dependency	Categorical Attributes	Numerical Attributes
PV-DM [15]		✓		
node2vec [8]	✓			
APE [3]			✓	
bag-of-words [this paper]	✓		✓	✓
IGE w/o attrs. [this paper]		✓		
IGE [this paper]		✓	✓	✓

are nodes and each edge represents an investment. Loans are partitioned into 6 groups according to their risk level.

- Stock: This dataset contains transaction records in two weeks in Chinese stock market. Nodes in this dataset are investors and stocks, and each edge represents a transaction. A example of this dataset has been shown in Figure 1. We treat *buy_or_sell* as a categorical attribute. 500 stocks are partitioned into 3 groups based on their Beta, a measure of systematic risk.
- Yelp: This dataset is a subset of Yelp[4] Challenge Dataset. We only extract top-10 categories[5] related businesses. Users and businesses are nodes in this dataset, and comments are edges. Instead of bag-of-words representations that we use in DBLP dataset, we use raw texts as various-length tuples of categorical attributes. Every business is assigned one or more categories.

4.2 Comparing Methods

We compare the following methods in the experiments and summarize them in Table 2:

- PV-DM [15]: This method is designed for learning representations of variable-length pieces of texts, *i.e.*, sentences, paragraphs and documents. We test this method because our method implicitly generates "sentences", *i.e.*, node induced lists, but PV-DM is unable to cope with attributes. In our experiments, "sentences" are sorted by time and we treat labeled nodes as documents, and nodes of another type as words.
- node2vec [8]: This method is one of the baseline methods for learning nodes embeddings in networks. It explores networks by a flexible biased random walk procedure.

[4]https://www.yelp.com

[5]Restaurants, Shopping, Food, Beauty & Spas, Home Services, Nightlife, Health & Medical, Bars, Automotive, Local Services

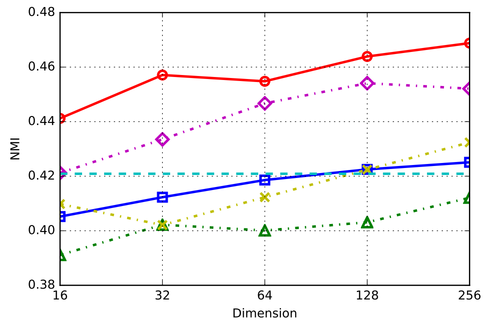
- APE [3]: This method is designed for detecting anomalous events. Given a set of heterogeneous categorical events, *i.e.*, events with different categorical attributes/entities, it tries to maximize the likelihood of the data by embedding different entities into a common latent space and then assessing the compatibility of entities. An interaction graph can be represented as a list of edges, which is equivalent to a sequence of events, and thus this method is able to learn node embeddings in interaction graphs.
- bag-of-words: This is a naive method. We first use one-hot vectors to encode nodes and categorical attributes of edges. A node representation is given by the concatenation of three parts: averaged vectors of adjacent nodes, averaged vectors of categorical attributes of incident edges and averaged remaining numerical attributes of incident edges.
- IGE w/o attrs.: This method is the simplified version of IGE described in Section 3.1, which takes an interaction graph as input and outputs embeddings of nodes. Note that this method ignores the attributes of edges.
- IGE: This is our proposal described in Section 3.3, which can learn node embeddings in interaction graphs.

Unless otherwise stated, we use the following settings of our proposals throughout the experiments: $\alpha = 0.5$, the number of factors $F = F' = 1024$, and the dimension of encoded attribute vector $D = 8$. Further, attributes encoding network is without hidden layers and embedding dimensions K and K' are same.

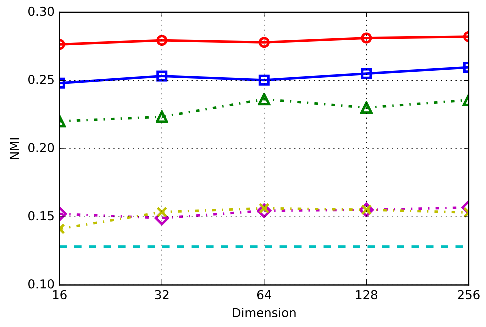
Though we have considered timestamp when sampling "context", we use timestamp again as an additional numerical attribute. We find that doing such can slightly improve the performance.

