HIN2Vec: Explore Meta-paths in Heterogeneous Information Networks for Representation Learning

Tao-yang Fu
The Pennsylvania State University
University Park, PA 16802, USA
txf225@cse.psu.edu

Wang-Chien Lee
The Pennsylvania State University
University Park, PA 16802, USA
wlee@cse.psu.edu

Zhen Lei
The Pennsylvania State University
University Park, PA 16802, USA
zlei@psu.edu

ABSTRACT

In this paper, we propose a novel representation learning framework, namely *HIN2Vec*, for heterogeneous information networks (HINs). The core of the proposed framework is a neural network model, also called HIN2Vec, designed to capture the rich semantics embedded in HINs by exploiting different types of relationships among nodes. Given a set of relationships specified in forms of meta-paths in an HIN, HIN2Vec carries out multiple prediction training tasks jointly based on a target set of relationships to learn latent vectors of nodes and meta-paths in the HIN. In addition to model design, several issues unique to HIN2Vec, including regularization of meta-path vectors, node type selection in negative sampling, and cycles in random walks, are examined. To validate our ideas, we learn latent vectors of nodes using four large-scale real HIN datasets, including Blogcatalog, Yelp, DBLP and U.S. Patents, and use them as features for multi-label node classification and link prediction applications on those networks. Empirical results show that HIN2Vec soundly outperforms the state-of-the-art representation learning models for network data, including DeepWalk, LINE, node2vec, PTE, HINE and ESim, by 6.6% to 23.8% of *micro-f_1* in multi-label node classification and 5% to 70.8% of *MAP* in link prediction.

KEYWORDS

representation learning, heterogeneous information network

1 INTRODUCTION

Network data analysis and mining is an important research field because network data, capturing phenomena in various networks, such as social networks, paper citation networks, and World Wide Web, are ubiquitous in the real world [6, 15, 29]. Network analysis often involves prediction tasks over nodes or edges, e.g., node classification [14], node clustering [23] and link prediction [20]. In order to achieve good performance in these tasks, a proper representation of network nodes and edges that captures embedded information in the network structure while preserving the original relationships amongst nodes is much needed to serve as input features to supervised machine learning algorithms. This is a preprocessing step for data mining and knowledge discovery, well known as *feature engineering*. A typical approach of feature engineering is to involve domain experts to manually design domain-specific representation of data, i.e., feature vectors of data, for specific prediction tasks. This approach, heavily relying on prior knowledge and experiences of domain experts, is time-consuming and expensive. This issue has given rise to a great deal of interest in representation learning in networks that aims to embed a network into a low-dimensional space and *represent each node as a low-dimensional feature vector* for supervised learning.

In this paper, we propose a new neural network (NN) model, namely *Heterogeneous Information Network to Vector (HIN2Vec)* for representation learning of nodes in heterogeneous information networks (HINs). The HIN2Vec model aims to capture the rich information in an HIN by exploiting various types of relationships among nodes and the network structure. HINs, such as Yelp social network [4], DBLP collaboratoin network [3], and U.S. patent citation network [2], are networks with nodes and edges belonging to different types. With heterogeneous types of nodes and edges, HINs are able to describe various types of relationships among nodes and thus contain very rich information. A *meta-path*, consisted of a sequence of node types and/or edge types, is usually used to denote a particular relationship between node pairs. Therefore, different meta-paths may have different semantics. For example, consider a DBLP collaboration network, which consists of three node types: Author, Paper and Venue, and two edge types: an author *writes* a paper, and a paper *is published* in a venue. A meta-path Author-Paper-Author describes the collaboration between two authors, while Author-Paper-Venue-Paper-Author describes the relationship where both authors have papers published in the same conference venue. We claim that encoding the rich information embedded in meta-paths and the whole network structure would help learning *meaningful* representation which is useful for various applications, because the different semantics of relationships are better captured.

To achieve this goal and to train the HIN2Vec model, we design a new learning framework (also called HIN2Vec) (shown in Figure 1) which, given an HIN and a set of targeted relationships specified in forms of meta-paths, learns latent vectors of both nodes and the targeted relationships in the HIN by predicting relationships between nodes. Compared with previous works, the HIN2Vec model preserves more contextual information, not only assuming that two nodes are relevant if there exists a relationship between them

ISBN 978-1-4503-4918-5/17/11...$15.00
DOI: https://doi.org/10.1145/3132847.3132953

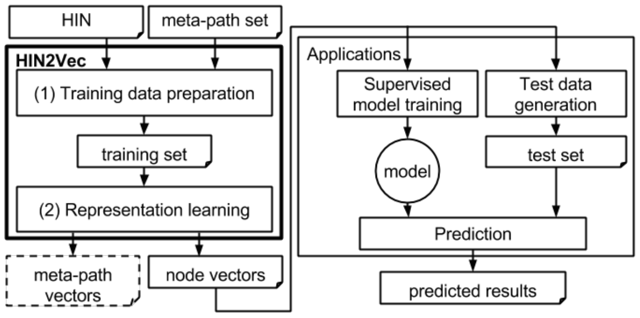


Figure 1: Overview of the HIN2Vec framework

(as previous works do) but also distinguishing the different relationships between nodes and treating them differently by learning relationship vectors jointly. In specific, the HIN2Vec framework consists of two phases: (1) *Training data preparation:* A data preparation approach based on *random walk* and *negative sampling* is developed to prepare training data in accordance with the targeted relationships for representation learning of nodes in HINs; and (2) *Representation learning:* The novel neural network model, HIN2Vec model, is designed to learn both node vectors and relationship vectors by maximizing the likelihood of predicting relationships among nodes jointly. The proposed neural network model is trained by predicting *multiple* heterogeneous relationships between node pairs simultaneously and *jointly*. This *multi-task learning* approach allows the proposed model to *jointly embed the rich information of different relationships and the overall network structure into node vectors*. Conceptually, relevant nodes which have relationships are close to each other. On the other hand, the relationship vectors indicate which dimension captures certain relationships, and are useful for providing analytical insights, such as grouping meta-paths with similar semantics. Furthermore, it is essential for the proposed model to be scalable for large-scale real-world datasets. We leverage asynchronous stochastic gradient descent in representation learning in parallel.

There are a few previous works on representation learning in homogeneous information networks [10, 24, 28]. Although these prior works all claim that their approaches are able to capture the structural information of networks, the specific objective functions used in their models tend to consider only part of aggregated information among nodes or limited types of relationships between nodes separately. There are also several existing works on representation learning in HINs [8, 11, 13, 25, 27]. However, some models aim to only capture limited types of relationships between nodes (e.g., one-hop or two-hop neighborhood between two nodes) [8, 11, 27], and some tend to miss the different semantics of relationships between nodes and only capture the aggregated information of relationships [8, 11, 13, 27]. Only one study [25] tries to capture different relationships between nodes. However, it highly depends on a user-guided way to determine a user-given meta-path set and the weight of each meta-path for representation learning. Moreover, some parts of its objective function to encode relationships between nodes, e.g., the multiplication of vectors of meta-paths, are illy defined.

The major contributions of our work are summarized as follows.

A novel idea for representation learning in HINs. With multiple types of nodes and edges, HINs are able to describe various types of relationships between nodes, which have different semantics. We show that by capturing various types of relationships between nodes would help representation learning because it better captures more detailed and precise information embedded in the network structure.