4.3 Experimental Results

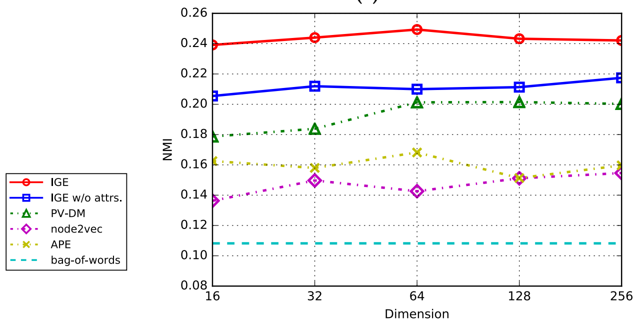
Since all comparing methods are task-independent, we test the learned embeddings on both classification and clustering tasks. We show the averaged result of 10 runs for each experiment.



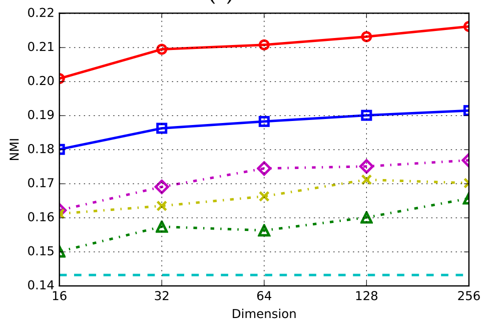
(a) DBLP



(b) PPD

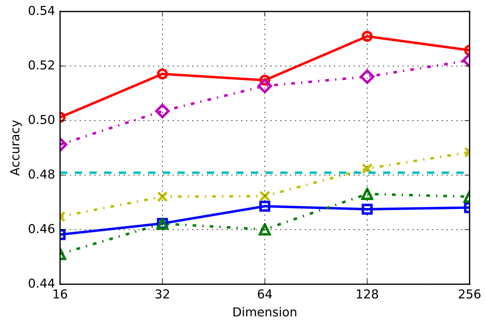


(c) Stock

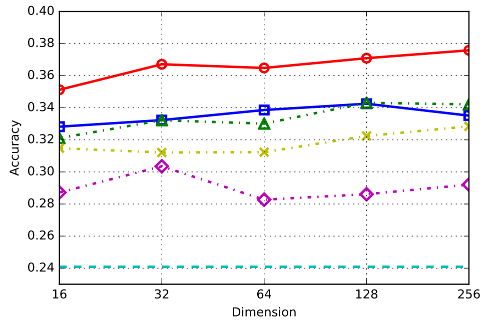


(d) Yelp

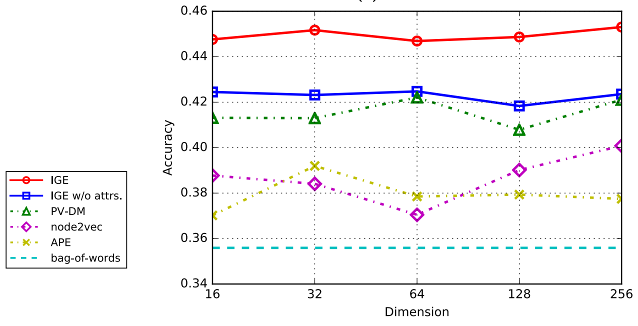
Figure 5: Experimental results on clustering.



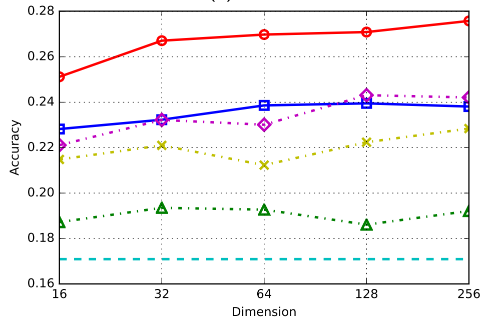
(a) DBLP



(b) PPD

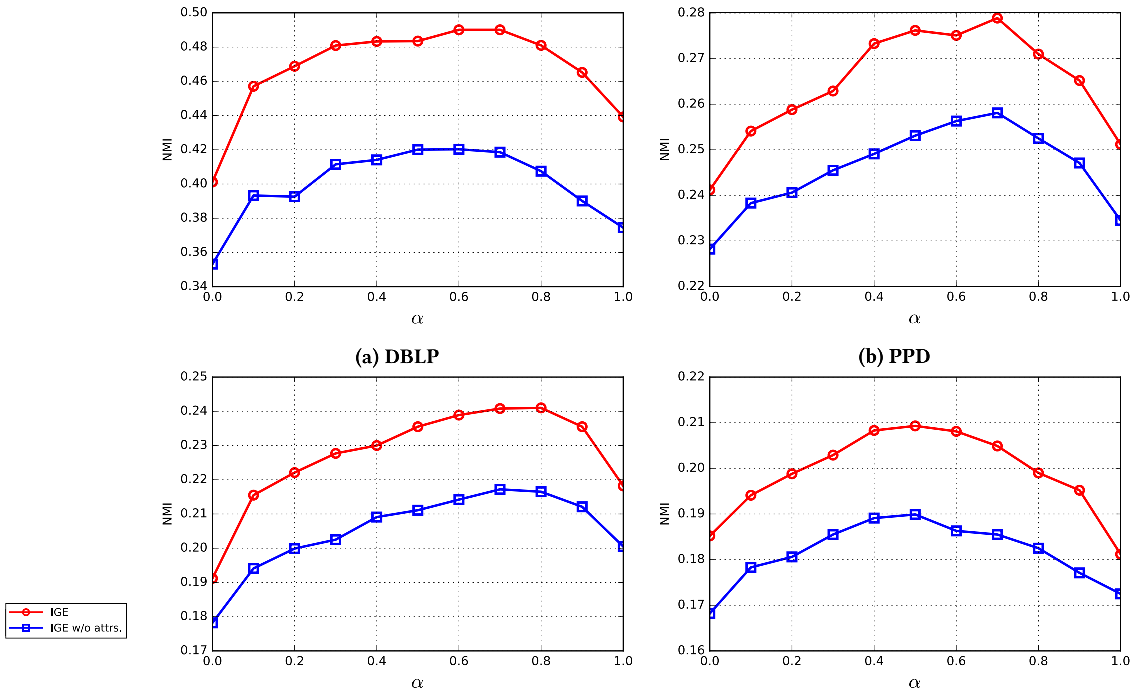


(c) Stock



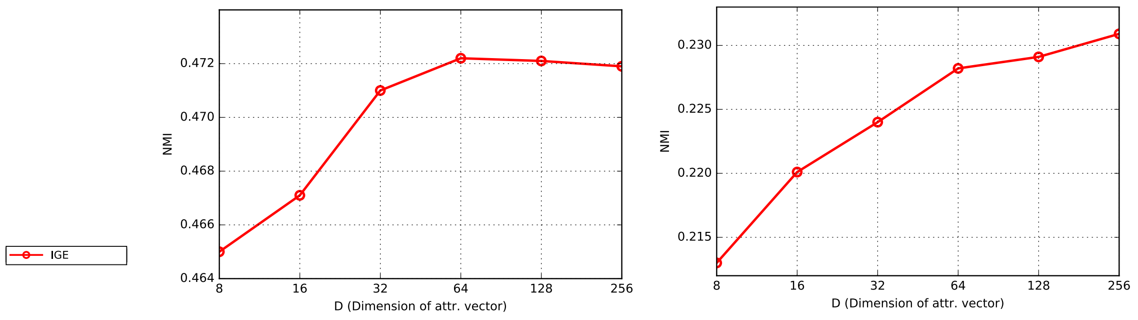
(d) Yelp

Figure 6: Experimental results on classification.



(a) DBLP (b) PPD (c) Stock (d) Yelp

Figure 7: Parameter sensitivity w.r.t α.



(a) DBLP (b) Yelp

Figure 8: Parameter sensitivity w.r.t D.

4.3.1 Clustering. We first use K-Means to test embeddings on the unsupervised task. We use Normalized Mutual Information (NMI) [23] score to evaluate clustering results. The NMI score is between 0 and 1. The larger the value, the better the performance. A labeling will have score 1 if it matches the ground truth perfectly, and 0 if it is completely random. Since entities in the Yelp dataset are multi-labeled, we ignore the entities that belong to multiple categories when calculate NMI score. The experimental results are shown in Figure 5.

4.3.2 Classification. We also consider the classification task. We perform 10-fold cross-validation and train logistic regression models with one-vs-rest scheme. The metric we adopt here is accuracy score. The larger the value, the better the performance. For consistency, we still use accuracy score for Yelp dataset: if the predicted category belong to the set of true categories, we treat it as a successful prediction. The experimental results are shown in Figure 6.

4.3.3 Analysis. From Figures 5 and 6 we can see, IGE outperforms the baseline methods consistently with increasing dimensions. The performance on DBLP and Yelp goes better as the embedding dimension increases, but on the other two datasets, the performance is relatively stable. We use straight lines to show the results of bag-of-words since it can only produce fixed-length representations. By comparing IGE and IGE w/o attrs., we can find IGE indeed improves the quality of embeddings by using attributes.

It is also noticeable that node2vec and bag-of-words perform well in DBLP but poorly in other three datasets. This is because there are only 20 conferences in DBLP, and they are highly correlated with labels. Thus by exploring structure, node2vec and bag-of-words produce sensible embeddings on DBLP dataset.