A new representation learning framework for HINs. We propose a two-phase framework to learn representations of nodes and meta-paths in HINs. The training data preparation algorithm in Phase 1 adopts random walk generation and negative sampling to prepare training data tailored for HIN2Vec. The HIN2Vec NN model, core of phase 2, is designed as a *logistic binary classifier* that predicts whether two input nodes has a specific relationship in order to efficiently learn model parameters, i.e., node vectors and meta-path vectors. Issues in these two phases, such as cycles in random walks, node selection in negative sampling, and regularization of meta-path vectors are studied.

Empirical study using real-world data. We evaluate HIN2Vec by conducting a comprehensive evaluation with two different applications, node classification and link prediction, using four large-scale real HIN datasets, including Blogcatalog, Yelp, DBLP and U.S. patents, in comparison with six state-of-the-art representation learning models. Empirical result shows that HIN2Vec soundly outperforms all existing models. In addition, an analysis on the learned meta-path vectors shows that the learned representation of relationships capture their semantics.

In the rest of this paper, we first review the related work in Section 2, and introduce HINs, meta-paths, problem definition and analysis in Section 3. We present the proposed HIN2Vec framework in Section 4 and show experiment results in Section 5. Finally, we conclude the paper in Section 6.

2 RELATED WORK

Recent development on representation learning has shed a light on alleviating the dependence of feature engineering on human knowledge and labors [7, 24, 28]. The goal of representation learning aims to automatically learn useful latent representations of data that are effective and discriminative as input features to supervised machine learning algorithms for various prediction tasks. Among the various approaches of representation learning, the neural network based learning models have received significant attention in recent years, and achieved successes in several empirical research studies of various domains, including speech recognition [12, 22], computer vision [9, 16], and natural language processing (NLP) [21].

Recently, research on representation learning has been extended to network data [8, 10, 11, 13, 24, 25, 27, 28]. However, instead of the complex Heterogeneous information networks (HINs) targeted in this paper, some prior works focus only on learning node vectors in homogeneous information networks [10, 24, 28]. Moreover, while they all claim that their approaches are able to capture the embedded structures of information networks, these models tend to consider only aggregated information among nodes or limited types of relationships. For instance, DeepWalk [24] and node2vec

[10] learn feature vectors of nodes by capturing the nearby neighborhood to each node by simulating uniform and parameterized random walks, respectively. LINE [28] captures 1-hop and 2-hop neighborhood relationships, separately, to learn two representations of nodes.

There are also several existing works on representation learning on HINs [8, 11, 13, 25, 27]. Some models aim to only capture *limited types* of relationships between nodes. Specifically, PTE [27] and HNE [8] learn feature vectors of nodes by capturing 1-hop neighborhood relationships between nodes. HEBE [11] captures 2-hop neighborhood between multiple nodes. Some tend to miss the different semantics of relationships between nodes and only capture the aggregated information of relationships [8, 11, 13, 27]. Only one study [25] tries to capture different relationships between nodes. However, it relies heavily on *user guidance* to determine a user-given meta-path set and the weight of each meta-path for representation learning. Moreover, some parts of its objective function to encode relationships between nodes, e.g., the multiplication of vectors of meta-paths, are illy defined.

3 PRELIMINARIES

In this section, we first introduce the notions of HINs and meta-paths, then define the tackled problem, and discuss the challenges.

3.1 Information Network and Meta-Path

We first define information networks and meta-paths.

Definition 1. *Information Network. An information network is a directed graph $G = (V, E, \Phi, \Psi)$, where V is the set of nodes; $E \subseteq V \times V$ is the set of edges in V. $\Phi : V \to A$ and $\Psi : E \to R$ are type mapping functions for nodes and edges, respectively. Here each node $v \in V$ is mapped to one particular node type in A, i.e., $\Phi(v) \in A$, and each link $e \in E$ belongs to a particular edge type in R, i.e., $\Psi(e) \in R$. When $|A| > 1$ or $|R| > 1$, the network is called a heterogeneous information network (HIN); otherwise, it is a homogeneous information network.*

The concept of meta-paths has been proposed to describe how nodes are related to each other [18] [26]. In the following, we define meta-paths.

Definition 2. *Meta-path. Given a heterogeneous information network $G = (V, E, \Phi, \Psi)$, a meta-path π is a sequence of node types $a_1, a_2, ..., a_n$ and/or edge types $r_1, r_2, ..., r_{n-1}$:*

$$\pi = a_1 \xrightarrow{r_1} ...a_i \xrightarrow{r_i} ... \xrightarrow{r_{n-1}} a_n$$

A path p, which goes though nodes $v_1, v_2, ..., v_n$, is an instance of the meta-path π, if $\forall i = 1, ..., n, a_i = \Phi(v_i)$ and $r_i = \Psi(v_i, v_{i+1})$.

3.2 Problem Definition and Analysis

The goal of this work is to learn a representation of nodes in an HIN. In the following, we formally define the problem.

Definition 3. *Representation Learning on HIN. Given a heterogeneous information network (HIN), denoted as a graph $G = (V, E, \Phi, \Psi)$. Representation learning aims to learn a function $f : V \to \mathbb{R}^d$ that projects each node $v \in V$ to a vector in a d-dimensional space $\mathbb{R}^d$, where $d \ll |V|$.*

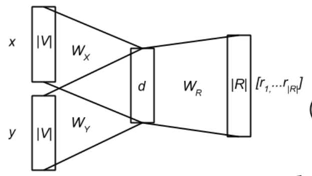


Figure 2: A conceptual model for HIN2Vec

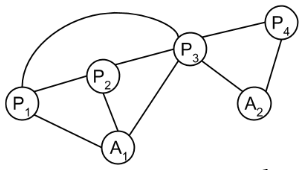


Figure 3: A paper-author HIN

In this paper, we propose a neural network model to tackle the representation learning problem on HIN. Our idea is to explore the rich information and network structure in HIN by capturing multiple relationships (i.e., meta-paths) between nodes and use them as prediction objectives simultaneously and jointly for learning of the node vectors. To realize this idea, nevertheless, we face the following challenges: (1) *Model design.* A well designed NN model is critical to effective and efficient learning of the HIN2Vec framework. A conceptual neural network model, developed in our preliminary design, trains a multi-label classifier to learn node vectors, but it faces excessive overhead in both training data preparation and model learning processes. The proposed HIN2Vec model is a better design. (2) *Regularization.* Due to the semantics and implications of latent vectors in the learning process, proper regularization on certain model parameters are required. (3) *Training data preparation.* Training data need to be prepared and tailored based on the learning logic behind the proposed HIN2Vec model. There is a trade-off, especially for large-scale HINs, between the computational/spacial efficiency and the quality of training data.

4 THE *HIN2VEC* FRAMEWORK

As introduced earlier, the HIN2Vec framework consists of two phases: *Training data preparation* and *Representation learning* (See Figure 1). In the following, we first present our approach for the representation learning phase (Section 4.1), where we discuss a conceptual design of NN model for our framework and its pitfalls, and then introduce the proposed HIN2Vec NN model and some issues to consider. Next, for the training data sampling phase, we propose an efficient training data sampling method, based on ideas of random walks and negative sampling, to generate training data for the proposed model and discuss related issues (Section 4.2).