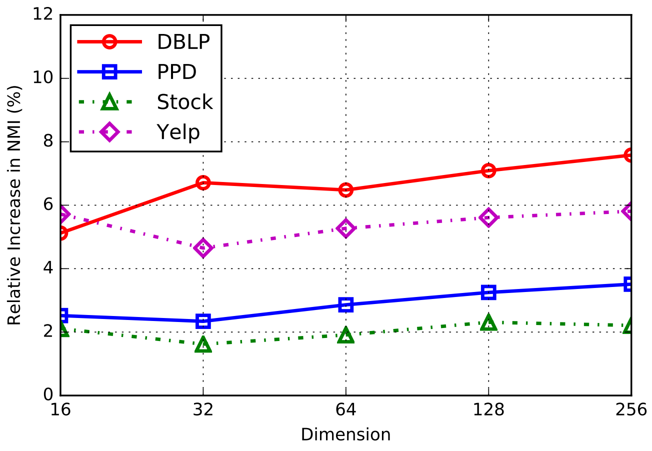


Figure 9: Results on clustering using concatenation of embeddings.

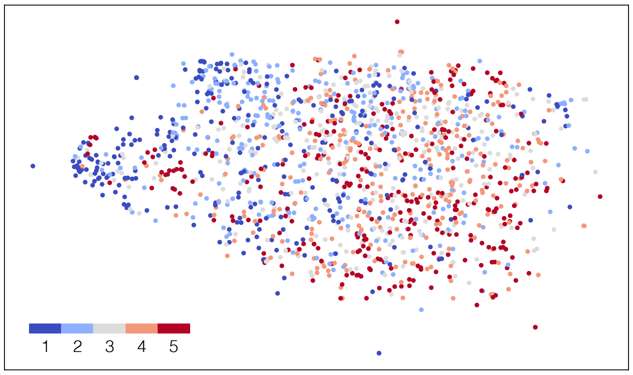


Figure 10: Visualization of attribute vectors of Yelp dataset.

4.3.4 Parameter Sensitivity. In this section, we only show the results on clustering since the results on classification show the same trends. We let the embedding dimension K be 128 in this section. We first evaluate the performance with respect to α in Eq.1. As we can see in Figure 7, α does adjust the importances of two prediction networks. The best α's are all above 0.5 in the first three datasets. Recall that the source nodes in these datasets both are humans (authors and investors), so we could say people's behavior is more informative for learning embeddings in attributed interaction graphs.

We also test the effect of D, the dimension of encoded attribute vector. In the previous experiments, we use low-dimensional attribute vectors because the number of attributes of edges in PPD and stock datasets is small. Now we exclude PPD and stock datasets. The results on DBLP and Yelp shown in Figure 8 demonstrate that increasing the dimension of attribute vector can slightly enhance the effectiveness of embeddings.

4.3.5 Concatenation of Embeddings. As listed in Table 2, IGE does not use structure information at all. Here, we concatenate the embeddings produced by node2vec and IGE, and train a K-Means model again. Results (Figure 9) show that the concatenation improves the NMI score. This means node2vec and IGE capture information from different perspectives and simply concatenating them brings better result.

4.3.6 Visualization of Encoded Attributes. In this part, we show the effectiveness of the encoding network. As we described, each edge can have totally different meanings when the interaction is associated with different attributes, so an effective encoding network should be able to distinguish between patterns of attributes.

Edges in Yelp dataset represent comments and are labeled: every comment is accompanied by a score ranging from 1 to 5, which reflects the users' attitude towards the business. To test whether the encoding network could capture this sentimental information, we randomly select 300 samples for each score, and then use t-SNE [17] to map their learned vectors into a 2-dimensional plane. Though it does not show clear boundaries, from the figure we find that the left part of plane are populated mainly by low-score reviews, while high-score reviews are concentrated in the right part. This means the encoding network does learn some patterns of attributes.

5 RELATED WORK

Node embeddings have been widely studied in the literature. Classical methods usually need to construct the affinity graph and solve eigenvectors, such as multidimensional scaling [4], Laplacian Eigenmap [1], IsoMap [27], LLE [22]. These methods are not scalable for large networks. There are some efficient methods learning embeddings by exploring the structure of networks. DeepWalk [21] and node2vec [8] first generate "sentences" of nodes by a random walk procedure, then feed these sentences to word embedding models like word2vec [19, 20] to get embeddings of nodes. LINE [25] make use of structure information in a different way: it defines the first-order and second-order proximities in networks, and directly optimizes objective functions that try to preserve the proximities. However, these methods are only suitable for static graphs without attributes. Each edge can have totally different meanings when the interaction is at different time or associated with different attributes. Ignoring these effects results in a huge information loss.