4.1 Representation Learning

As discussed, our idea to learn node vectors for HIN applications lies in jointly learning a model for multiple prediction tasks, one for each meta-path. Thus, an intuitive approach is to develop an NN model that predicts a set of targeted relationships between any given pair of nodes.

Conceptually, we can adopt a single-hidden-layer feedforward neural network model (as illustrated in Figure 2) that takes a pair of nodes $x, y \in V$ as the input to predict the probabilities $P(r_i|x, y) (i = 1..|R|)$ of relationships between x and y in the target relationship set R as the output. In the model, the input layer takes in two one-hot vectors, $\vec{x}$ and $\vec{y}$ both of length $|V|$, representing x and y. In the latent layer, $\vec{x}$ and $\vec{y}$ are transformed into latent vectors $W'_X \vec{x}$ and $W'_Y \vec{y}$, where W_X and W_Y are two $|V| \times d$ matrices representing the transformation, W'_X and W'_Y are their transpose matrices, and d is the dimensionality of the hidden layer. Finally, in the output

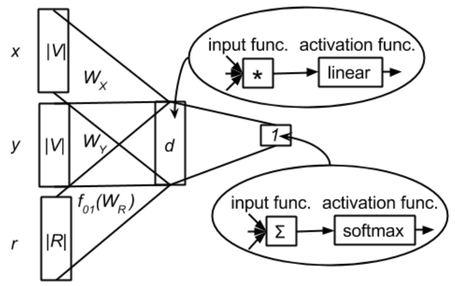


Figure 4: The HIN2Vec NN model

layer, the model outputs a vector of length $|R|$ that predicts the probability of each relationship $r \in R$ between x and y, where W_R is a $|R| \times d$ matrix representing the transformation from the latent layer to output. In summary, this conceptual model can be seen as a multi-label classifier, and the three matrices, W_X, W_Y and W_R, collect the feature vectors of input node pairs and their relationships.[1]

To illustrate the conceptual NN model and issues arising in the HIN2Vec framework, Figure 3 shows a simple HIN consisting of four papers (P) P_1 ... P_4, and two authors (A) A_1 and A_2. In this example, an edge between two papers denotes their citation relationship (we ignore the direction of citations for simplicity) and an edge between a paper and an author denotes the authorship. Let R denote a target relationship set that includes all relationships between nodes within 2 hops, i.e., $R = \{$P-P, P-A, A-P, P-P-P, P-P-A, P-A-P, A-P-P, A-P-A$\}$, specified by meta-paths. To train the conceptual model based on our idea of HIN2Vec, training data need to be prepared for eight prediction tasks corresponding to R. For instance, P_1 and A_1 have two relationships, P-A and P-P-A. A training data entry is $\langle x : P_1, y : A_1, output : [0,1,0,0,1,0,0,0] \rangle$.

In practice, however, an issue arising in preparing the training set for this model is the high cost in scanning the whole network to find all the relationships in R for each pair of nodes in the network. In addition, during the training process, the transformation matrices, i.e., W_X, W_Y and W_R, need to be updated in accordance with *all* the relationships, whether they are in presence (positive) between two input nodes or not (negative).

4.1.1 *The HIN2Vec model.*

To address the aforementioned issues, we propose the *HIN2Vec* NN model that reduces the prediction tasks of the conceptual NN model (i.e., predicting the probabilities of relationships between two nodes) into new prediction tasks *— whether two nodes, x and y, have a specific relationship r*. As to be discussed later, this design avoids scanning for *all* relationships in data preparation as well as examining/updating for *all relationships* during training.

As shown in Figure 4, the HIN2Vec model is a *binary classifier* that takes a pair of nodes x and y, and a certain relationship $r \in R$ as the input to predict whether the relationship r exists between x and y. The input layer takes in three one-hot vectors, $\vec{x}$, $\vec{y}$, and $\vec{r}$, which are transformed into latent vectors $W'_X\vec{x}$, $W'_Y\vec{y}$, and $f_{01}(W'_R\vec{r})$ in the latent layer. Note that the treatment on r is different from x and y due to their different semantics and implications in the learning process. Here we apply a regularization function $f_{01}(.)$ to restrict the values of latent vector for r to be between 0 and 1. We apply Hadamard function, i.e., element-wise multiplication, to aggregate the three vectors, denoted by $W'_X\vec{x} \odot W'_Y\vec{y} \odot f_{01}(W'_R\vec{r})$ and then apply the Identity function for activation. Finally, the output layer, taking Summation as the input function and Sigmoid function for activation, computes $sigmoid\left(\sum W'_X\vec{x} \odot W'_Y\vec{y} \odot f_{01}(W'_R\vec{r}\right)$ to realize logistic classification. Notice that, in the framework, we can regularize W_X and W_Y to be the same matrix or two different matrices, depending on whether we would like to emphasize on the individual roles of input nodes in the relationships. In our experiments, we make W_X and W_Y to be identical, so W_X and W_R consist of learned node vectors and meta-path vectors, respectively.

4.1.2 *Regularization of W_R.*

As discussed, the regularization function $f_{01}(.)$ of W_R regularizes values in W_R within 0 and 1. The goal is two-fold: 1) it avoids negative values in W_R, which would inverse the sign in the multiplication of node vectors, thus interfering the learning of node vectors. Suppose we do not apply $f_{01}(.)$ on W_R, and some values in a relationship vector are negative. The signs of corresponding values in the two node vectors may become opposite to optimize $W'_R(W'_X\vec{x} \odot W'_Y\vec{y})$ in the model. However, if these nodes are actually positively related, the signs of values in these two nodes vectors should be the same so as to stay positive after passing through the Sigmoid function at the output layer. 2) it prevents values in W_R from becoming too large, which decreases the effectiveness of learning. In specific, because HIN2Vec adopts element-wise multiplication to aggregate the three vectors $W'_X\vec{x}$, $W'_Y\vec{y}$, and $W'_R\vec{r}$. If a value of $W'_R\vec{r}$ is greater than 1, it decreases the importance of values of $W'_X\vec{x}$ and $W'_Y\vec{y}$ in the multiplication, thus interfering the learning of node vectors.

Several functions, e.g., Sigmoid function or Binary Step function, may serve the purpose described above. We empirically show that the Binary Step function outperforms Sigmoid.

4.1.3 *Optimization Objective.*

The learning of HIN2Vec model parameters (i.e., node vectors and meta-path vectors) is realized by setting multiple optimization objectives, one for each meta-path in R, to facilitate iterative training upon the training data. It is critical to set the objective functions properly.

To train HIN2Vec, a training set D containing training data in the form of $\langle x, y, r, L(x, y, r) \rangle$ is extracted from the HIN. Here $L(x, y, r)$, a binary value, indicates whether x and y have r. With D, HIN2Vec is trained by the backpropagation training algorithm in conjunction with stochastic gradient descent. It goes backwards to adjust the weights in W_X, W_Y, and W_R for each entry in D, attempting to maximize the objective function O, which is the multiplication of $O_{x,y,r}(x, y, r)$ for each training data entry in D, where $O_{x,y,r}(x, y, r)$ quantifies how HIN2Vec correctly predicts $L(x, y, r)$. To ease the computation in the optimization process, we maximize $\log O$ rather than directly maximize O. The objective function O and derivation of $\log O$ are shown below.

$$O \propto \log O = \sum_{x,y,r \in D} \log O_{x,y,r}(x, y, r)$$

In the above, $O_{x,y,r}(x, y, r)$ quantifies how HIN2Vec correctly predicts a training data $\langle x, y, r, L(x, y, r) \rangle$ based on $P(r|x, y)$, which is the output of HIN2Vec and is the predicted probability that x and

[1]As a conceptual model, here we do not discuss some implementation details such as aggregation and activation operations in detail.

y have the relationship r in the network. Specifically, for a training data entry $\langle x,y,r,L(x,y,r)\rangle$, when $L(x,y,r)$ is 1, $O_{x,y,r}(x,y,r)$ aims to maximize $P(r|x,y)$; otherwise, $O_{x,y,r}(x,y,r)$ aims to minimize $P(r|x,y)$. Thus, $O_{x,y,r}(x,y,r)$, $\log O_{x,y,r}(x,y,r)$ and $P(r|x,y)$ are derived as below,

$$O_{x,y,r}(x,y,r) = \begin{cases} P(r|x,y), & \text{if } L(x,y,r)=1 \\ 1-P(r|x,y), & \text{if } L(x,y,r)=0 \end{cases}$$

$$\log O_{x,y,r}(x,y,r) = L(x,y,r)\log P(r|x,y) + [1-L(x,y,r)]\log[1-P(r|x,y)]$$

$$P(r|x,y) = sigmoid\left(\sum W'_X\vec{x}\odot W'_Y\vec{y}\odot f_{01}(W'_R\vec{r})\right)$$

We then apply the stochastic gradient descent algorithm to maximize the objective function O. Specifically, for each training data entry, $\langle x,y,r,L(x,y,r)\rangle$, it goes backwards to adjust the weights in $W'_X\vec{x}, W'_Y\vec{y}$, and $W'_R\vec{r}$ based on the gradients of $\log O_{x,y,r}(x,y,r)$ differentiated by $W'_X\vec{x}, W'_Y\vec{y}$, and $W'_R\vec{r}$, respectively, by

$$W'_X\vec{x} := W'_X\vec{x} + \frac{d\log O_{x,y,r}(x,y,r)}{dW'_X\vec{x}}$$
$$W'_Y\vec{y} := W'_Y\vec{y} + \frac{d\log O_{x,y,r}(x,y,r)}{dW'_Y\vec{y}}$$
$$W'_R\vec{r} := W'_R\vec{r} + \frac{d\log O_{x,y,r}(x,y,r)}{dW'_R\vec{r}}$$

4.2 Training Data Preparation

We develop an efficient algorithm to sample the HIN while extracting training data for HIN2Vec in the form of $\langle x,y,r,L(x,y,r)\rangle$. Notice that there is a trade-off in data collection between the computational efficiency (e.g., to sample training data randomly instead of enumerating) and the quality (e.g., the training dataset should cover as many nodes pairs with correct semantics in their relationships as possible). Therefore, it's essential to design an efficient data sample extraction scheme for training data preparation.

We apply random walks in the task for several advantages. First, it is computationally efficient for both space and time complexity. The space complexity of storing the direct neighbors of a node to select the next node in a random walk is $O(|V|)$, and the space complexity of storing the path of a random walk with length l is $O(l)$. The time complexity of selecting the next node is $O(1)$ (by some random selection methods, such as Alias method [5, 17]), and the time complexity of generating a path with length l by a random walk is $O(l)$. Moreover, subsequences of a random walk are also random walks, which can be used to generate training data without re-generating new random walks. Suppose we consider all relationships within w-hop neighborhood. Given a random walk of length $l > w$, it can generate w training data for $l-w$ nodes. Thus, the time complexity of generating training data given a random walk is $O(w(l-w))$. Therefore, if we consider 1 to w-hop neighbor relationships, the space complexity and time complexity of generating training data by a random walk with length l is $O(|V|+l)$ and $O(l+w(l-w))$, respectively. When l and w are both much smaller than $|V|$, using random walks to generate training data is scalable for large-scale networks.

Consider the example of paper-author network discussed earlier. Suppose we generate a random walk P_1,P_2,A_1,P_3,A_1. Let the target relationship set R include all relationships with metapath length no greater than 2-hop. We can generate training data for the first node P_1 as $\langle P_1,P_2,\text{P-P}\rangle$ and $\langle P_1,A_1,\text{P-P-A}\rangle$, for the second node P_2 as $\langle P_2,A_1,\text{P-A}\rangle$ and $\langle P_2,A_3,\text{P-A-P}\rangle$, and so on.

There are issues in applying random walks to HIN2Vec data preparation. First, a random walk may give rise to *cycles*, which hurt the quality of the training data, because a path instance with cycles may not comply to the semantics of the corresponding metapath. For example, the third node A_1 in the above random walk generates a training data entry $\langle A_1,A_1,\text{A-P-A}\rangle$ by the path A_1,P_3,A_1. However, an author has no coauthorship with herself. Thus, we eliminate data entries with any cycle by checking duplicate nodes.

Second, random walks are used to sample positive data, but the model also needs negative data for learning. Thus, while generating positive samples via random walks, we also generate negative data entries following the ideas of *negative sampling* in Word2Vec [21]. For each sampled positive entry, $\langle x,y,r\rangle$, we generate negative data entries by randomly replacing one of the three values with other x', y', or r', where x' and y' are randomly selected nodes, and r' is a randomly selected relationship from R. A generated negative data entry, $\langle x'',y'',r''\rangle$ indicates that x'' and y'' are not expected to have a certain relationship r''. However, for HIN2Vec, more erroneous negative data (which actually a positve data) might be generated by replacement of r compared with by replacement of x or y since $|V|$ is usually much larger than $|R|$, while $|R|$ is usually not very large (tens to hundreds). To maintain efficiency and the cleanness of the negative samples, we only sample negative data by randomly replacing x or y, and filter out erroneously generated positive samples. To do so, we also ensure that the randomly selected x' or y' has the same node type as x or y. For example, for a positive data $\langle P_1,P_2,\text{P-P}\rangle$, we randomly select a paper, says P_3, to replace P_2 (or P_1) to generate a negative data that indicates P_2 (or P_1) does not have a citation relationship with P_3. In other words, we do not select a node with a different node type, says authors, for replacement. We empirically examine the effect of avoiding cycles in random walks and ensuring the same node type in negative sampling.

Table 1: Statistics of Datasets

Blogcatalog	User	Group		
	10312	39		
Yelp	User	Business	City	Category
	630639	86810	10	807
DBLP	Paper	Author	Venue	
	53464	54949	20	
U.S. Patents	Patent	Inventor	Assignee	Class
	295145	293848	31805	14

5 EXPERIMENTS

In this section, we conduct a comprehensive evaluation on HIN2Vec. We first introduce four real-world HINs used for experiments and six models for representation learning on networks. Then, we evaluate HIN2Vec and those models by two applications: *multi-label classification for nodes* and *link prediction for edges.*

5.1 Datasets and Models for Evaluation

Our evaluation involves two social network datasets (Blogcatalog and Yelp) and two scientific publication datasets (DBLP and U.S. Patents). Some statistics of the HINs extracted from these datasets are summarized in Table 1.