Like multimodal versions [12, 14, 18] of word embedding models, TADW [29] incorporates text features of nodes into embedding learning. TADW or multimodal DeepWalk/node2vec cannot directly handle attributed interaction graphs either, because the basic atom is node here, but attributes are on edges. TransE [2], TransH [28] and TransR [16] are relational data embedding methods that take triples *(head_entity, relation, tail_entity)* as inputs. Though an edge in interaction graphs can be treated as a complicated triple, the space of relations are exponentially large even infinite due to heterogeneous attributes of edges, which makes these method inapplicable to interaction graphs. These methods do not consider dynamics of edges either.

6 CONCLUSION

In this paper, we generalize embedding techniques to attributed interaction graphs and propose IGE. Different from previous work, IGE investigates the temporal dependency of edges instead of the structure of graphs. IGE contains two coupled multiplicative neural networks for prediction and an attributes encoding network. Experimental results on various real-world datasets prove the effectiveness of the learned embeddings by IGE on both clustering and classification tasks.

As for future work, we will consider applying the model to edge embeddings. Besides, our proposal does not use structure information at all, and we will try to incorporate this into the model. Our proposal maps two sets of nodes to two different latent spaces. It is also a potential work to embed nodes in the same space.

ACKNOWLEDGMENT

This work is supported in part by the National Natural Science Foundation of China Projects No.91546105, No.U1636207, NSF through Grant IIS-1718310, the National High Technology Research and Development Program of China No.2015AA020105, the Shanghai Science and Technology Development Fund No. 16JC1400801, No.17511105502, No.16511102204, Special Program for Applied Research on Super Computation of the NSFC-Guangdong Joint Fund (the second phase) under Grant No.U1501501.

REFERENCES

[1] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems*, pages 585–591, 2001.

[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795, 2013.

[3] Ting Chen, Lu-An Tang, Yizhou Sun, Zhengzhang Chen, and Kai Zhang. Entity embedding-based anomaly detection for heterogeneous categorical events. *arXiv preprint arXiv:1608.07502*, 2016.

[4] Trevor F Cox and Michael AA Cox. *Multidimensional scaling*. CRC press, 2000.

[5] Yuxiao Dong, Jing Zhang, Jie Tang, Nitesh V Chawla, and Bai Wang. Coupledlp: Link prediction in coupled networks. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 199–208, 2015.

[6] Nadia Ghamrawi and Andrew McCallum. Collective multi-label classification. In *ACM International Conference on Information and Knowledge Management*, pages 195–200, 2005.

[7] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

[8] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 855–864, 2016.

[9] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. In *Annual Meeting of the Association for Computational Linguistics*, pages 1–10, 2015.

[10] Zhao Kang, Chong Peng, Ming Yang, and Qiang Cheng. Top-n recommendation on graphs. In *ACM International Conference on Information and Knowledge Management*, pages 2101–2106, 2016.

[11] Diederik Kingma and Jimmy Ba. ADAM: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] Ryan Kiros, Ruslan Salakhutdinov, and Rich Zemel. Multimodal neural language models. In *International Conference on Machine Learning*, pages 595–603, 2014.

[13] Ryan Kiros, Richard Zemel, and Ruslan R Salakhutdinov. A multiplicative model for learning distributed text-based attribute representations. In *Advances in Neural Information Processing Systems*, pages 2348–2356, 2014.

[14] Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. Combining language and vision with a multimodal skip-gram model. *arXiv preprint arXiv:1501.02598*, 2015.

[15] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.

[16] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI Conference on Artificial Intelligence*, pages 2181–2187, 2015.

[17] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[18] Junhua Mao, Jiajing Xu, Kevin Jing, and Alan L Yuille. Training and evaluating multimodal word embeddings with large-scale web annotated images. In *Advances in Neural Information Processing Systems*, pages 442–450, 2016.

[19] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[21] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 701–710, 2014.

[22] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[23] Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3(Dec):583–617, 2002.

[24] Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *International Conference on Machine Learning*, pages 1017–1024, 2011.

[25] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *International World Wide Web Conference*, pages 1067–1077, 2015.

[26] Graham W Taylor and Geoffrey E Hinton. Factored conditional restricted boltzmann machines for modeling motion style. In *International Conference on Machine Learning*, pages 1025–1032, 2009.

[27] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[28] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI Conference on Artificial Intelligence*, pages 1112–1119, 2014.

[29] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. Network representation learning with rich text information. In *International Joint Conference on Artificial Intelligence*, pages 2111–2117, 2015.