Blogcatalog is a blog dataset of Blogcatalog, released by Arizona State University [1]. We use all bloggers (U) and their groups (G) as nodes to form a social network, which contains friendships (U-U) and users' groups (U-G) as edges.

Yelp is a social media dataset, released in Yelp Dataset Challenge [4]. We extract data of the top 10 cities with the most businesses to form a network, which includes users (U), businesses (B), cities(C) and categories (T) as nodes, and friendships (U-U), users' reviews (B-U), businesses' cities (B-L) and businesses' categories (B-C) as edges. We filter small categories with less than 10 businesses.

DBLP is a bibliographic dataset in computer science [3]. We extract papers published between 1994 to 2014 of 20 conferences in 4 research fields to form a network [2]. The network includes papers (P), authors (A) and venues (V) as nodes, and authorships (P-A), papers' venues (P-V) as edges.

U.S. Patent is a patent dataset of United States Patent and Trademark Office (USPTO) [2]. We extract patents issued between 1998 to 2012 in 14 drug related patent classes to form a network [3]. The network contains patents (P), inventors (I), assignees (A) and patent classes (C) as nodes, and inventorships (P-I), patents' assignees (P-A), patents' classes (P-C) and citations (P$\rightarrow$P) as edges.

We evaluate HIN2Vec against six state-of-the-art representation learning models. Among them, DeepWalk, LINE and node2vec, are designed for homogeneous information networks. The are applied by treating all node and edges in the HINs as homogeneous ones.

DeepWalk [24] learns d-dimensional node vectors by capturing node pairs within w-hop neighborhood via *uniform* random walks in the network.

LINE [28] learns node vectors by considering first- and second-order proximities of nodes in a network separately. We apply LINE to learn $d/2$ dimensions by capturing first-order information, and the other $d/2$ dimensions by capturing the second-order information, and then use both to form d-dimensional node vectors.

node2vec [10] is generalized from DeepWalk. It learns d-dimensional node vectors by capturing node pairs within w-hop neighborhood via *parameterized* random walks.

PTE [27] decomposes an HIN to a set of bipartite networks by edge tyes, and learns d-dimensional node vectors by capturing 1-hop neighborhood of the resulting bipartite networks.

HINE [13] learns d-dimensional node vectors by capturing path counts or path constrained random walks [19] of node pairs within w-hop neighborhood.

ESim [25] learns d-dimensional node vectors by paths between nodes along a given set of meta-paths. This model also learns meta-path vectors, which is used to shift node vectors in the designed objective function.

[2]20 conferences are "AAAI", "CVPR", "ECML", "IJCAI", "SIGMOD", "VLDB", "PODS", "EDBT", "ICDE", "ICDM", "KDD", "PAKDD", "PKDD", "SDM", "ECIR", "SIGIR", "WSDM", "WWW", "CIKM", that belong to 4 research fields, including DM, DB, IR, and ML.

[3]Drug related patent classes are 128, 351, 433, 424, 435, 623, 514, 600, 601, 602, 604, 606, 607, 800.

5.2 Multi-label Classification of Nodes

In this section, we evaluate the models by multi-label classification of nodes. We first introduce the experimental setup, including the process of classification, the preparation of labeled datasets and the default settings of parameters in the compared models. Then, we perform sensitivity tests on parameters of HIN2Vec to determine their default settings. Next, we examine several issues in the HIN2Vec framework, including the regularization on meta-path vectors W_R, cycles in random walks, and negative sampling with the same node type. Finally, we show the experimental results and analyze the meta-path vectors learned by HIN2Vec.

5.2.1 *Experimental Setup.*

After learning the node vectors, we select a set of nodes, which are assigned one or more labels from a finite label set, and use their representations as feature vectors to learn and test a linear SVM classifier with five-fold cross validation. We use *micro-*f_1 score and *macro-*f_1 score as metrics for evaluation.

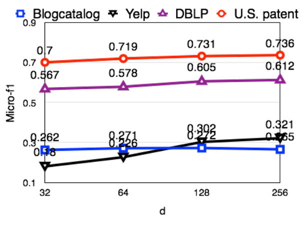
In Blogcatalog, users are categorized by 39 groups, which serve as labels. Thus, all users are included in the labeled dataset for user group classification. In Yelp, as the majority of businesses are restaurants, we select 10 main cuisines in restaurants' categories as labels, and select 13,111 restaurants with at least one of those labels to form a labeled dataset for restaurant type classification [4]. In DBLP, we use the 4 research fields as labels, and assign an author with a field if he/she has a publication in a conference in that field to form a labeled dataset for author classification. Finally, for U.S. Patents, we use 14 drug related patent classes as labels, and involve all patents to form a labeled dataset for patent classification. For these node classifications, we eliminate from HINs, groups in Blogcatalog, categories in Yelp, venues in DBLP and patent classes in U.S. Patents, when we learn the representation of nodes.

Regarding default parameters, the dimensionality of node vectors, d, is set to 128 for all approaches. The negative sampling rate n is set to 5. The initial learning rate α in stochastic gradient descent is 0.025. The context window for DeepWalk and node2vec is set to 4 because they achieve good performance. Also, we use all meta-paths with length $w \leq 4$ for HINE, ESim and HIN2Vec. For ESim, we set the weight of training data sampling of each meta-path based on its length. For example, a meta-path with length l, its sampling weight is $1/l$. We also try equal weighting of each meta-path, but weighting by meta-path length performs better. For node2vec, the two parameters q and p for parameterized random walks are set to 4 and 1, respectively. The number of training data sampling or the length of random walks for generating training data varies for each model and for each dataset, in order to obtain converged results.

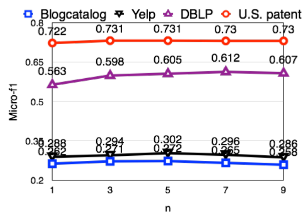
5.2.2 *Parameter Tuning in HIN2Vec.*

Parameters settings in HIN2Vec affect the node representation learning and the application performance. To decide the default settings, we vary the values of important parameters to observe how the *micro-*f_1 changes in node classification in the networks. The results are shown in Figure 5.

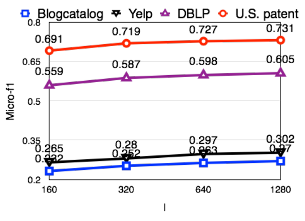
[4]10 main cuisines are "American", "Mexican", "Italian", "Chinese", "Japanese", "Thai", "Indian", "Canadian", "Middle Eastern" and "Greek"



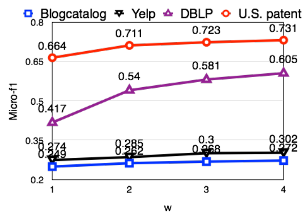
(a) Dimensionality d



(b) Number of Negative Samples n



(c) Length of Random Walks l



(d) Length of Meta-paths w

Figure 5: Parameter Tuning

No. of dimensionality. First of all, Figure 5(a) shows that setting the number of dimension d at 128 is reasonable. General speaking, a small d is not sufficient to capture the information embedded in relationships between nodes, but a large d may lead to noises and cause overfitting. In Blogcatalog, the best performance is achieved when d is 128. In the other three networks, increasing d up to 256 continuously improves the performance, though their improvements are small (about 0.7% to 6.2%). Generally, a larger network may need a larger d to capture the information embedded in relationships between nodes.

No. of negative samples per positive sample. As shown in Figure 5(b), the performance does not change much when the number of negative sampling per positive sample n is set at between 3 to 7. Thus, setting n to 5 is a good choice.

Length of Random Walks. A longer random walk can generates more sample data. Figure 5(c) suggests that when the length of random walks l is increased (resulting in more sample data), the performance continues to improve and converge when l is set to 1280 or 2560.

Length of Meta-Paths. Finally, we study the impact of the maximum length w of meta-paths of interest. Figure 5(d) shows that the maximum length of meta-paths does not affect the performance significantly in Blogcatalog, Yelp and U.S. Patents, because that 1-hop and 2-hop relationships, e.g., friendships in Blogcatalog and citations in U.S. Patents, are the most important relationship in these networks. However, a larger w is still helpful in these networks, and the performance improves by 9% when w is 4. On the other hand, capturing meta-paths with larger w is crucial in DBLP because some longer meta-paths have important semantic meaning. For examples, A-P-A-P-A (two authors have co-authorship with the same author) in DBLP would help the classification as these two authors are likely to be in the same research field.

Based on these results, for HIN2Vec, the dimensionality of node representation, d, is set to 128, the negative sampling rate n is set to 5, and to use all meta-paths with length $w \leq 4$, because that these parameter values can achieve good performance in the all

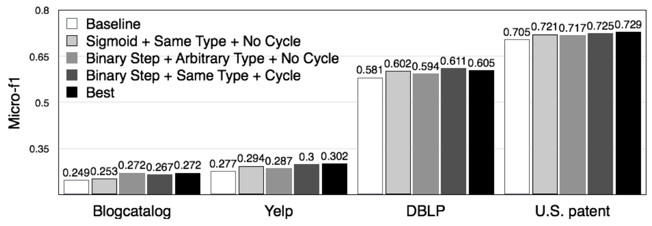


Figure 6: Comparison of approaches to issues in HIN2Vec

networks. Also, these parameter values are set for the fair comparison with the compared models. The length of random walks in HIN2Vec is set to 1280 to obtained converged performance.

5.2.3 Study of unique issues in HIN2Vec. As discussed previously, there are several unique issues arising due to HIN data preparation and HIN2Vec design. In this section, we examine the issues of i) regularization functions on relationship vectors W_R, ii) negative sampling with the same node types, and iii) cycle elimination in random walks. For (i), HIN2Vec uses the Binary Step function instead of Sigmoid; for (ii), HIN2Vec ensures negative sampling choosing the same types of nodes as the corresponding positive samples rather than arbitrary node types, and for (iii), HIN2Vec eliminates cycles from random walks instead of leaving them in the data samples. We perform experiments to compare the different choices in these issues and justify our decisions.

Specifically, we compare 5 settings. First, the Baseline setting adopts Sigmoid function (Sigmod), uses arbitrary node types in negative sampling (Arbitrary Type), and has cycles in random walks (Cycle). On the other hand, the Best setting applies Binary Step function (Binary Step), ensures the same node type in negative sampling (Same Type), and eliminates cycles in random walks (No Cycle). Three additional settings regarding particular issues are tested: a) Sigmod+ Same Type + No Cycle, b) Binary Step+ Arbitrary Type + No Cycle, and c) Binary Step + Same Type + Cycle. Figure 6 shows that the Best setting used in HIN2Vec performs the best, outperforming the Baseline setting in all the networks by about 3.4% to 9.2%.

Regarding the issue of regularization functions, we find that using Binary Step is better than using Sigmoid in all networks, by about 0.5% to 7.4%. The main difference is that Sigmoid function regularizes values to a continuous number between 0 to 1 and Binary Step function regularizes values to 0 or 1 by the threshold 0.5. When a value in a meta-path vector is regularized to a number between 0 to 1 by Sigmoid, it amplifies the importance of the multiplication of two input node vectors, thus mis-guiding representation learning.

The result also shows that Same Type is better than Arbitrary Type, by a margin of about 1.6% to 5% (excluding Blogcatalog network because that it only contains one node type, and thus does not have this issue.) The result suggests that ensuring the same node type in negative sampling is necessary.

Finally, No Cycle and Cycle do not show significant difference. Actually, eliminating cycles in random walks does not always perform better than leaving cycles in training data. Perhaps this is because the number of erroneous data is relatively small. However, our experiments show that avoiding the training data with cycles improve the efficiency while maintaining the effectiveness. Thus, HIN2Vec chooses to remove cycles from training data.

5.2.4 Evaluation of models. The performance of node classification by all evaluated models is summarized in Table 2. HIN2Vec outperforms all the state-of-the-art models. As shown, the improvement ratio (compared with the best of these existing models, marked by '*') ranges from 6.4% to 23.8%. We have the following observations from the comparison.

Exploring meta-paths of longer length is useful. Compared with LINE and PTE, which only capture 1-hop or 2-hop neighborhood of nodes, other models usually have better performance (except ESim) because they also capture "longer" relationships between nodes.

Distinguishing different meta-paths between nodes is useful. Comparing with DeepWalk, node2vec, HINE and HIN2Vec, all of which consider relationships of nodes within w(=4)-hop neighborhood, HIN2Vec further distinguishes different relationships between nodes, whereas others only capture aggregated information. Thus, HIN2Vec is able to capture more detailed information of network structure and thus can achieve higher performance in node classification.

Appropriate model design to distinguish relationships between nodes is crucial. Comparing with ESim which also captures meta-path relationships between nodes, HIN2Vec outperforms ESim in all four networks. We argue the superiority of HIN2Vec is due to a better model design that precisely captures the relationships between nodes.

5.2.5 Analysis of Meta-paths. In addition to node vectors, HIN2Vec also learns meta-path vectors as a side-product. In this section, we analyze the representations of relationships (i.e., meta-path vectors) by clustering those vectors to examine whether meta-paths with relevant semantics are clustered together. We use Yelp and U.S. Patent networks as the study cases because they have more complex schema of networks, with more meta-paths than Blogcatalog and DBLP. We select top 20 meta-paths in these two networks with the highest frequency when generating the training data by random walks, and then apply K-Means algorithm to group meta-paths into 6 clusters. The results are shown in Table 3.

In Yelp, we observe that each of clusters 1 to 5 have clear and distinct semantics. Specifically, Cluster 1 groups meta-paths of friendships (except 3-hop friendship). Cluster 2 groups meta-paths between two businesses via only friendships (except B-U-U-B which is not in top 20 frequent meta-path set). Cluster 3 contains meta-paths between business to users via multiple hops of friendships. In Cluster 4, B-U has unique semantics that no other meta-path is similar with it. Cluster 5 contains meta-paths with cities. Cluster 6 contains several meta-paths that do not share semantics, and due to the lack of space, we skip to show meta-paths in Cluster 6.

Among the 20 meta-paths in the U.S. Patent network, each of clusters 1 to 5 also shows obvious semantics. Specifically, Clusters 1 contains co-citing and co-cited meta-paths, which are relevant because two patents are likely in the same research field if they are co-cited or co-cite the same patents. Cluster 2 contains 1-hop and 2-hop citation meta-paths, which both indicate knowledge flow among patents. Cluster 3 has meta-paths describing the relationships between inventors and his/her citing and cited patents. In Cluster 4, the patents' assignees, P-A, has unique semantics, and thus no other meta-path is similar to it. Cluster 5 contains relationships between patents via bibliographic information (inventors and assignees) (except P-I). However, Cluster 6 also contains several meta-paths with no similar semantics.

These analyses suggest that the representations of meta-paths in HIN2Vec capture semantics of meta-paths very well, placing meta-path vectors in the latent space closely with other vectors of relevant semantics.

5.3 Link Prediction

In this section, we demonstrate that the same node vectors learned by HIN2Vec can be used effectively for another application, *link prediction.* We first introduce the experimental setup including the experimental flow of link prediction and metrics for evaluation. Then, we study different vector functions that transform two node vectors into one vector for the purpose of link prediction. Finally, we show the experimental results of HIN2Vec, in comparison with other models.

5.3.1 Experimental Setup. We model the link prediction problem as a recommendation problem that aims to rank node pairs in terms of their relevancy, which may lead to potential linkage between them. Specifically, given a network, we first generate a sub-network by selecting an edge class and randomly removing a certain fraction (20% in our experiments) of edges of the selected edge class as missing edges. After removing edges, we apply representation learning models on the resulting sub-network. Then, to perform the recommendation on the sub-network, we apply supervised models to rank node pairs which are more likely to have missing edges. In specific, we randomly select 2000 nodes to form a training set. For a selected node, we first use its neighbors within a certain number of hops as candidates to form a set of candidate node pairs (a pair contains the node and a candidate node). This candidate selection step aims to reduce the number of node pairs to be ranked. We choose an appropriate hop number for different networks to ensure the coverage of missing edges is greater than 85%. To form a training set, for each node pair, we label it with 1 if they have a missing edge, and with 0, otherwise. We study different vector functions (to be discussed later) to construct a feature vector for the node pair based on the two nodes' representations. Finally, we apply linear SVM to perform ranking, by using the predicted confidence of each data with five-fold cross validation. We use Mean Average Precision (MAP) and The top-k recall (recall@k) as metrics for evaluation.

(1) Mean Average Precision (MAP): the mean of the average precision scores for the ranking result of each node v. The average precision score (AveP) is the sum of the top-k precision (precision@k), the percentage of top-k ranking results that hit the ground truth over k. The equations of AveP and MAP are shown below.

$$AveP(v) = \Sigma_{k=1}^{n} \frac{precision@k}{k} \qquad MAP = \frac{\Sigma_{i=1}^{K} AveP(v_i)}{K}$$

(2) The top-k recall (recall@k): the percentage of the ground truth or indicators ranked in the top-k returned results.

$$recall@k = \frac{\#\ of\ hits\ in\ top{-}k}{\#\ of\ positive\ documents\ in\ theground\ truth}$$

To construct a feature vector for a node pair based their node vectors, we study four vector functions, *Hadamard, Average, Minus*

Table 2: Performance Evaluation of Node Classification

	Blogcatalog		Yelp		DBLP		U.S. Patents	
	micro-f1	macro-f1	micro-f1	macro-f1	micro-f1	macro-f1	micro-f1	macro-f1
DeepWalk	0.244	0.140	0.276	0.165	0.481	0.463	0.675	0.676
LINE	0.239	0.128	0.270	0.163	0.449	0.429	0.66	0.663
node2vec	0.246	0.141	0.276	0.166	0.491	0.470	0.676	0.677
PTE	0.179	0.096	0.222	0.130	0.417	0.394	0.547	0.555
HINE	*0.250	*0.144	*0.278	*0.169	0.475	0.461	*0.681	*0.685
ESim	0.207	0.102	0.229	0.132	*0.514	*0.496	0.610	0.562
HIN2Vec	**0.272(9.9%)**	**0.158(11.3%)**	**0.302(7.9%)**	**0.192(12.0%)**	**0.605(23.8%)**	**0.594(20.1%)**	**0.729(6.6%)**	**0.732(6.4%)**

Table 3: Clusters of Meta-paths

Cluster	Yelp	U.S. Patents
1	U-U, U-U-U, U-U-U-U-U	P→P←P, P←P→P
2	B-U-B, B-U-U-U-B	P←P, P←P←P
3	B-U-U, B-U-U-U-U	I-P→P, I-P←P
4	B-U	P-A
5	B-C, B-C-B, U-B-C	P-A-P, P-I-P, P-I
6	...	...

Table 4: Vector Functions of Node Pairs

Functions	Hadamard	Average	Minus	Abs. Minus
Description	$\vec{v}_{1i} * \vec{v}_{2i}$	$\frac{\vec{v}_{1i}+\vec{v}_{2i}}{2}$	$\vec{v}_{1i} - \vec{v}_{2i}$	$\lvert\vec{v}_{1i} - \vec{v}_{2i}\rvert$

and *Absolute Minus*. The equations are shown in Table 4. Specifically, Hadamard function is the element-wise multiplication of two vectors. Average function obtains the centroid of two vectors in the latent space. Minus function yields the difference between two vectors in the latent space. Finally, Absolute Minus function calculates the distance of the two vectors in each dimension of the latent space.

Regarding the edge class of missing edges in each network for experiments, we choose friendships (U-U) in Blogcatalog, users' business reviews (B-U) in Yelp, authorships (P-A) in DBLP and patent citations (P→P) in U.S. Patent network.

Regarding default settings, we use the same parameter values used in node classification experiments. Additionally, the default feature vector function for node pairs is Hadamard function because it has the best performance for all the models.

5.3.2 *Vector Functions for Node Pairs.*

In this section, we study the effect of aforementioned vector functions for node pairs and summarize their performance in Table 5. First of all, Hadamard function outperforms all the other functions because it matches the objective function of HIN2Vec (and other models). In specific, if two nodes are more relevant (i.e., with more relationships between them), the sum of element-wise multiplication of their vectors should be larger (i.e., their vectors are closer in the latent space). This is also the reason why Absolute Minus performs well because it captures the distance in each dimension of the two nodes' vectors. However, Average function and Minus function do not have good performance. Average function gets the centroid of two nodes' vectors but does not well capture the similarity between the two nodes (i.e., the closeness of the two nodes in the latent space). For example, two pairs of nodes may have the same centroid, but one pair is distant and the other pair is close. On the other hand, Minus function yields the difference between two node vectors but does not capture their similarity. For example, two pairs of nodes can be similar (close in the latent space) but quite different in terms of their node vectors.

5.3.3 *Evaluation.*

The performance of link prediction is summarized in Table 6. ESim fails in Yelp and U.S. Patent networks, because it exhausts all memory (320G) on our server. HIN2Vec outperforms the existing representation learning models, and the improvement ratio (compared with the best one of the existing works, which is marked by '*') varies from 5.0% to 70.8% for MAP, and 10.8 to 24.3% for recall@100. We have the following observations.

Capturing meta-paths of longer length is useful for link prediction in complex HINs. Compared with LINE and PTE, which only capture 1-hop or 2-hop neighborhood of nodes, other models usually have better performance in most of the networks (except for Blogcatalog), perhaps because they captures relationships between nodes with larger hop number. For friendship recommendation in Blogcatalog, 1-hop or 2-hop friendship in the network already provides useful information, consistent with the notion that two users are more likely to be friends if they have multiple common friends. Thus, capturing friendships with larger hop numbers does not help in other models except for HIN2Vec. The reason why HIN2Vec still outperforms LINE in Blogcatalog is that not only it captures multiple hop relationships, but also it distinguishes them.

Distinguishing different relationships between nodes is useful. Compared with DeepWalk, node2vec, HINE, HIN2Vec distinguishes relationships between nodes, rather than only capturing aggregated information. Thus, when applying SVM to train a model, the model can increase or reduce the feature weights of the dimensions of the relationship if it is helpful or not for prediction, respectively. Thus, it achieves higher performance in link prediction. This is also the reason why HIN2Vec outperforms LINE in Blogcatalog.

6 CONCLUSIONS

This study focuses on representation learning in HINs. Prior works in representation learning in networks only consider limited types of relationships among nodes, or only capture aggregated information of relationships. To fill in this gap, we design a novel neural network model, HIN to Vectors (HIN2Vec), that enables users to capture rich semantics of relationships and the details of the network structure to learn representations of nodes in HINs. Moreover, the proposed model also learn representations of meta-paths,

Table 5: Performance Evaluation of Vector Functions

	Blogcatalog		Yelp		DBLP		U.S. Patents	
	MAP	recall@100	MAP	recall@100	MAP	recall@100	MAP	recall@100
Hadamard	**0.141**	**0.279**	**0.028**	**0.138**	**0.265**	**0.751**	**0.176**	**0.602**
Average	0.074	0.245	0.004	0.033	0.005	0.124	0.008	0.063
Minus	0.050	0.171	0.004	0.030	0.004	0.114	0.009	0.059
Abs. minus	0.130	0.238	0.023	0.119	0.249	0.750	0.130	0.540

Table 6: Performance Evaluation of Link Prediction

	Blogcatalog		Yelp		DBLP		U.S. Patents	
	MAP	recall@100	MAP	recall@100	MAP	recall@100	MAP	recall@100
DeepWalk	0.124	0.227	*0.021	0.110	0.230	*0.710	0.093	0.500
LINE	*0.134	*0.249	0.017	0.104	0.086	0.580	0.091	0.400
node2vec	0.125	0.229	*0.021	*0.111	*0.231	*0.710	0.095	*0.503
PTE	0.067	0.139	0.004	0.034	0.071	0.324	0.030	0.243
HINE	0.085	0.179	0.016	0.097	0.205	0.697	*0.103	0.495
ESim	0.132	0.185	x	x	0.179	0.633	x	x
MPE	**0.141(5.0%)**	**0.279(10.8%)**	**0.028(31.8%)**	**0.138(24.3%)**	**0.265(12.8%)**	**0.751(5.8%)**	**0.176(70.8%)**	**0.602(19.9%)**

which can be used for meta-path analysis. Empirically, we demonstrate that the proposed HIN2Vec model is able to automatically learn feature vectors for nodes in HINs to support a variety of HIN applications, including multi-label node classification and link prediction, in several real-world networks. HIN2Vec also soundly outperforms all the compared models in those experiments.

As for our next step, we plan to explore the regularization for representations of nodes to learn sparse representations, which may capture more distinct latent topics of each nodes and meta-paths.

ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation under Grant No. IIS-1717084 and SMA-1360205.

REFERENCES

[1] 2009. BlogCatalog3. http://socialcomputing.asu.edu/datasets/BlogCatalog3. (2009).
[2] 2014. USPTO PatentView. http://www.dev.patentsview.org/workshop/participants.html. (2014).
[3] 2017. How can I download the whole dblp dataset. http://dblp.uni-trier.de/faq/How+can+I+download+the+whole+dblp+dataset. (2017).
[4] 2017. Yelp dataset. https://www.yelp.com/dataset_challenge. (2017).
[5] Joachim H Ahrens and Ulrich Dieter. 1989. An alias method for sampling from the normal distribution. *Computing* 42, 2-3 (1989), 159–170.
[6] Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *Science* 286, 5439 (1999), 509–512.
[7] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 8 (2013), 1798–1828.
[8] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C Aggarwal, and Thomas S Huang. 2015. Heterogeneous network embedding via deep architectures. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 119–128.
[9] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. 2012. Multi-column deep neural networks for image classification. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR 2012)*. IEEE.
[10] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. (2016).
[11] Huan Gui, Jialu Liu, Fangbo Tao, Meng Jiang, Brandon Norick, and Jiawei Han. 2016. Large-Scale Embedding Learning in Heterogeneous Event Data. (2016).
[12] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, and Tara N Sainath. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine* 29, 6 (2012), 82–97.
[13] Zhipeng Huang and Nikos Mamoulis. Heterogeneous Information Network Embedding for Meta Path based Proximity. *arXiv preprint arXiv:1701.05291* (????).
[14] Ming Ji, Jiawei Han, and Marina Danilevsky. 2011. Ranking-based classification of heterogeneous information networks. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD 2011)*.
[15] Jon M Kleinberg. 2000. Navigation in a small world. *Nature* 406, 6798 (2000).
[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS 2012)*. 1097–1105.
[17] Richard A Kronmal and Arthur V Peterson Jr. 1979. On the alias method for generating random variables from a discrete distribution. *The American Statistician* 33, 4 (1979), 214–218.
[18] Ni Lao and William W. Cohen. 2010. Relational Retrieval Using a Combination of Path-constrained Random Walks. *Machine Learning* 81 (Oct. 2010), 53–67.
[19] Ni Lao and William W Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine learning* 81, 1 (2010), 53–67.
[20] David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. *Journal of the American society for information science and technology* 58, 7 (2007), 1019–1031.
[21] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS 2013)*. 3111–3119.
[22] Abdel-rahman Mohamed, George E Dahl, and Geoffrey Hinton. 2012. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing* 20, 1 (2012), 14–22.
[23] Tore Opsahl and Pietro Panzarasa. 2009. Clustering in weighted networks. *Social networks* 31, 2 (2009), 155–163.
[24] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD 2014)*. ACM, 701–710.
[25] Jingbo Shang, Meng Qu, Jialu Liu, Lance M Kaplan, Jiawei Han, and Jian Peng. 2016. Meta-Path Guided Embedding for Similarity Search in Large-Scale Heterogeneous Information Networks. *arXiv preprint arXiv:1610.09769* (2016).
[26] Yizhou Sun, Rick Barber, Manish Gupta, Charu C. Aggarwal, and Jiawei Han. 2011. Co-author Relationship Prediction in Heterogeneous Bibliographic Networks. In *Proc. of the 2011 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2011)*. Kaohsiung, Taiwan, 121–128.
[27] Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD 2015)*.
[28] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale information network embedding. In *Proceedings of the International Conference on World Wide Web (WWW 2015)*. ACM, 1067–1077.
[29] Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of small-world networks. *Nature* 393, 6684 (1998), 440–442.